UNIVERSITY OF GRANADA

DEPARTMENT OF COMPUTER SCIENCE
AND ARTIFICIAL INTELLIGENCE

PHD PROGRAM IN INFORMATION AND
COMMUNICATION TECHNOLOGIES

PhD THESIS DISSERTATION

Constrained Clustering:

Taxonomy, New Optimization Models, and Hybridizations with Singular Problems of Machine Learning

PhD CANDIDATE
Germán González Almagro

PhD ADVISORS
Salvador García López & José Ramón Cano de Amo

Granada, May 2023

Editor: Universidad de Granada. Tesis Doctorales
Autor: German González Almagro
ISBN: 978-84-1117-878-8
URI: https://hdl.handle.net/10481/82216

El doctorando / *The PhD candidate* **Germán González Almagro** y los directores / *and the advisors* **Salvador García López & José Ramón Cano de Amo**

Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y, hasta donde nuestro conocimiento alcanza, en la realización del trabajo se han respetado los derechos de otros autores a ser citados cuando se han utilizado sus resultados o publicaciones.

Guarantee, by signing this doctoral thesis, that the work has been done by the doctoral candidate under the direction of the thesis supervisors and, as far as our knowledge reaches, the rights of the authors to be cited (when their results or publications have been used) have been respected in the performance of this work.

Granada, May 2023.

The PhD candidate:

Sgd.: Germán González Almagro

The PhD advisor:

Sgd.: Salvador García López

The PhD advisor:

Sgd.: José Ramón Cano de Amo

Funding

This doctoral thesis has been funded by the following research projects: the predoctoral scholarship PREDOC_01648, the research projects PP2016.PRI.I.02 and PP2019.PRI.I.06, the national research projects TIN2017-89517-P and PID2020-119478GB-I00, and the regional research project A-TIC-434-UGR20.

«*To learn which questions are unanswerable, and not to answer them: this skill is most needful in times of stress and darkness.*».

- Ursula K. Le Guin, The Left Hand of Darkness.

«*First electricity, now telephones. Sometimes I feel as if I'm living in an H.G. Wells novel.*».

- Lady Violet Crawley, The Dowager Countess of Grantham. Downton Abbey.

Agradecimientos

No exagero cuando digo que llevo pensando cómo escribir los agradecimientos de esta tesis prácticamente desde que decidí hacer el doctorado. No son pocas las ocasiones en las que me he visto a mí mismo dando vueltas en la cama pensando en esto, o mirando al infinito mientras repasaba nombres en mi cabeza. Total, para que al final, aunque parezca mentira, haya llegado el día de acabar de escribirla y siga sin saber qué poner aquí. No es que no tenga claro a quién quiero nombrar, al contrario, lo tengo cristalino. Lo que me pasa es que no sé cómo mostrarle a todo el mundo el agradecimiento que se merece. Un doctorado es un proyecto enorme, imposible de afrontar solo, y solamente puede superarse si te rodeas de la gente adecuada. La aportación de todas las personas que quiero mencionar merece muchísimo más que un nombre impreso en el prefacio de este documento, pero hacerles justicia me llevaría más páginas de las que restan del libro que tienes en las manos. Por eso, he optado por nombrar a todos los que pueda, a todos de los que me acuerde. A todos ellos les pido perdón por tan escaso reconocimiento en comparación a su labor. Este logro es tan vuestro como mío.

Como no podía ser de otra forma, empiezo por agradecer a mis directores, Salva y José Ramón, el esfuerzo, la dedicación y la confianza que me han brindado. Innumerables consejos y reuniones fundamentan este agradecimiento y puedo decir, sin lugar a dudas, que este proyecto llega a su fin con éxito gracias a ellos. Con Salva no solo me he reunido para hablar de esta tesis, no. También nos hemos reunido para desayunar, ir a conciertos, planificar mi futuro académico o jugar a juegos de mesa con otros compañeros. La cercanía y la sinceridad con las que siempre me ha tratado han hecho de él un magnífico director. Dentro del ámbito académico, no puedo dejar de agradecer a Dani Peralta y Eli de Poorter, que me dieron la mejor acogida que podía desear durante mi breve estancia en la preciosa ciudad de Gante.

Me gustaría continuar dándoles las gracias a todos mis compañeros y amigos del trabajo. Desde los compañeros del DB-4 del CITIC, José, Jesús, Gustavo, Sergio (González) y Nacho (Moya), ya doctores todos, hasta mis actuales compañeros de lo que era el DB-5 y adjuntos, Juanlu, Laura (Hernández), Paco, Iván, Nuria, Elena, Nacho (Aguilera), José Daniel, Guille, Víctor y Prá. A todos ellos les agradezco toda la ayuda y consejos que en algún momento me han prestado y, desde luego, los interminables desayunos en la cafetería o en la sala de espera del nuevo despacho. En especial, quiero agradecer a Juanlu, que es coautor de varios de mis artículos, que ha pasado innumerables horas conmigo en el CPD, en plena pandemia, intentando que Hércules funcionara; y con quien viví la improbable aventura que nos llevó a conocer Japón.

Quiero dar las gracias también a todos mis amigos fuera del ámbito académico. Al ir a escribir sus nombres, me he sorprendido de que sean tantos, pero intentaré no dejarme a nadie fuera. A los que me acompañan desde el instituto (aunque no estudiáramos todos en el mismo) y que hoy considero mis hermanos, Sergio (Carrasco), Javi (Fernández), Antonio (Milán), Alberto, Yesu y Antonio (Maldonado). A los que conocí durante la carrera, Laura (Calle), Javi (León) y Cris, que tanto me han ayudado y enseñado en esa y otras etapas y que siguen siendo una parte fundamental de mi vida. A mis amigos de las Magic, Tony, Juanma y Marta, con los que tantas horas he pasado girando cartas sobre una mesa y con los que tantas horas arrebatadas por la pandemia tengo que recuperar. Y, finalmente, a un grupo que no sé como llamar, pero a los que les debo poder ser quien soy hoy; imprescindibles para mí en tantos y tantos aspectos, Pablo (Baeyens), Toni, Nacho (Casares), Antonio (Martín), Chelu, Irene, Meru, Pablo (Sánchez), Sergio (Castro), José María, Laura (Kiràly), Alejandro y, desde luego, a Eva, con la que contraje una deuda insaldable el día que me los presentó a todos, y que me enseñó cuánto te puede cambiar la vida la gente buena. A todos vosotros, gracias, de corazón. Todo el tiempo que pase con vosotros siempre me sabrá a poco.

Aquí quiero darles las gracias a todas las personas que, sin ser coautores de la tesis o de los artículos del compendio, de una forma u otra, han colaborado en su elaboración. A Javi (León) y a Fede por todas las horas que han dedicado a leer y corregir este documento. A Toni, por el diseño de la portada y otras cuestiones artísticas. A Diego y a Gustavo, por su ayuda con el formato y el código LaTeX de este documento. Y a Elena, por darme la idea (y el código) para sacar una de las gráficas mas ilustrativas de esta tesis. A todos vosotros, gracias por vuestra ayuda.

Ya termino, y no puede ser de otra forma. Este último lugar esta reservado para las personas que me llevan de la mano por la vida, a las personas sin las que no entiendo mi mundo. A mi familia, mis padres, Carmina y Germán, y a mi hermano Nacho—que siempre, siempre, sin excepción—me han apoyado y ayudado, por difícil que fuera la tarea o el momento, y que tantos sacrificios han hecho por mí; a ellos les debo la vida. A mis padrinos, Tatá y Ká, que tantos recuerdos ocupan en mi memoria. A mi abuela Malli y mi abuelo Pepe, que siguen aquí conmigo, donde siempre han estado. A mi abuelo Germán, al que debo mi nombre y que ya hace tanto que nos dejó. A mi abuela Illa, que me vio empezar esta tesis pero no ha podido verme terminarla, y a la que, con el buen humor que la caracterizaba, tantos pensamientos me sorprendo dedicando cada día. Y ya por último, a Fede, que se despierta conmigo cada día y me regala un beso y una sonrisa, la que tantas veces a mí me falta, que hace del tiempo que paso con él un verdadero tesoro, que ilumina todos los días de mi vida desde que lo conozco, que saca la mejor versión de mí que solo él puede ver, que me hace ser siempre mejor persona, y al que nunca podré agradecerle todo por más que escriba. De nuevo, a todos vosotros, gracias, de corazón.

Me ha quedado un poco largo, pero me niego a decir menos de lo que he dicho.

Gracias.

Abstract

In the golden era of information, vast amounts of data are available to perform analysis on and extract valuable insight from. The area of science devoted to this problem is known as Knowledge Discovery in Databases; particularly, it is its branch of Data Science where this thesis is framed in. Specifically, this thesis focuses on clustering techniques that are capable of including relational information into the clustering process. This type of information does not fit into the classic supervised and unsupervised learning paradigms. However, the Semi-Supervised Learning (SSL) paradigm does provide us with the tools to perform clustering in the presence of relational information. This task is known as Constrained Clustering (CC).

Four objectives shape this thesis:

1. A comprehensive study in the area of CC, from an SSL standpoint. The goal of this objective is to produce the first comprehensive analysis of the CC state-of-the-art, including a standardization of the experimental procedures and a ranking of all CC methods proposed so far.

2. The development of metaheuristic-based proposals for the CC problem, from both single-objective and multi-objective optimization perspectives. Two metaheuristic-based methods are designed to achieve this objective. Both are first proposed in this thesis and have been specifically designed to obtain quality solutions for the CC problem. An empirical study compares both methods against the state-of-the art in their specific areas, demonstrating their superiority.

3. The proposal of hybrid models for the CC problem. Motivated by the high quality results that hybrid methods usually obtain in the field of CC, this thesis introduces a new hybrid model that combines the two broadest categories in the area: partitional CC and constrained distance metric learning. This proposal also includes a procedure to automatically determine the relevance of the pieces that make up the set of relational information. The empirical study carried out provides evidence of the proposal being superior to the state-of-the-art.

4. Finally, motivated by the existence of real-world problems where multiple types of background knowledge are available, this thesis tackles the issue of combining relational and monotonicity information. This combination gives rise to the monotonic

constrained clustering paradigm. The suitability of the problem is proved, and a baseline algorithm is proposed. The experimental study shows the superiority of monotonic constrained clustering algorithm over both purely constrained and purely monotonic clustering algorithms. This finding is backed by positive results in a classic set of benchmarks and a real-world monotonic constrained clustering problem.

The four objectives have been successfully addressed, and the thesis has made significant contributions to the field of CC from the point of view of SSL. The comprehensive study carried out in the first objective provides a solid basis for understanding the state-of-the-art in CC and enables the standardization of experimental procedures, which is crucial for a scientifically sound comparison of different methods. The development of metaheuristic-based proposals in the second objective provides new and efficient techniques to solve the CC problem, while the proposed hybrid models in the third objective demonstrate the potential of combining different approaches to further improve the quality of the results. Finally, the proposed monotonic constrained clustering paradigm in the fourth objective addresses the problem of combining multiple types of background knowledge and achieves superior results over purely constrained and purely monotonic clustering algorithms.

Resumen

La edad de oro de la información trae consigo la generación de enormes cantidades de datos, disponibles para ser analizados con el fin de extraer de ellos información valiosa. El área de la ciencia que se encarga de esta tarea se conoce como extracción de conocimiento en bases de datos (*Knowledge Discovery in Databases* - KDD). En concreto, esta tesis se centra en las técnicas de *clustering* que son capaces de considerar información relacional. Este tipo de información no encaja en los paradigmas supervisado y no supervisado considerados clásicos en el aprendizaje automático. Sin embargo, el paradigma de aprendizaje semisupervisado (*Semi-Supervised Learning* - SSL) nos proporciona las herramientas necesarias para aplicar técnicas de *clustering* en presencia de dicha información relacional. Esta tarea se conoce como agrupamiento restringido o *clustering* con restricciones (*Constrained Clustering* - CC).

Esta tesis aborda los siguientes cuatro objetivos:

1. El primero consiste en un estudio exhaustivo en el área del CC desde el punto de vista del SSL. Su finalidad es realizar el primer análisis exhaustivo del estado del arte en CC que incluya una estandarización de los procedimientos experimentales y una clasificación de todos los métodos de CC propuestos hasta ahora.

2. El segundo aborda el desarrollo de propuestas basadas en metaheurísticas para el CC, incluyendo técnicas de optimización tanto monoobjetivo como multiobjetivo. Para plantear este objetivo se han diseñado dos métodos. Ambos se proponen por primera vez en esta tesis y han sido diseñados específicamente para el CC. Un estudio empírico compara ambos métodos con el estado del arte en sus respectivas áreas y demuestra su superioridad.

3. El tercer objetivo tiene como finalidad investigar modelos híbridos para el CC. Motivada por la ya demostrada capacidad de dichos modelos para obtener resultados de calidad en el ámbito del CC, esta tesis incorpora un nuevo modelo híbrido que combina las dos categorías más amplias en el área: el CC particional y el aprendizaje métrico de distancias con restricciones. Esta propuesta también incluye un procedimiento para determinar automáticamente la relevancia de los elementos que conforman el conjunto de información relacional. El estudio empírico realizado proporciona evidencia de que nuestra propuesta es superior al estado del arte.

4. Por último, motivada por la existencia de problemas para los que están disponibles múltiples tipos de información incompleta (como la información relacional), esta tesis plantea cómo combinar información relacional y de monotonicidad. Dicha combinación da lugar al paradigma del *clustering* monotónico con restricciones. Tras demostrar la relevancia del problema que se aborda, se propone un algoritmo básico para resolver el mismo. El estudio experimental muestra la superioridad de este algoritmo sobre otros que solo son capaces de considerar información relacional o de monotonicidad por separado. Los hallazgos relacionados con este objetivo están respaldados por resultados positivos en baterías de pruebas estándar y en un caso de aplicación específico.

La tesis aborda los cuatro objetivos descritos con éxito. De esta manera, quedan suficientemente demostradas sus aportaciones en su campo de estudio. La revisión de la literatura llevada a cabo en el primer objetivo proporciona una base sólida para comprender el estado del arte en el CC. Permite la estandarización de los procedimientos experimentales, lo que es crucial para una posterior comparación científicamente fundamentada de diferentes métodos. El desarrollo de propuestas basadas en metaheurísticas en el segundo objetivo proporciona nuevas técnicas eficientes para resolver el CC, mientras que los modelos híbridos propuestos en el tercer objetivo demuestran el potencial de combinar diferentes enfoques para mejorar aún más la calidad de los resultados. Finalmente, el paradigma del *clustering* monotónico restringido, propuesto en el cuarto objetivo, aborda la combinación de múltiples tipos de información incompleta, logrando resultados superiores a los obtenidos con modelos anteriores.

Table of Contents

List of Abbreviations

3SHACC	Three Stages Hybrid Agglomerative Constrained Clustering
AHC	Agglomerative Hierarchical Clustering
ARI	Adjusted Rand Index
C-L	Cannot-Link
CC	Constrained Clustering
DILS	Dual Iterative Local Search
DML	Distance Metric Learning
EM	Expectation-Minimization
HC	Hierarchical Clustering
ILS	Iterative Local Search
KDD	Knowledge Discovery in Databases
LSI	Learning from Side Information
LS	Local Search
M-L	Must-Link
MCC	Monotonic Constrained Clustering
MCDA	Multi Criteria Decision Aid
ME-MOEA/D	Memetic Elitist - MOEA/D
ML	Machine Learning
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition
MOEA	Multi-Objective Evolutionary Algorithm

MOP	Multiobjective Optimization Problem
NMI	Non-Monotonic Index
PCKM-Mono	Pairwise Constrained K-Means - Monotonic
PF	Pareto Front
PS	Pareto Set
RI	Rand Index
SRWU	Shanghai Ranking of World Universities
SSL	Semi-Supervised Learning

Chapter I

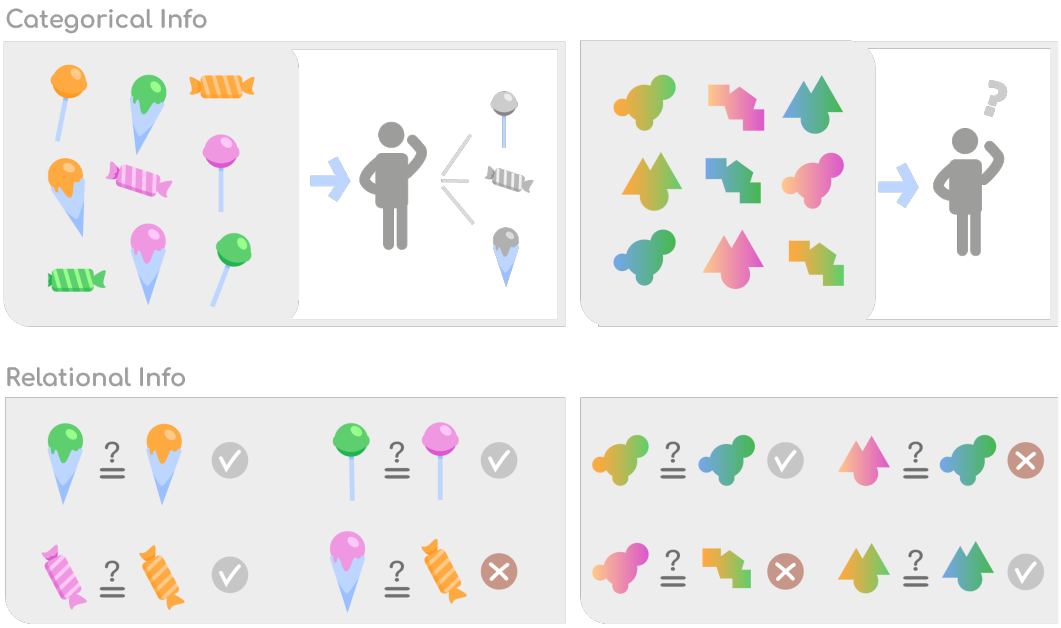
PhD Dissertation

«*The most exciting phrase to hear in science, the one that heralds the most discoveries, is not “Eureka!” (I found it!) but “That’s funny...”*».

– Isaac Asimov.

1 Introduction

We kindly ask the reader to consider the following situation. You are given two, quite bizarre, objects. You have time to examine them and learn everything you can about them, without any help. Then, someone comes to you and asks you to place them into a category. If you are free to give any answer, and you are lucky enough to have recognized the objects, you will give a precise category for them. However, if you did not recognized them, and you were not able to learn much about them, you will probably just make up a category, or say the category of something you know is similar to them. In short, we can agree that the number of possible answers to the question "What are the categories of the two objects?" is virtually infinite, only bounded by your knowledge or imagination. Now, let us consider another scenario. In the same setup, with the same two objects, you are asked whether they belong to the same category or not. Now you have only three possible answers: "Yes", "No", or "I don't know". This is really good news! We have bounded the possible answers from virtually infinite to just three by changing the question and giving up categorical information in favor of relational information. Moreover, the level of knowledge required to categorize the two objects is much higher than the knowledge required to tell if they belong to similar or to distinct categories. This makes relational knowledge easier to access. The image below tries to illustrate the two scenarios of our example.



This is the essence of the research carried out during this PhD, and what all the studies presented in this thesis focus on. This dissertation and the subsequent studies included in Chapter II formalize the concepts presented in the above situations: the objects, the questions, the categories, the answers, the relational information, the answering entity, etc. All of these concepts need to be formalized for this problem to be understood by a computer—for it to be computable. Our goal is to study how the "Yes", "No", or "I don't know" type of

information (relational information) can be used to infer the final categories of the objects, how a machine can learn from them. In order to do so, we frame this PhD within the data science research area.

Recent technological advancements have led to the generation and storage of massive amounts of data by various organizations and entities, including governments, private companies, and research institutes. These organizations have become increasingly interested in extracting useful insights from the data, which can give them a competitive advantage and drive innovation. As a result, data science has emerged as a leading field for research, development and innovation.

However, the standards in data science have become more rigorous, and the range of applications with varying restrictions has increased. Therefore, the correct implementation of the Knowledge Discovery in Databases (KDD) process has become crucial [PF91]. This process involves a set of stages that enable the identification of valuable patterns and relationships within the data. By following the KDD process, organizations can extract meaningful insights from their data, which can lead to new breakthroughs and a significant competitive advantage [PF91, HKP12]. The stages of KDD can be described as follows:

- **Problem specification**: the requirements and objectives of the discovery process are identified. This helps to establish a clear understanding of what the data mining process aims to achieve.
- **Data extraction**: involves selecting relevant data from various sources with the help of expert knowledge. The extracted data are then consolidated into a single dataset to be processed in subsequent stages.
- **Data preprocessing**: aims to transform the data into a format that can be handled by data mining techniques [GLH15]. It involves cleaning the data of any impurities, such as noise, missing or redundant information, and irrelevant data. The ultimate goal of data preprocessing is to obtain quality data, also known as Smart Data, for use in subsequent stages [GGLGH19].
- **Data mining**: involves extracting patterns, relationships, and/or models from the processed dataset [WFH+05]. The type of knowledge to be extracted determines the category of the data mining problem and the group of feasible techniques. Selecting the best technique for each task is a complex engineering process that requires optimization and validation of the different techniques available.
- **Interpretation and evaluation**: the extracted knowledge is analyzed and described to be easily understood and useful. This helps to ensure that the insights obtained from the data are actionable and provide meaningful value to the organization.

The process of data mining is a critical aspect of the KDD process, as it involves extracting valuable patterns, relationships, or trends hidden within the data. To this end, data mining algorithms must make use of as much information as possible, trying not to dismiss

anything available for the task [HKP12, WFH+05]. Data mining is shaped by two major approaches, which divide it into two distinct areas based on the type of knowledge used to perform learning [BN06]:

- **Supervised learning**: in supervised learning, the goal is to build a classifier or regressor that, trained with a set of examples (or instances) X and their corresponding output values (or labels) Y, can predict the value of unseen inputs. Classification [DH+06] and regression [DS98] are examples of classic supervised learning tasks.
- **Unsupervised learning**: in unsupervised learning, only the set of examples X is available, and no output value is provided. Here, the goal is to discover some underlying structure in the data. For example, in unsupervised clustering the goal is to infer a mapping from the input to clusters (groups) of similar instances [Mir12]. Association rules learning [CSP+07] is another example of classic unsupervised learning.

However, these two learning paradigms are very limited with respect to the type of information they can handle: either all instances are labeled (supervised), or none of them are (unsupervised). This is very restrictive when it comes to, for example, a subset of labeled data, or a different kind of information, such as relational information. The Semi-Supervised Learning (SSL) area arises to overcome these drawbacks. SSL is the branch of Machine Learning (ML) that tries to combine the benefits of these two approaches [CSZ10]. To do so, it makes use of both labeled and unlabeled data, or other kinds of expert knowledge. In classification or regression, for example, unlabeled data may also be available in addition to the (expected) set of labeled data. Similarly, when considering clustering problems, a smaller subset of labeled data (or other types of knowledge about the dataset) may be available. Generally, supplementary data that fits neither in the supervised nor unsupervised learning paradigm might also be at the disposal of the researchers. Failing to take advantage of this information does not optimally use the available sources of knowledge about the matter, and thus a need for SSL [VEH20] emerges.

With regards to the applicability of SSL, a natural question arises [CSZ10]: in comparison with supervised and unsupervised learning, can SSL obtain better results? A positive answer to this question can be readily inferred, as otherwise neither this thesis nor most of the studies cited in it would exist. However, there is an important condition imposed for the answer to be affirmative: the distribution of instances in X must be representative of the true distribution of the data. Formally, the underlying marginal distribution $p(X)$ over the input space must contain information about the posterior distribution $p(Y|X)$. Then, SSL is capable of making use of unlabeled data to obtain information about $p(X)$ and, therefore, about $p(Y|X)$ [VEH20]. Luckily, this condition appears to be fulfilled in most real-world learning problems, as suggested by the wide variety of fields where SSL is successfully applied. Nonetheless, the way in which $p(X)$ and $p(Y|X)$ are related is not always the same. This leads to the SSL assumptions, introduced in [CSZ10] and formalized in [VEH20]. A brief summary of these assumptions following [VEH20] is presented; please refer to the studies referenced for more details.

- **Smoothness assumption**: two instances that are close in the input space should have the same label.
- **Low-density assumption**: decision boundaries should preferably pass through low-density spatial regions.
- **Manifold assumption**: in tasks where data can be represented in Euclidean space, instances in a high-dimensional input space are usually gathered along lower-dimensional structures known as manifolds: locally Euclidean topological spaces.
- **Cluster assumption**: data points which belong to the same cluster also belong to the same class. This assumption can be seen as a generalization of the previous three specific assumptions.

As in other ML paradigms, the transduction versus induction dichotomy can be found in SSL. Usually, semi-supervised classification methods comprise the vast majority of the SSL field; therefore, the aforementioned dichotomy is explained in terms of classification as follows:

- **Inductive methods**: inductive methods aim to build a classifier capable of outputting a label for any instance in the input space. Unlabeled data can be used to train the classifier, but the predictions for unseen instances are independent of each other once the training phase is completed. An example of inductive method in supervised learning is linear regression [VEH20].
- **Transductive methods**: transductive methods do not build a classifier for the entire input space: their predictions are limited to the data used during the training phase. Transductive methods do no have separated training and testing phases. An example of transductive method in unsupervised learning is Hierarchical Clustering (HC) [VEH20].

Figure 2 helps us contextualize semi-supervised learning and its derivatives within the overall ML landscape. General SSL literature [Zhu05, CSZ10, ZG09] usually divides SSL methods into two categories: semi-supervised classification and semi-supervised clustering. Further dichotomies have been made in later literature. In [VEH20, Zho21] semi-supervised classification methods are taxonomized by taking into account the inductive versus transductive dichotomy. Some of the categories found in these taxonomies have been further studied: [ST14] proposes a taxonomy for graph-based semi-supervised methods, and [TGH15] does the same for the self-labeling field. Concerning semi-supervised clustering, [Bai13] proposes a high level taxonomy with 4 types of methods, while [DB07, BDW08] focus on the specific area of Constrained Clustering (CC). The supervised and unsupervised learning paradigms are included in Figure 2 for the sake of contextualization only. Consequently, only classic and widely-known tasks belonging to these areas have been included in the diagram.

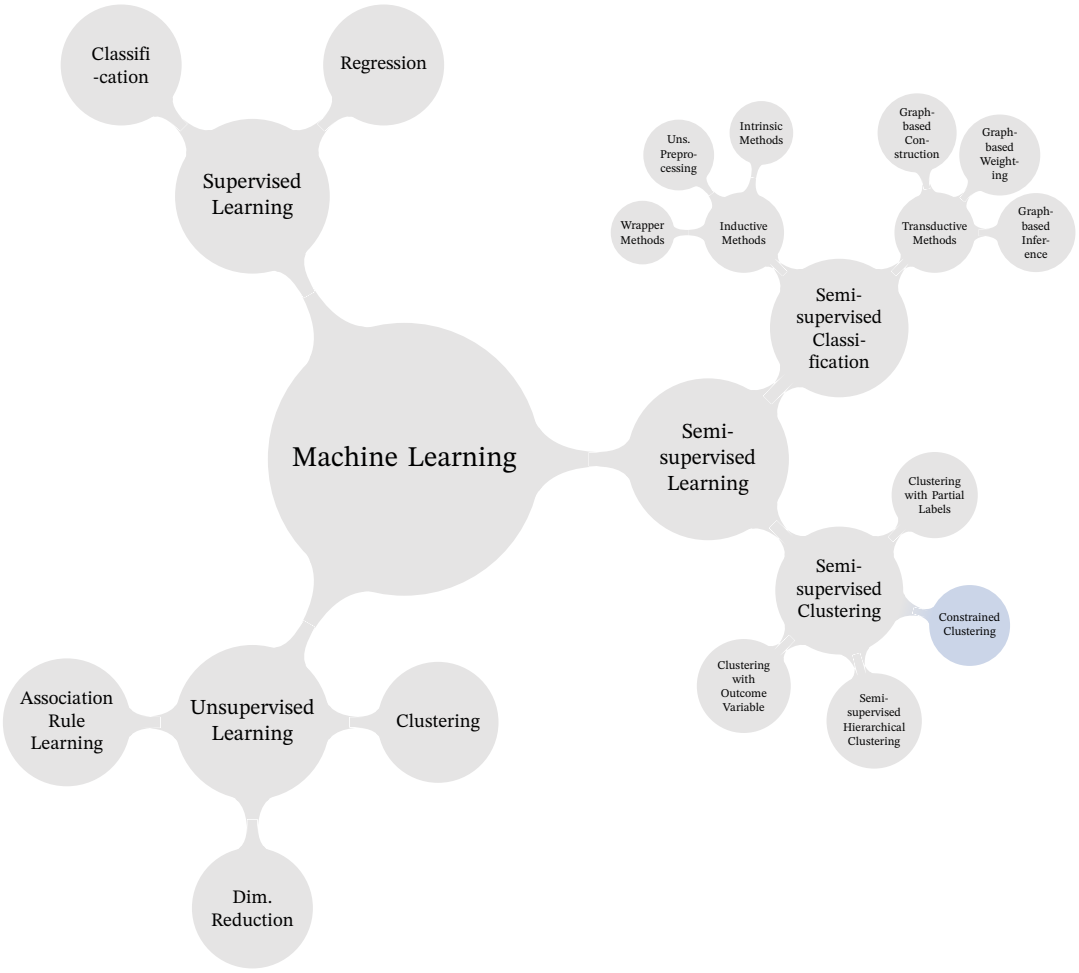


Figure 1: Mindmap of the overall ML landscape.

The area of semi-supervised clustering has been widely studied and successfully applied in many fields since its inception. It can be seen as a generalization of classic clustering which is able to include background knowledge in the clustering process [CSZ10]. Many types of background knowledge have been considered in semi-supervised clustering [Bai13], although the most studied one is the pairwise instance-level must-link and cannot-link constraints [BDW08]. It relates instances indicating if they belong to the same class (must-link) or to different classes (cannot-link), just like in the example at the beginning of this thesis. In the literature, the problem of performing clustering in the presence of this type of information is referred to as CC (marked in Figure 2 in blue). As Section 2 will show, clustering under must-link and cannot-link constraints is **NP**-complete [DR05b]. Consequently, it has to be tackled with approximate methods. The first objective of this thesis is to carry out an exhaustive study on these methods, with the aim of creating a taxonomy to categorize and organize them. This will further the knowledge of the area and, subsequently, foster innova-

tion. To the best of our knowledge, this will be the first attempt to produce such taxonomic study.

Metaheuristics are a class of optimization algorithms designed to tackle complex, non-linear optimization problems that are unfeasible to solve in practice using exact methods. Unlike traditional optimization techniques, such as linear programming or gradient-based methods, metaheuristics do not rely on explicit problem-specific knowledge, but instead on heuristics and stochastic search to find high-quality solutions. Within metaheuristics, evolutionary algorithms are a family of optimization algorithms that are inspired by natural selection. They simulate the process of evolution through genetic operators such as mutation, crossover, and selection to evolve a population of candidate solutions. The idea is to create a population of potential solutions and allow them to evolve and adapt through generations, gradually improving their fitness over time to arrive at high-quality solutions. Memetic evolutionary algorithms are a type of evolutionary algorithm that include exploitation procedures to accelerate the convergence process [GP10].

Metaheuristic evolutionary algorithms are highly flexible and can thus be applied to a wide range of optimization tasks, such as: crude oil time series [KABA20], COVID-19 disease recognition through X-ray images [AK20], digital currency forecasting [AKB19] and control of unmanned aerial vehicles [Alt20], among others. Classic clustering is no exception to this trend, with many studies presenting excellent results [NP14, HLZC19, JGGF16], although very little work has been done on CC. The second objective of this thesis is to approach CC from a metaheuristic point of view, aiming to experiment with multiple existing optimization models to eventually design a specific one for CC.

Within the field of metaheuristics, Multi-Objective Evolutionary Algorithms (MOEAs) [CLVV$^+$14] are particularly interesting to approach clustering. Many measures can be used to guide the clustering process towards a quality solution [SSZL05], although it is often not straightforward to integrate constraints into a single function that can be optimized by a standard optimizer. This issue is also found in the CC framework, as even more quality measures need to be used in order to include constraints. Constraint-related quality measures often contradict classic clustering quality measures, which complicates their integration into a single-objective function optimizable by a single-objective evolutionary algorithm. Multi-objective optimization schemes provide us with a powerful tool to overcome all these drawbacks. In its second objective, this thesis also aims to address CC with MOEAs. The objective is to design a new optimization model specifically for CC, including memetic procedures if deemed necessary.

Metaheuristics are not the only family of approximate methods that represent a promising approach to CC. Within the classic clustering paradigm, two broad categories can be found in the literature: partitional clustering and HC. In partitional clustering, a partition assigning every instance from the dataset to a specific cluster from among a fixed number of them is built, whereas HC obtains a tree-like hierarchical structure coding a set of partitions that allows the user to choose any cluster granularity between one and the number of instances in the dataset (more details in Section 2.1). Both of them have been applied to many real-world problems [ESA$^+$20], although when it comes to CC a significant imbalance

favoring partitional methods can be observed; very little work has been done to integrate constraints into HC methods [DR05a, BB06, KKM02, ZL11] with respect to the number of existing partitional CC methods. The third objective of this thesis is to dive into the use of hybrid agglomerative HC methods for CC, which should combine partitional CC methods and constrained Distance Metric Learning (DML) methods to address CC (see Section 2.6 for more details). In addition, the automatic generation of constraint weights is explored. Weighted constraints can be leveraged to guide the clustering process towards high quality solutions more effectively than unweighted constraints, although very little work has been done on automatic generation of constraint weights and their integration into distance-based CC methods.

Recently, a new type of background knowledge coming from the supervised learning paradigm has been integrated into unsupervised learning. Monotonic classification is a particular case of supervised learning where classes are a set of ordered categories and classification models must respect monotonicity constraints among instances based on their descriptive features. This means that, if an instance x_i has greater feature values than those of instance x_j, its assigned class must also be higher (greater) in the ordering than that of x_j [RDSDD21]. Considering the classic example of house pricing: for two houses in the same neighborhood, the bigger ones are constrained to have higher prices than smaller houses when the rest of the features of the houses are similar [GGL$^+$21]. This defines an order relationship between houses (instances) based on the value of their features, and therefore models that predict house prices must take this into account to produce accurate results. Monotonicity constraints are a type of background knowledge that can be leveraged to produce more accurate predictive models [CGK$^+$19], and has been successfully applied in real-world problems such as fraudulent firm classification [Pan20], real-time dynamic malware detection [CLSR18], or analysis of learning activities based on student opinion surveys [CAA$^+$17]. Additionally, a recent study from the Alan Turing Institute states that considering underlying data monotonicity in ML models leads to fairer applications [Les19]. In [RDSDD21] a methodology to perform clustering in the presence of monotonicity information (ordered clustering) is proposed within the Multi Criteria Decision Aid (MCDA) framework. The fourth and last objective of this thesis is to approach the combination of monotonicity information and pairwise instance-level constraints. This results in a novel clustering paradigm, which needs to be formalized and tackled with new methods. This new paradigm can be employed in real-world problems where both types of information are available, such as the Shanghai Ranking of World Universities (SRWU) dataset, to yield better results than purely CC and purely monotonic clustering methods.

Finally, to conclude this part we summarize the structure of this thesis, which is composed of two parts: the PhD dissertation, in Chapter I, and the publications that back the knowledge and conclusions presented in it, in Chapter II. The dissertation is split into 8 sections. Section 2 introduces the technical background of the concepts and terminology that will appear in subsequent sections. The justification, objectives and methodology that establish the foundation for this thesis are introduced in Sections 3, 4 and 5, respectively. Afterwards, Section 6 contains a summary of the research carried out in this thesis. Finally, in

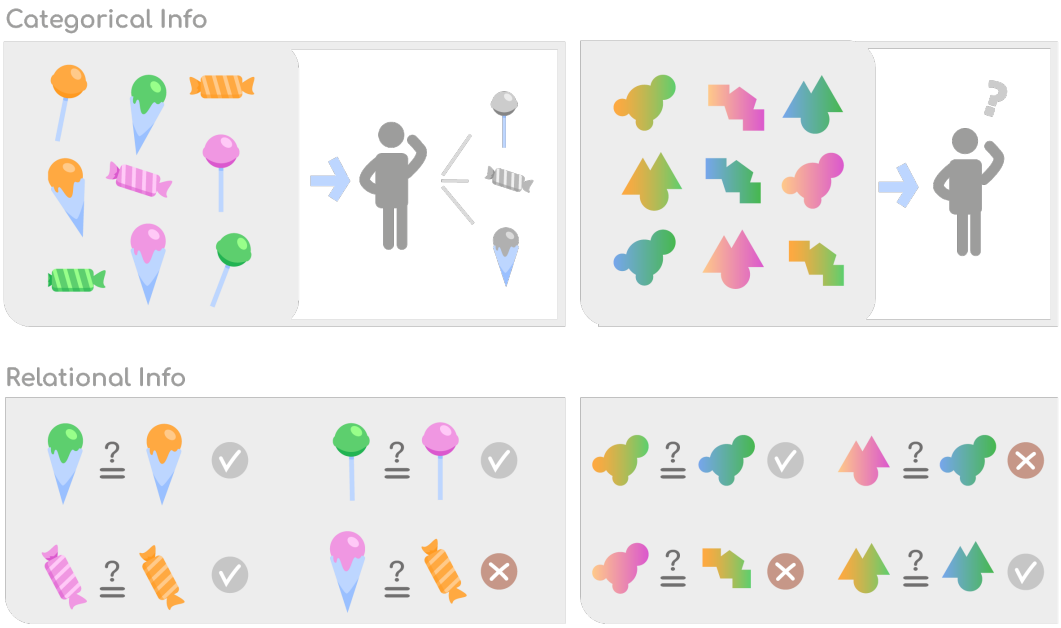
Section 8 the conclusions derived from the research are presented along with future research lines.

The second part (Chapter II) gathers the publications that support the knowledge and the conclusions discussed in the dissertation. From the 5 publications presented in this part, three of them are published in international and indexed journals, and two of them are currently under review. The publications are the following:

- Semi-Supervised Constrained Clustering: An In-Depth Overview, Ranked Taxonomy and Future Research Directions.
- DILS: Constrained Clustering Through Dual Iterative Local Search.
- ME-MOEA/D$_{CC}$: Multiobjective Constrained Clustering Through Decomposition - based Memetic Elitism.
- 3SHACC: Three Stages Hybrid Agglomerative Constrained Clustering.
- Semi-supervised Clustering with Two Types of Background Knowledge: Fusing Pairwise Constraints and Monotonicity Constraints.

Introducción

Rogamos al lector que considere la siguiente situación. Se le entregan dos objetos que desconoce. Tiene tiempo para examinarlos y aprender todo lo que pueda sobre ellos, sin ninguna ayuda. Al cabo de un rato, alguien se le acerca y le pide que los clasifique en una categoría. Si es libre de dar cualquier respuesta y tiene la suerte de haber reconocido los objetos, les asignará una categoría precisa. Sin embargo, si no los ha reconocido y no ha podido analizarlos, probablemente se inventará una categoría o los asociará a la de algo que le resulte similar. En resumen, podemos estar de acuerdo en que el número de respuestas posibles a la pregunta «¿cuáles son las categorías de los dos objetos?» es prácticamente infinito, solo limitado por sus conocimientos o su imaginación. Consideremos ahora otro escenario. En la misma situación, con los mismos dos objetos, se le pregunta si pertenecen a la misma categoría o no. Ahora sólo tiene tres respuestas posibles: «sí», «no» o «no lo sé». ¡Genial! Hemos acotado las posibles respuestas de prácticamente infinitas a sólo tres cambiando la pregunta y eliminando información categórica en favor de información relacional. Además, el nivel de conocimiento necesario para categorizar los dos objetos es mucho mayor que el necesario para saber si pertenecen a categorías similares o no. Esto facilita el acceso al conocimiento relacional. La siguiente imagen trata de ilustrar los dos escenarios descritos anteriormente.



Esta es la esencia de la investigación llevada a cabo durante este doctorado y en lo que se centran todos los estudios presentados en esta tesis. Esta disertación y los estudios posteriores incluidos en el Capítulo II formalizan los conceptos que se presentan en las situaciones anteriores: los objetos, las preguntas, las categorías, las respuestas, la información relacional, la entidad que responde, etc. Es necesario formalizar todos estos conceptos para que

este problema pueda ser comprendido por un ordenador, para que sea computable. Nuestro objetivo es estudiar cómo la información de tipo «sí», «no» o «no lo sé» (información relacional) puede utilizarse para inferir las categorías finales de los objetos, cómo una máquina puede aprender de ellas. Para ello, planteamos este doctorado en el área de investigación de la ciencia de datos.

Los recientes avances tecnológicos han dado lugar a la generación y almacenamiento de cantidades masivas de datos por parte de diversas organizaciones y entidades, entre ellas gobiernos, empresas privadas e institutos de investigación. Estas organizaciones están cada vez más interesadas en extraer información útil de los datos, lo que puede proporcionarles una ventaja competitiva e impulsar la innovación. Como resultado, la ciencia de los datos ha surgido como un campo puntero para la investigación, el desarrollo y la innovación.

Sin embargo, los estándares en la ciencia de datos son cada vez más rigurosos y las aplicaciones tienen requisitos más específicos. Por ello, la correcta implementación del proceso de extracción de conocimiento en bases de datos (*Knowledge Discovery in Databases* - KDD) se ha vuelto crucial [PF91]. Este proceso implica un conjunto de etapas que permiten la identificación de patrones y relaciones. Siguiendo el proceso KDD, los organismos pueden extraer información valiosa de sus datos, lo que conlleva nuevos avances y una significativa ventaja competitiva [PF91, HKP12]. Las etapas del KDD pueden describirse así:

- **Especificación del problema**: se identifican los requisitos y objetivos del proceso KDD. Esto ayuda a establecer una comprensión clara de lo que el proceso de extracción de datos pretende lograr.
- **Extracción de datos**: consiste en seleccionar los datos pertinentes de diversas fuentes con la ayuda de conocimientos especializados. A continuación, los datos extraídos se consolidan en un único conjunto de datos que se procesará en etapas posteriores.
- **Preprocesamiento de datos**: tiene como objetivo transformar los datos en un formato que pueda ser utilizado por las técnicas de minería de datos [GLH15]. Implica limpiar los datos de cualquier impureza, como ruido, información incompleta o redundante y datos irrelevantes. El objetivo final del preprocesamiento de datos es obtener datos de calidad, también conocidos como *Smart Data*, para su uso en etapas posteriores [GGLGH19].
- **Minería de datos**: consiste en extraer patrones, relaciones o modelos del conjunto de datos procesados [WFH$^+$05]. El tipo de conocimiento que debe extraerse determina la categoría del problema de minería de datos y el grupo de técnicas viables. La selección de la mejor técnica para cada problema es un complejo proceso de ingeniería que requiere la optimización y validación de las técnicas disponibles.
- **Interpretación y evaluación**: los conocimientos extraídos se analizan y describen para que sean fácilmente comprensibles y útiles. Esto ayuda a garantizar que los conocimientos obtenidos a partir de los datos sean procesables y aporten un valor significativo a la organización que pretenda utilizarlos.

La minería de datos es un aspecto crítico del KDD, ya que implica la extracción de patrones, relaciones o tendencias ocultas en los datos. Para ello, los algoritmos de minería de datos deben hacer uso de la mayor cantidad de información posible, tratando de no descartar nada de lo disponible para la tarea [HKP12, WFH+05]. La minería de datos está conformada por dos grandes enfoques, que la dividen en dos áreas diferenciadas en función del tipo de conocimiento utilizado para realizar el aprendizaje [BN06]:

- **Aprendizaje supervisado**: en el aprendizaje supervisado, el objetivo es construir un clasificador o regresor que, entrenado con un conjunto de ejemplos (o instancias) X y sus correspondientes valores de salida (o etiquetas) Y, pueda predecir el valor de entradas no vistas. La clasificación [DH+06] y la regresión [DS98] son ejemplos de tareas clásicas de aprendizaje supervisado.
- **Aprendizaje no supervisado**: en el aprendizaje no supervisado, solo se dispone del conjunto de ejemplos X y no se proporciona ningún valor de salida. En este caso, el objetivo es descubrir alguna estructura subyacente en los datos. Por ejemplo, en el clustering no supervisado, el objetivo es inferir un mapeo de la entrada a *clusters* (grupos) de instancias similares [Mir12]. El aprendizaje de reglas de asociación [CSP+07] es otro ejemplo de aprendizaje clásico no supervisado.

Sin embargo, estos dos paradigmas de aprendizaje están muy limitados en cuanto al tipo de información que pueden utilizar: o todas las instancias están etiquetadas (supervisado), o ninguna lo está (no supervisado). Esto es muy restrictivo cuando se trata, por ejemplo, de un subconjunto de datos etiquetados o de otro tipo de información, como la información relacional. El aprendizaje semisupervisado (*Semi-Supervised Learning* - SSL) surge para abordar estos inconvenientes. El SSL es la rama del aprendizaje automático (Machine Learning - ML) que intenta combinar las ventajas de los dos enfoques anteriores [CSZ10]. Para ello, hace uso de datos etiquetados y no etiquetados, o de otros tipos de conocimiento experto. En clasificación o regresión, por ejemplo, los datos no etiquetados también pueden estar disponibles además del conjunto (esperado) de datos etiquetados. Del mismo modo, al considerar problemas de *clustering*, puede estar disponible un subconjunto más pequeño de datos etiquetados (u otros tipos de conocimiento sobre el conjunto de datos). Por lo general, los investigadores también pueden disponer de datos complementarios que no encajan ni en el paradigma del aprendizaje supervisado ni en el del aprendizaje no supervisado. Si no se aprovecha esta información, no se utilizan de forma óptima las fuentes de conocimiento disponibles, por lo que surge la necesidad del SSL [VEH20].

En cuanto a la aplicabilidad del SSL, surge una pregunta evidente [CSZ10]: en comparación con el aprendizaje supervisado y no supervisado, ¿puede el SSL obtener mejores resultados? Se puede deducir fácilmente una respuesta afirmativa, ya que de lo contrario no existirían ni esta tesis ni la mayoría de los estudios citados en ella. Sin embargo, se impone una condición importante para que la respuesta sea afirmativa: la distribución de instancias en X debe ser representativa de la auténtica distribución de los datos. Formalmente, la distribución marginal subyacente $p(X)$ sobre el espacio de entrada debe contener información

sobre la distribución posterior $p(Y|X)$. Así, el SSL es capaz de hacer uso de datos no etiquetados para obtener información sobre $p(X)$ y, por tanto, sobre $p(Y|X)$ [VEH20]. Por suerte, esta condición parece cumplirse en la mayoría de los problemas reales de aprendizaje, como sugiere la amplia variedad de campos en los que se aplica con éxito SSL. No obstante, la forma en que $p(X)$ y $p(Y|X)$ se relacionan no siempre es la misma. Esto nos lleva a los supuestos SSL, presentados en [CSZ10] y formalizados en [VEH20]. A continuación, se muestra un breve resumen de estos supuestos, basándose en [VEH20] (consulte las referencias para saber más).

- **Hipótesis de uniformidad**: dos instancias cercanas en el espacio de entrada deben tener la misma etiqueta.
- **Hipótesis de baja densidad**: las fronteras de decisión deben pasar preferentemente por regiones del espacio de baja densidad.
- **Hipótesis de variedad**: en los problemas en los que los datos pueden representarse en el espacio euclidiano, las instancias de un espacio de entrada de alta dimensionalidad suelen agruparse a lo largo de estructuras de menor dimensión conocidas como colectores: espacios topológicos localmente euclidianos.
- **Hipótesis de agrupamiento**: las instancias que pertenecen al mismo grupo también pertenecen a la misma clase. Este supuesto puede considerarse una generalización de los tres supuestos anteriores.

Al igual que en otros paradigmas de ML, la diferenciación entre transducción e inducción se da en el SSL. Por lo general, los métodos de clasificación semisupervisada comprenden la gran mayoría del campo del SSL; por lo tanto, la dicotomía antes mencionada se explica en términos de clasificación de la siguiente manera:

- **Métodos inductivos**: los métodos inductivos tratan de construir un clasificador capaz de producir una etiqueta para cualquier instancia del espacio de entrada. Se pueden utilizar datos no etiquetados para entrenar el clasificador, pero las predicciones para instancias no vistas son independientes unas de otras una vez completada la fase de entrenamiento. Un ejemplo de método inductivo en el aprendizaje supervisado es la regresión lineal [VEH20].
- **Métodos transductivos**: los métodos transductivos no construyen un clasificador para todo el espacio de entrada: sus predicciones se limitan a los datos utilizados durante la fase de entrenamiento. Los métodos transductivos no tienen fases de entrenamiento y prueba separadas. Un ejemplo de método transductivo en el aprendizaje no supervisado es el *clustering* jerárquico (*Hierarchical Clustering*- HC) [VEH20].

La Figura 2 nos ayuda a contextualizar el aprendizaje semisupervisado y sus derivados dentro del panorama general del ML. La literatura general sobre el SSL [Zhu05, CSZ10,

ZG09] suele dividir los métodos de SSL en dos categorías: clasificación semisupervisada y *clustering* semisupervisado. En literatura posterior se han hecho otras dicotomías. En [VEH20, Zho21] los métodos de clasificación semisupervisada se taxonomizan teniendo en cuenta las diferencias entre inducción y transducción. Algunas de las categorías propuestas en estas taxonomías se han estudiado más a fondo: [ST14] propone una taxonomía para los métodos semisupervisados basados en grafos y [TGH15] hace lo mismo para el campo del autoetiquetado. En cuanto al *clustering* semisupervisado, [Bai13] propone una taxonomía de alto nivel con cuatro tipos de métodos, mientras que [DB07, BDW08] se centra en el área específica del CC. Los paradigmas de aprendizaje supervisado y no supervisado se incluyen en la figura 2 únicamente a efectos de contextualización. En consecuencia, sólo se han incluido en el diagrama las tareas clásicas y ampliamente conocidas pertenecientes a estas áreas.

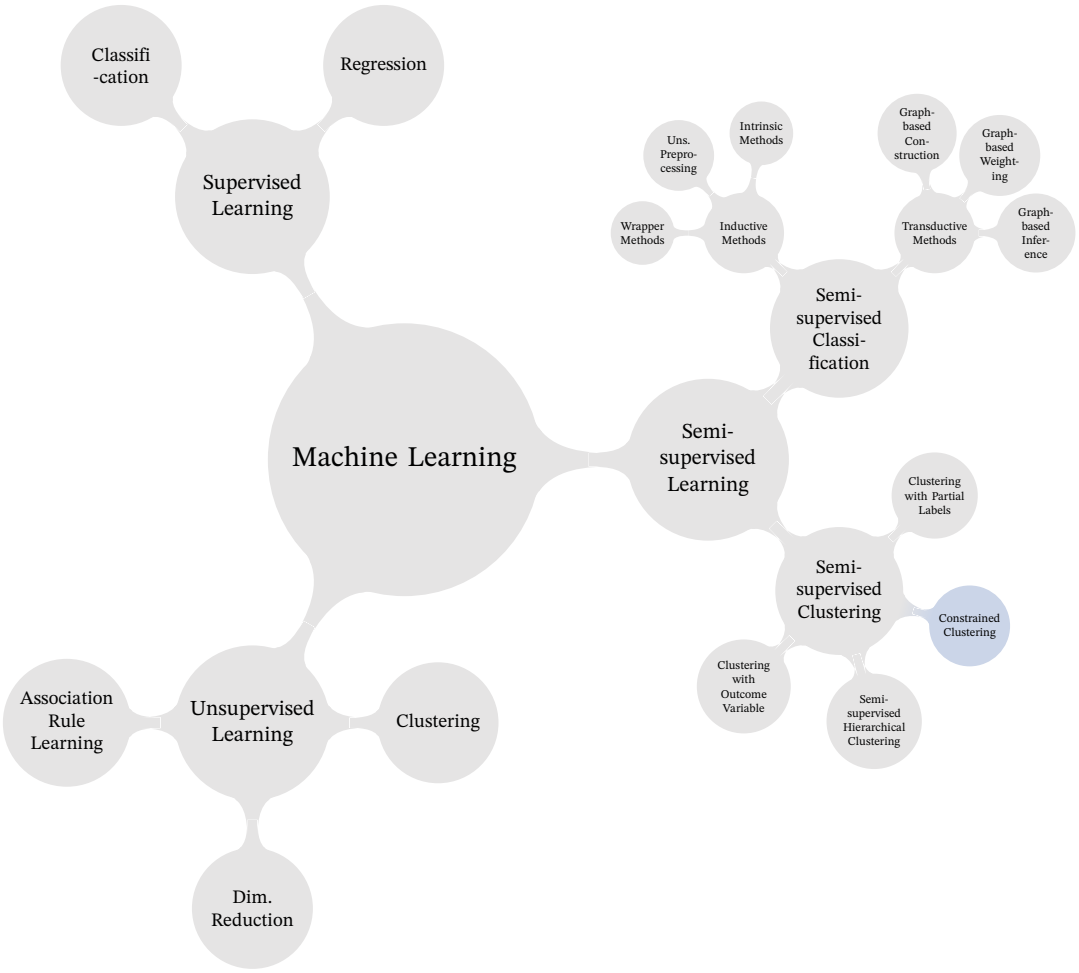


Figura 2: *Mindmap* del panorama general en el ML.

El área del *clustering* semisupervisado ha sido estudiada en profundidad y aplicada con éxito en múltiples campos desde sus inicios. Puede verse como una generalización del problema clásico de *clustering* que es capaz de incluir varios tipos de conocimiento en el proceso de *clustering* [CSZ10]. Se han considerado muchos tipos de conocimiento en el *clustering* semisupervisado [Bai13], aunque el más estudiado son las restricciones a nivel de instancia *Must-Link (M-L)* y *Cannot-Link (C-L)* [BDW08]. Este relaciona instancias indicando si pertenecen a la misma clase (M-L) o a clases diferentes (C-L), tal y como se planteaba en el ejemplo utilizado para presentar esta tesis. En la literatura, el problema de aplicar *clustering* en presencia de este tipo de información (información parcial) se denomina agrupamiento restringido o *clustering* con restricciones (*Constrained Clustering* - CC) (señalado en la Figura 2 en azul). Como se mostrará en la Sección 2, el CC es un problema **NP**-completo [DR05b]. En consecuencia, debe abordarse con métodos aproximados. El primer objetivo de esta tesis es realizar un estudio exhaustivo sobre estos métodos, con el fin de crear una taxonomía que los categorice y organice. De este modo, se ampliará el conocimiento en el área y, en consecuencia, se fomentará la innovación. Hasta donde sabemos, esto supondría el primer intento de elaborar un estudio taxonómico de este tipo.

Las metaheurísticas son una clase de algoritmos de optimización diseñados para abordar problemas de optimización complejos no lineales que no se pueden resolver de forma práctica utilizando métodos exactos. A diferencia de las técnicas de optimización tradicionales, como la programación lineal o los métodos basados en gradientes, las metaheurísticas no se basan en conocimientos explícitos específicos del problema, sino en la heurística y la búsqueda estocástica para encontrar soluciones de calidad. Dentro de las metaheurísticas, los algoritmos evolutivos son una familia de algoritmos de optimización que se inspiran en la selección natural. Simulan el proceso de evolución mediante operadores genéticos como la mutación, el cruce y la selección para hacer evolucionar una población de soluciones candidatas a un problema. La idea es crear una población de soluciones potenciales y dejar que evolucionen y se adapten a través de generaciones, mejorando gradualmente su aptitud con el tiempo para llegar a soluciones de alta calidad. Los algoritmos miméticos son un tipo de algoritmo evolutivo que incluyen procedimientos de explotación del espacio de soluciones para acelerar el proceso de convergencia [GP10].

Los algoritmos evolutivos son altamente flexibles, por lo que pueden aplicarse a un amplio abanico de problemas de optimización, tales como: tratamiento de crudo mediante series temporales [KABA20], reconocimiento de COVID-19 a través de imágenes de rayos X [AK20], predicciones en banca digital [AKB19] y control de vehículos aéreos no tripulados [Alt20], entre otros. El *clustering* clásico no es una excepción a esta tendencia, puesto que existen muchos estudios que presentan excelentes resultados [NP14, HLZC19, JGGF16], aunque se ha trabajado muy poco en CC. El segundo objetivo de esta tesis es abordar el problema del CC desde un punto de vista metaheurístico, a través de la experimentación con modelos de optimización existentes para finalmente diseñar uno específico para elCC.

Dentro del campo de las metaheurísticas, los algoritmos evolutivos multiobjetivo (*Multi-Objective Evolutionary Algorithms* - MOEAs) [CLVV$^+$14] son particularmente interesantes para abordar el *clustering*. Se pueden utilizar muchas medidas para guiar el proceso de *clus-*

tering hacia una solución de calidad [SSZL05], aunque a menudo no es sencillo integrar las restricciones en una única función que pueda ser optimizada mediante métodos estándar. Este problema también se plantea en el marco del CC, ya que para incluir las restricciones es necesario utilizar aún más medidas de adecuación. Las relacionadas con las restricciones suelen contradecir las relacionadas con el *clustering* clásico, lo que complica su integración en una única función objetivo que sea optimizable por un algoritmo de optimización monoobjetivo. Los esquemas de optimización multiobjetivo nos proporcionan una poderosa herramienta para superar este inconveniente. En su segundo apartado, esta tesis pretende abordar el problema del CC mediante MOEAs. El objetivo es diseñar un nuevo modelo de optimización específico para el CC, incluyendo procedimientos meméticos si fuera necesario.

Las metaheurísticas no son la única familia de métodos aproximados que representan una aproximación prometedora al problema del CC. Dentro del paradigma clásico de *clustering*, la literatura diferencia dos grandes familias: clustering particional y HC. En el *clustering* particional, se construye una partición que asigna cada instancia del conjunto de datos a un *cluster* específico de entre un número fijo de ellos, mientras que el HC obtiene una estructura jerárquica en forma de árbol que permite al usuario elegir cualquier nivel para formar distintas particiones (más información en la Sección 2.1). Ambos se han aplicado a multitud de problemas reales [ESA+20], aunque cuando se trata del CC se observa un desequilibrio significativo a favor de los métodos particionales. Se ha trabajado muy poco para integrar restricciones en los métodos de HC [DR05a, BB06, KKM02, ZL11] respecto al número de métodos de CC particionales existentes. El tercer objetivo de esta tesis es profundizar en el uso de métodos aglomerativos híbridos de HC para el CC, que deben combinar métodos de CC particionales y métodos de aprendizaje métrico de distancias (Distance Metric Learning - DML) con restricciones (ver Sección 2.6 para más información). Además, se explora la generación automática de pesos de las restricciones. Las restricciones ponderadas se pueden utilizar para guiar el proceso de *clustering* hacia soluciones de alta calidad de manera más eficaz que las restricciones no ponderadas, aunque se ha trabajado muy poco en la generación automática de dichos pesos y en su integración en métodos de CC basados en la distancia.

Un nuevo tipo de información parcial procedente del paradigma del aprendizaje supervisado se ha integrado hace poco en el aprendizaje no supervisado. La clasificación monotónica es un caso particular del aprendizaje supervisado en el que las clases son un conjunto de categorías ordenadas y los modelos de clasificación deben respetar restricciones de monotonicidad entre instancias. Esto supone que, si los valores de características de una instancia x_i son mayores mayores que los de la instancia x_j, su clase asignada también debe ser superior en el ordenamiento que la de x_j [RDSDD21]. Consideremos el ejemplo clásico del precio de las casas: para dos casas en el mismo barrio, las más grandes deben tener precios más altos que las más pequeñas cuando el resto de las características de las casas son similares [GGL+21]. Esto define una relación de orden entre las casas (instancias) basada en el valor de sus características y, por lo tanto, los modelos que predicen los precios de las casas deben tenerlo en cuenta para producir resultados precisos. Las restricciones de monotonicidad son un tipo de información parcial que puede aprovecharse para producir mo-

delos predictivos más precisos [CGK+19], y se ha aplicado con éxito en casuísticas como la clasificación de empresas fraudulentas [Pan20], la detección dinámica de *malware* en tiempo real [CLSR18]o el análisis de actividades de aprendizaje basado en encuestas [CAA+17]. Además, un estudio reciente del Instituto Alan Turing afirma que considerar la monotonicidad de los datos subyacentes en modelos de ML produce aplicaciones más justas [Les19]. En [RDSDD21] se propone una metodología para realizar *clustering* en presencia de información de monotonicidad (*clustering* ordenado) dentro del marco del análisis de decisión multicriterio (Multi Criteria Decission Aid - MCDA). El cuarto y último objetivo de esta tesis es abordar la combinación de información de monotonicidad y restricciones a nivel de instancia (M-L y C-L). Esto da lugar a un nuevo paradigma de *clustering* que necesita ser formalizado y abordado con nuevos métodos. Este nuevo paradigma se puede emplear en problemas en los que se dispone de ambos tipos de información, como el conjunto de datos del ranking mundial de la universidad de Shanghai (Shanghai Ranking World University - SRWU), para obtener mejores resultados que los métodos de *clustering* que consideran solo restricciones a nivel de instancia o solo restricciones de monotonicidad.

En último lugar, para concluir esta introducción presentamos un resumen de la estructura de esta tesis, compuesta de dos partes: la disertación doctoral, en el Capítulo I, y las publicaciones que avalan los conocimientos y conclusiones expuestos en la misma, en el Capítulo II. La disertación se divide en 8 secciones. La sección 2 profundiza en el trasfondo técnico de los conceptos y terminología utilizados en las secciones posteriores. La justificación, los objetivos y la metodología que sientan las bases de esta tesis se indican en las secciones 3, 4 y 5, respectivamente. Posteriormente, en la Sección 6 se presenta un resumen de la investigación llevada a cabo. Finalmente, en la Sección 8 se exponen las conclusiones derivadas de la investigación junto con futuras líneas de investigación.

La segunda parte (Capítulo II) recoge las publicaciones que avalan los conocimientos y las conclusiones discutidas en la disertación. De las cinco publicaciones presentadas, tres de ellas están publicadas en revistas indexadas internacionales y dos de ellas están actualmente en revisión. Las publicaciones son las siguientes:

- Semi-Supervised Constrained Clustering: An In-Depth Overview, Ranked Taxonomy and Future Research Directions.
- DILS: Constrained Clustering Through Dual Iterative Local Search.
- ME-MOEA/D$_{CC}$: Multiobjective Constrained Clustering Through Decomposition-based Memetic Elitism.
- 3SHACC: Three Stages Hybrid Agglomerative Constrained Clustering.
- Semi-supervised Clustering with Two Types of Background Knowledge: Fusing Pairwise Constraints and Monotonicity Constraints.

2 Preliminaries

This section introduces the technical knowledge necessary to understand the remainder of the dissertation (Chapter I). Firstly, the technical background in classic clustering is given in Section 2.1. Afterwards, CC is described formally in Sections 2.2 and 2.3. Section 2.4 is about the experimental setups used in all the studies included in this thesis. It introduces the concepts related to the common evaluation procedures. The rest of the sections are devoted to describing the key aspects of the new methods proposed in this thesis: Section 2.5 introduces multi-objective optimization, Section 2.6 defines and formalizes DML, and Section 2.7.1 explains monotonicity constraints and their potential role in MCDA.

2.1 Background on classic clustering

Partitional clustering can be defined as the task of grouping the instances of a dataset into K clusters. A dataset X consists of n instances, and each instance is described by u features. More formally, $X = \{x_1, \cdots, x_n\}$, with the ith instance noted as $x_i = (x_{[i,1]}, \cdots, x_{[i,u]})$. A typical clustering algorithm assigns a class label l_i to each instance $x_i \in X$. As a result, we obtain the list of labels $L = [l_1, \cdots, l_n]$, with $l_i \in \{1, \cdots, K\}$, that effectively splits X into K non-overlapping clusters c_i to form a partition called C. The list of labels producing partition C is referred to as L^C. The criterion used to assign an instance to a given cluster is the similarity to the rest of elements in that cluster, and the dissimilarity to the rest of instances of the dataset. This value can be obtained with some kind of distance measurement [JMF99].

HC methods produce an informative hierarchical structure of clusters called dendrogram. Partitions as described above, with a number of clusters ranging from 1 to n, can always be obtained from a dendrogram by just selecting a level from its hierarchy and partitioning the dataset according to its structure. Typically, agglomerative HC methods start with a large number of clusters and iteratively merge them according to some affinity criteria until a stopping condition is reached. Every merge produces a new level in the hierarchy of the dendrogram. Formally, given an initial partition with n_c clusters $C = \{c_1, \cdots, c_{n_c}\}$ (usually $n_c = n$), a traditional agglomerative CC method selects two clusters to merge by applying Equation 1.

$$\{c_i, c_j\} = \underset{c_i, c_j \in C, i \neq j}{\operatorname{argmax}} A(c_i, c_j), \tag{1}$$

with $A(\cdot, \cdot)$ being a function used to determine the affinity between the two clusters given as arguments. This function needs to be carefully chosen for every application, as it greatly affects the result of the clustering process. Some conventional methods to measure affinity between clusters are worth mentioning, such as single linkage, average linkage and complete linkage [JMF99]. Nevertheless, different measures are employed in out-of-lab applications, as the manifold structures usually found in real-world datasets can hardly be captured by the classic affinity measures mentioned above. Typically, classic partitional clustering methods are less algorithmically complex than HC methods, with the former featuring $\mathcal{O}(n)$ complexity and the latter $\mathcal{O}(n^2)$ [DB07].

2.2 Background on pairwise constraints

In most clustering applications, it is common to have some kind of information about the dataset that will be analyzed. In CC this information is given in the form of pairs of instances that must, or must not, be assigned to the same cluster. We can now formalize these two types of constraints:

- M-L constraints $C_{=}(x_i, x_j)$: instances x_i and x_j from X must be placed in the same cluster. The set of M-L constraints is referred to as $C_{=}$.

- C-L constraints $C_{\neq}(x_i, x_j)$: instances x_i and x_j from X cannot be assigned to the same cluster. The set of C-L constraints is referred to as $C_{\neq}$.

The goal of CC is to find a partition (or clustering) of K clusters $C = \{c_1, \cdots, c_K\}$ of the dataset X that ideally satisfies all constraints in the union of both constraint sets, called $CS = C_{=} \bigcup C_{\neq}$. As in classic clustering, the sum of instances in each cluster c_i is equal to the number of instances in X, which we have defined as $n = |X| = \sum_{i=1}^{K} |c_i|$.

Knowing how a constraint is defined, M-L constraints are an example of an equivalence relation; therefore, M-L constraints are reflexive, transitive and symmetric. This way, given constraints $C_{=}(x_a, x_b)$ and $C_{=}(x_b, x_c)$, then $C_{=}(x_a, x_c)$ is verified. In addition to this, if $x_a \in c_i$ and $x_b \in c_j$ are related by $C_{=}(x_a, x_b)$, then $C_{=}(x_c, x_d)$ is verified for any $x_c \in c_i$ and $x_d \in c_j$ [DB07].

It can also be proven that C-L constraints do not constitute an equivalence relation. However, analogously, given $x_a \in c_i$ and $x_b \in c_j$, and the constraint $C_{\neq}(x_a, x_b)$, then it is also true that $C_{\neq}(x_c, x_d)$ for any $x_c \in c_i$ and $x_d \in c_j$ [DB07].

Regarding the degree to which constraints need to be met in the output partition or dendrogram of any CC algorithm, a simple dichotomy can be made: hard pairwise constraints must necessarily be satisfied, while soft pairwise constraints can be violated to a variable extent. This distinction is introduced in [DB07] and adopted by later studies. The major advantages in favor of soft over hard constraints are the resiliency to noise in the constraint set, the flexibility in the design of cost/objective functions, and their optimization procedures. The ability to consider soft, hard, or both types of constraints is a defining element for CC methods.

In [DWB06] two measures designed to characterize the quality of a given constraint set are proposed: **informativeness** (or informativity [DB07]) is used to determine the amount of information in the constraint set that the CC algorithm could determine on its own, and **coherence**, which measures the amount of agreement between the constraints themselves. These two measures were proposed in early stages of the development of the CC area; however, they have not been used consistently in later studies.

2.3 The feasibility problem

Given that CC adds a new element to classic clustering, we must consider how it affects its complexity in both of its forms: partitional and hierarchical. Intuitively, the clustering problem goes from its classic formulation "find the best partition for a given dataset" to its constrained form "find the best partition for a given dataset satisfying all constraints in the constraint set". The formalization of these concepts is tackled in [DR05b, DB07, DR09], where the feasibility problems for partitional and hierarchical CC are defined as in 2.1 and 2.2 respectively, with $CS = C_{\neq} \cup C_{=}$ (the joint constraint set). Given these two definitions, we say that **a partition C for a dataset X is feasible when all constraints in CS are satisfied by** C. Note that there exist constraint sets for which a feasible partition can never be found. For example, no feasible partition exists for $CS_1 = \{C_{=}(x_1, x_2), C_{\neq}(x_1, x_2)\}$ regardless of the value of K. Similarly, the feasibility of partitions such as $CS_2 = \{C_{\neq}(x_1, x_2), C_{\neq}{\neq}(x_2, x_3)C_{\neq}, (x_1x_3)\}$ depends on the value of K. In this case, the feasibility problem for CS_2 can be solved for $K = 3$ but not for $K = 2$.

Definition 2.1. ***Feasibility Problem for Partitional CC*** *: given a dataset X, a constraint set CS, and the bounds on the number of clusters $k_l \leq K \leq k_u$, is there a partition C of X with K clusters that satisfies all constraints in CS? [DR05b]*

In [DR05b] it is proven that, when $k_l = 1$ and $k_u \geq 3$, the feasibility problem for partitional CC is **NP**-complete, by reducing it from the Graph K-Colorability problem. It is also proven that it is not harder, so both have the same complexity. Table 1 shows the complexity of the feasibility for different types of constraints.

Definition 2.2. ***Feasibility Problem for Hierarchical CC****: given a dataset X, the constraint sets CS, and the symmetric distance measure $D(x_i, x_j) \geq 0$ for each pair of instances, can X be partitioned into clusters so that all constraints in CS are satisfied? [DR09]*

Please note that the definition of the feasibility problem for partitional CC (in Definition 2.1) is significantly different from the definition of the feasibility problem for hierarchical CC (in 2.2). Particularly, the formulation of hierarchical CC imposes no restriction on the number of clusters K, which is equivalent to considering that a partition that satisfies all constraints can be produced at any level of the dendrogram [DR09]. In [DR05a] a reduction from the One-in-three 3SAT with positive literals (which is **NP**-complete) for the problem in Definition 2.2 is used to prove the complexities presented in Table 1 for hierarchical CC. It is worth mentioning that, for hierarchical CC, the dead-ends problem arises: a hierarchical CC algorithm may find scenarios where no merge/split can be carried out without violating a constraint. Previous solutions based on the transitive closure of the constraint sets have been proposed for this special case, although they involve not generating a full dendrogram [DR05b].

Overall, complexity results in Table 1 show that the feasibility problem under C-L constraints is intractable, and hence CC is intractable too. This leads to Observation 2.1. For more details on the complexity of CC please see [DR05b].

Constraints	Partitional CC	Hierarchical CC	Dead Ends?
M-L	**P**	**P**	No
C-L	**NP**-complete	**NP**-complete	Yes
M-L and C-L	**NP**-complete	**NP**-complete	Yes

Table 1: Feasibility problem complexities for partitional and hierarchical CC and dead-ends found in hierarchical CC [DR05b].

Observation 2.1. ***Knowing that a feasible solution exists does not help us find it****. The results from Table 1 imply that the mere existence of a feasible solution for a given set of constraints does not mean it will be easy to find.*

With respect to the dead-ends problem, a full dendrogram considering constraints can be obtained by switching from a hard interpretation of constraints to a soft one. This means that every level in the dendrogram tries to satisfy as many constraints as possible, but constraint violations are allowed in order for the algorithm to never reach a dead-end.

Some interesting results, both positive and negative, about the nature of pairwise constraints are proved and discussed in [DB07], as well as some workarounds for problems related to the use of constraints in clustering.

2.4 External validity indices

Validity indices are used to objectively evaluate the performance of a given method independently of the benchmarks it is tested in. This means that the output value of the validity indices is independent from the characteristics of the benchmarks datasets, such as their size or their number of features in the case of classification datasets. In the case of CC, one of the most popular external validity indices is the Adjusted Rand Index (ARI), which is an adjusted version of the previous Rand Index (RI). This section introduces both of these validity indices, as the ARI is the common quality measure in all the experimental studies presented in this thesis.

The RI measures the degree of agreement between two partitions. It can be used to measure the quality of a partition obtained by any CC algorithm by giving the ground-truth partition as one of them. Therefore, the two compared partitions are C and C^*. The RI views C and C^* as collections of $n(n-1)/2$ pairwise decisions. For each x_i and x_j in X, they are assigned to the same cluster or to different clusters by a partition. The number of pairings where x_i is in the same cluster as x_j in both C and C^* is taken as a; conversely, b represents the number of pairings where x_i and x_j are in different clusters. The degree of similarity between C and C^* is computed as in Equation 2 [Ran71], where n is the number of instances in X. The output value range for the RI is $[0, 1]$, with high values indicating a high level of agreement between the two partitions, and a low value indicating a low level of agreement.

$$\mathrm{RI} = \frac{a+b}{n(n-1)/2} \tag{2}$$

The RI can be conveniently formulated in terms of the elements of a confusion matrix as well [ZDX19]. Equation 3 defines these elements in terms of cluster memberships in a partition, which can be referred to as: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). Equation 4 makes use of these elements to give a new definition for the RI.

$$\begin{aligned}
\mathrm{TP} &= \{(x_i, x_j) | l_i^{C^*} = l_j^{C^*},\ l_i^C = l_j^C,\ i \neq j\} \\
\mathrm{FP} &= \{(x_i, x_j) | l_i^{C^*} = l_j^{C^*},\ l_i^C \neq l_j^C,\ i \neq j\} \\
\mathrm{TN} &= \{(x_i, x_j) | l_i^{C^*} \neq l_j^{C^*},\ l_i^C \neq l_j^C,\ i \neq j\} \\
\mathrm{FN} &= \{(x_i, x_j) | l_i^{C^*} \neq l_j^{C^*},\ l_i^C = l_j^C,\ i \neq j\}
\end{aligned} . \tag{3}$$

$$\mathrm{RI} = \frac{|\mathrm{TP}| + |\mathrm{TN}|}{|\mathrm{TP}| + |\mathrm{FP}| + |\mathrm{TN}| + |\mathrm{FN}|} \tag{4}$$

The ARI is the corrected-for-chance version of the RI. This correction is done by taking into account the expected similarity of all comparisons between partitions specified by a random model that acts as the baseline. This modifies the output value range of the original RI, transforming it into $[-1, 1]$ and slightly changing its interpretation. In ARI, a high output value still means a high level of agreement between the two partitions, and a low value means a low level of agreement. However, a value lower than 0 means that the results obtained are worse than those expected from the average random model. Equation 5 gives the formalization for the ARI [HA85].

$$\mathrm{ARI} = \frac{\mathrm{RI} - \text{Expected Index}}{\text{Maximum Index} - \text{Expected Index}}, \tag{5}$$

where Expected Index is the degree of similarity with a random model, Maximum Index is assumed to be 1, and RI is the RI value computed for partitions C and C^*. Both the RI and the ARI can measure the quality of any given CC method with respect to the ground truth by simply feeding the true labels into these indices as one of the partitions to be compared. Please note that CC methods are usually evaluated in classification datasets, as the constraint set needs to be generated on the basis of some kind of oracle, which is normally the labels set.

2.5 Multiobjective optimization

The Multiobjective Optimization Problem (MOP) is formalized as in Equation 6:

$$\begin{aligned} &\text{minimize} \;\; F(y) = (f_1(y), \cdots, f_m(y)) \\ &\mathbf{s.t.} \;\; y \in \Omega \end{aligned}, \tag{6}$$

where Ω is the variable space and $F : \Omega \rightarrow R^m$ consists of m real-valued functions (objective functions). R^m is known as the objective space and $\{F(y)|y \in \Omega\}$ defines the attainable object set. If $y \in R^n$ and Ω is defined as in Equation 7, with h_j being continuous functions, then the MOP in Equation 6 is said to be continuous.

$$\Omega = \{y \in R^n | h_j(y) \leq 0, j = 1, \cdots, m\}. \tag{7}$$

MOP techniques aim to balance all objective functions in Equation 6; this task is not trivial in the general case due to conflicts among the objective functions. An MOP technique finds a trade-off which can be defined in terms of Pareto optimality. Let $v, w \in R^m$, then, v dominates w if and only if $f_i(v) \leq f_i(w) \forall i \in \{1, \cdots, m\}$ and if $\exists j | f_j(v) < f_j(w), j \in \{1, \cdots, m\}$. This is: v dominates w if and only if v is better than w in at least one objective and as good as w in the rest, and is denoted as $w \prec v$.

A point $y^* \in \Omega$ is said to be Pareto optimal if there is no other point $y \in \Omega$ such that y dominates y^*. The set of Pareto optimal points is referred to as the Pareto Set (PS). Pareto Front (PF) is formed by the objective vectors associated with the points in PS. An MOP technique aims to find the best possible approximation to the PF for any given optimization task. An MOP definition for maximization problems can be obtained by reversing all inequalities.

In real-life applications of multiobjective optimization, a PF needs to be obtained so that a decision maker can later select the preferred solution. Being the MOP defined as above, there are no restrictions in the size of the PF so, in theory, very large or even infinite Pareto optimal vectors could be found in some cases. This is why obtaining the full PF is usually not feasible, or at least very time-consuming. Moreover, if the Pareto approximation to the PF is too large, the decision maker would have trouble choosing a solution from it due to the sheer amount of information. Most multiobjective optimization methods struggle to find a set of well-distributed Pareto optimal vectors with a reasonable size that constitutes a good approximation to the entire PF. Evolutionary algorithms have proven to be excellent at finding this approximation to the PF [Mie12], resulting in a whole new family of algorithms called MOEA [CLVV+14].

2.6 Distance metric learning

The vast majority of methods making up the data science algorithms and techniques corpus use distance measures. They are used to determine similarities between instances in the dataset from which we want to extract information. Clustering can be found among these techniques, with the assignation rule from the k-means algorithm being the foundation of the automatic clustering concept [M+67]. However, there is an infinite number of distance measures that can be employed for this task, and finding the one that better suits our dataset is crucial to obtain high quality results in any application. DML arises to meet this need, with algorithms capable of finding distance metrics that capture hidden features or relations in our datasets that standard measures like the Euclidean distance might miss. Combining DML algorithms and distance-based learning algorithms results in more complete and adaptive approaches to a wide variety of problems [SGH21].

One of the techniques that has helped developing DML is known as Learning from Side Information (LSI), sometimes also referred to as Mahalanobis Metric for Clustering [XJRN03]. It directly connects with the SSL paradigm, particularly with the constraint-based SSL area, as it incorporates side information referring to similar and dissimilar pairs of instances in the dataset, which can be easily compared with the must-link and cannot-link constraint sets. Given a pair of examples x_i and x_j, LSI can be viewed as a method to bring these instances closer if they are similar ($(x_i, x_j) \in C_=$) or space them out if they are dissimilar ($(x_i, x_j) \in C_{\neq}$). Formally, LSI searches for a positive semidefinite matrix $M \in S_d(\mathbb{R})_0^+$ optimizing Equation 8.

$$\begin{array}{ll} \min_M & \sum_{(x_i,x_j)\in C_=} \|x_i - x_j\|_M^2 \\ s.t.: & \sum_{(x_i,x_j)\in C_{\neq}} \|x_i - x_j\|_M \geq 1 \end{array} . \tag{8}$$

where $\|x_i - x_j\|_M = \sqrt{(x_i - x_j)^T M (x_i - x_j)}$. However, Equation 8 is hard to optimize with traditional methods, so its authors propose an equivalent form in Equation 9, which can be optimized using the projected gradient ascent method.

$$\begin{array}{ll} \max_M & \sum_{(x_i,x_j)\in C_{\neq}} \|x_i - x_j\|_M \\ s.t.: & \sum_{(x_i,x_j)\in C_=} \|x_i - x_j\|_M^2 \leq 1 \end{array} . \tag{9}$$

2.7 Monotonicity constraints in classification

Monotonicity constraints were originally integrated into the supervised learning classification task, leading to monotonic classification. It can be viewed as a special case of standard classification where the classes constitute a set of ordered categories. Monotonic classification models must respect monotonicity constraints between the feature values of the instances and their class labels [CGK+19].

Formally, monotonic classification aims to predict the class label y_i from an instance x_i with $y \in \mathcal{Y} = \{l_1, \cdots, l_m\}$. The categories in $\mathcal{Y}$ are arranged in an order relation $\prec$ in the form $l_1 \prec l_2 \prec \cdots \prec l_m$. In doing so, features and class labels are monotonically constrained by the background knowledge, i.e., $x_i \succeq x_j \rightarrow f(x_i) \geq f(x_j)$, where $x_i \succeq x_j$ implies that all features in x_i compare to features in x_j with operator $\geq$, this is: $x_{i,q} \geq x_{j,q}\ \forall q \in \{1, \cdots, u\}$ [KS12]. This relationship between instances is referred to as dominance. In this case x_1 dominates x_2. The goal of monotonic classification is to build a classifier that does not violate monotonicity constraints (pairwise dominance relationships). The result is a monotonic classifier [CGK$^+$19].

Much in the same way as it is done with CC methods, a distinction can be made in monotonic classifiers: soft monotonic models try to minimize the number of monotonic constraint violations, while hard monotonic models always produce monotonic predictions (never violate monotonic constraints) [GGL$^+$21].

2.7.1 Partially ordered data clustering in MCDA

In [RDSDD21] the monotonicity constraints are integrated into unsupervised learning to produce the ordered clustering framework. Particularly, they are integrated into the MCDA paradigm, which is a subfield of operational research that concerns the structuring and resolution of decision problems including multiple criteria [Roy96]. To do so, the classic symmetrical notion of distance in pattern recognition is replaced with the asymmetrical notion of preference from the MCDA paradigm. The preference of an instance over another evaluates the global advantages of the former over the latter according to some preference criteria. The notion of preference can be seen as a decomposition of a distance measure by taking into account the sign of the differences. To cluster instances in an MCDA context, the similarity between every pair of instances is evaluated in terms of preferences taking all the other alternatives into account. With this in mind, two instances are similar if they both rank either higher or lower in preference with respect to the same set of instances. To formalize these concepts, let us consider the weighted L_1 distance (for the maximization case and without loss of generality) as in Equation 10, which can be simplified as in Equation 11, with $w_d \in [0, 1]$ being the weight assigned to the dth feature.

$$L_1(x_i, x_j) = \sum_{d=1}^{u} w_d |x_{[i,d]} - x_{[j,d]}|. \tag{10}$$

$$L_1(x_i, x_j) = \sum_{d: x_{[i,d]} > x_{[j,d]}}^{u} w_d x_{[i,d]} - w_d x_{[j,d]} + \sum_{d: x_{[j,d]} > x_{[i,d]}}^{u} w_d x_{[j,d]} - w_d x_{[i,d]}. \tag{11}$$

Consequently, let us define the preference of x_i over x_j as in Equation 12. To put this into words, $r(x_i, x_j)$ quantifies the sum of differences between x_i and x_j limited to the features in which x_i has higher (lower) values than x_j for the maximization (minimization)

case. Intuitively, the preference $r(x_i, x_j)$ indicates the cumulative quantified value of the advantage of x_i over x_j. Please note that, as it has already been mentioned, the preference is not symmetrical: $r(x_i, x_j) \neq r(x_j, x_i)$ in most cases.

$$r(x_i, x_j) = \sum_{d : x_{[i,d]} > x_{[j,d]}}^{u} w_d x_{[i,d]} - w_d x_{[j,d]}. \tag{12}$$

Finally, note that the weighted L_1 distance between two instances can always be expressed as in Equation 13. This decomposition can be done the same way for any L_p distance.

$$L_1(x_i, x_j) = r(x_i, x_j) + r(x_j, x_i). \tag{13}$$

3 Justification

SSL has become an important area of research in the field of ML, especially in applications where labeled data is scarce or expensive to obtain. CC is a specific form of SSL where constraints are imposed on the clustering process to guide the formation of clusters. These constraints can be in the form of pairwise instance relationships, class labels, or other forms of domain knowledge. The incorporation of such constraints can significantly enhance the quality of clustering and make it more useful in practice. As such, a thesis focused on CC in the context of SSL can make significant contributions to both the theory and practice of ML. The specific reasons that motivate this thesis are listed below.

- Firstly, CC is an important and challenging problem in the field of ML, and it has a wide range of practical applications in various fields such as bioinformatics, image and video analysis, and natural language processing, among others. Therefore, the development of new and improved techniques for CC is highly desirable, and it has the potential to make a significant impact on many areas of research and industry.
- Secondly, the current state-of-the-art in CC is limited, and there is a significant gap between the existing techniques and the desired outcomes. This gap is an opportunity for innovative research to address the limitations of current methods and raise the standards of solution quality in CC.
- Thirdly, the proposed thesis project aims to contribute to the development of a unified framework and formal taxonomy for CC. This is a necessary step towards a better understanding of the field and the development of more efficient and effective techniques.
- Fourthly, the project proposes the application of evolutionary metaheuristics to CC, which is a promising line of research that has not been fully explored yet. This presents an opportunity to develop novel algorithms that considerably boost the performance of CC methods.
- Finally, the project encourages the exploration of the links between CC and other non-standard learning paradigms, which can lead to further innovation in information combination techniques.

In summary, a thesis focused on CC is justified due to the relevance and the challenging nature of the problem, the shortcomings of current techniques, the need for a unified framework and formal taxonomy, the potential for the application of evolutionary metaheuristics, and the opportunity to explore the relationship with other non-standard learning paradigms. The results of this research can have a substantial impact on various fields and can pave the way for further research and innovation in CC.

4 Objectives

Once the main concepts of the state-of-the-art have been introduced, the objectives that have driven this thesis can be elaborated on. In the first place, they include the creation of a previously nonexistent general reference in the CC area, which serves as a comprehensive survey of the state-of-the-art in the area. Afterwards, and with the knowledge gathered in the mentioned study as a foundation, CC is tackled from different points of view, including metaheuristic and classic clustering-based approaches. Finally, the relationship between CC and other non-standard learning paradigms is addressed. These objectives can be broken down as follows:

Creation of a taxonomy and a ranking of CC methods. To fulfill this first objective, it is necessary to carry out an analysis of existing CC methods, focusing on their similarities and differences to produce a consistent taxonomy. This taxonomy may contain categories that are present in classic clustering surveys, existing categories in the area of CC and, ideally, newly identified CC categories. An interesting byproduct of this study will be a statistical analysis of the most frequent experimental setup when testing the capabilities of any given CC method. It can be obtained by carefully gathering information from the experimental section of the papers in the analyzed corpus. We aim to produce a ranking of all CC methods analyzed that can later be used to identify the most promising ones.

Study of CC from the point of view of metaheuristics. As mentioned in previous sections, CC is intractable in practice; hence, metaheuristics represent a promising approach. In this second objective, CC is addressed from the single-objective and multi-objective optimization paradigms. Firstly, a single-objective metaheuristic approach must be considered to tackle CC. This can be done by designing a suitable representation scheme and a proper objective function, which ideally will combine classic clustering and CC indicators. Secondly, multi-objective approaches are considered to address CC. Metaheuristics have proven to be excellent optimization algorithms in this area, and therefore the combination of both must be taken into account. To this end, the creation of a new representation scheme is needed, and the set of objective functions to be optimized must be chosen.

Hybrid models for CC. How to integrate constraints into the clustering process remains one of the major challenges within the CC area. Usually, the combination of multiple integration models results in higher performance when the constraint set is free of noise and other imperfections. Two categories shape the landscape of CC methods: constrained partitional methods and constrained DML methods. As these two paradigms are compatible with each other, the combination of both, with the redundant subsequent inclusions of constraints, is promising in scenarios with reliable constraint sets. To fulfill this third objective, the combination of CC models which belong to the two major CC categories must be studied. Ideally, this would result in a new hybrid CC model with the inclusion of redundant constraints.

Combining multiple types of background knowledge. Constraints are not the only type of background knowledge that has drawn attention from the scientific community. In fact, there are many variations of classic ML problems whose existence is derived from the modification of the initial conditions or the information given to a classic one. Real-world applications are not limited to a single type of background knowledge. In fact, there is no limit in this regard. With this in mind, the combination of instance-level pairwise constraints and other types of background knowledge is studied in this fourth objective.

5 Methodology

The research conducted throughout this thesis has been carried out following the scientific method. In this particular case, it requires both practical and theoretical methodologies. The general guidelines applied in all studies included in this thesis are summarized here:

- **Observation**: through the study of the SSL task, and focusing on CC. The goal of this stage is to identify research opportunities, which could result in new, successful models to address CC and extend its applicability.
- **Formulation of hypotheses**: design of new CC algorithms, with an emphasis on their scalability with respect to the amount of constraint-based information available. The models designed and developed must fulfill the objectives described in previous sections.
- **Experimental data collection**: the designed models are tested on diverse scenarios to obtain results as representative of their capabilities as possible. These results are later analyzed using external quality indices.
- **Contrasting the hypotheses**: the results obtained are compared with representative models from the existing literature, with the aim of analyzing their quality in terms of efficiency and effectiveness. To this end, a set of representative models is chosen on the basis of a comprehensive literature review. These methods are implemented and published, for the sake of reproducibility of results.
- **Validation of hypotheses**: hypotheses formulated in the experiments are proven or disproven following objective quality indicators and statistical testing. If any given hypothesis is rejected, it must be modified and the previous steps repeated from that point on.
- **Scientific thesis**: relevant conclusions are extracted in view of the outcomes of the research process. All the results and conclusions obtained must be gathered and synthesized into a documentary report of the thesis.

6 Summary

The body of knowledge compiled in this thesis is found in 5 different studies, published in scientific journals and conferences. The aim of this section is to summarize and introduce these studies, whose results will be discussed later (in Section 7). The publications are listed below:

- González-Almagro, G., Peralta, D., De Poorter, E., Cano, J. R., & García, S. (2023). Semi-Supervised Constrained Clustering: An In-Depth Overview, Ranked Taxonomy and Future Research Directions. *arXiv preprint arXiv:2303.00522*.

- González-Almagro, G., Luengo, J., Cano, J. R., & García, S. (2020). DILS: constrained clustering through dual iterative local search. Computers & Operations Research, 121, 104979. DOI: `https://doi.org/10.1016/j.cor.2020.104979`.

- Gonzalez-Almagro, G., Rosales-Perez, A., Luengo, J., Cano, J. R., & Garcia, S. (2021). ME-MEOA/Dcc: Multiobjective constrained clustering through decomposition-based memetic elitism. Swarm and Evolutionary Computation, 66, 100939. DOI: `https://doi.org/10.1016/j.swevo.2021.100939`.

- Gonzalez-Almagro, G., Suárez, J. L., Luengo, J., Cano, J. R., & García, S. (2022). Three Stages Hybrid Agglomerative Constrained Clustering (3SHACC): Three stages hybrid agglomerative constrained clustering. Neurocomputing, 490, 441-461. DOI: `https://doi.org/10.1016/j.neucom.2021.12.018`.

- González-Almagro, G., Suárez, J. L., Sánchez-Bermejo, P., Cano, J. R., & García, S. (2023). Semi-supervised Clustering with Two Types of Background Knowledge: Fusing Pairwise Constraints and Monotonicity Constraints. *arXiv:2302.14060*.

The remainder of this section is organized according to the publications listed above and the objectives described in Section 4. Firstly, Section 6.1 presents a summary of the newly-created taxonomy. The knowledge gained through the study of the scientific corpus analyzed to create the taxonomy is leveraged in subsequent studies to design a suitable experimental setup. In Section 6.2, this experimental setup is used to apply two metaheuristic-based methods to CC. One of them implements a single-objective optimization procedure, while the other is designed to handle multi-objective problems. Afterwards, Section 6.3 introduces a new hybrid optimization method for CC which combines DML and classic HC techniques. Additionally, it proposes a completely unsupervised and automatic constraint weighting procedure. Finally, Section 6.4 studies the combination of two types of background knowledge: instance-level pairwise constraints, and monotonicity constraints. The resulting method is applied in a real-world scenario which had never been addressed from such point of view.

6.1 Creation of a taxonomy and a ranking of CC methods

The last published survey of the state-of-the-art in the field of CC dates back to the year 2007. Since then, a plethora of new types of background knowledge and CC methods have been discovered. The goal of this study is two-fold. Firstly, it serves as the general reference the CC area has been lacking for years. Secondly, it helped us to learn about the intricacies of SSL and CC, such as the differences between approaches, the experimental procedures used to prove their capabilities, and the research opportunities.

Overall, the study provides the reader with everything needed to understand the CC problem, from basic to advanced concepts. It starts with a taxonomy of types of background knowledge, which were found during the process of building the corpus of CC methods that had to be reviewed; overall, 33 different types of background knowledge were found which could be organized into 5 families. Next, it introduces all the necessary background related to classic clustering and CC, focusing on CC and diving into the relevant advanced concepts and structures. This includes its early history and a review of its applications. Afterwards, a statistical study on the experimental elements used to demonstrate the capabilities of CC methods is presented, which later constitutes the foundation of an objective scoring system. We believe that this scoring system is one of the most valuable contributions of this study. It can be employed to evaluate and rank the studies belonging to any scientific corpus; in this particular case, it gives the reader an objective toolkit with which to navigate the literature in a more effective way. Finally, a taxonomy of 307 CC methods is proposed. A total of 29 CC categories are identified: some of them are classic clustering categories, others are CC categories found in previous studies, and the remaining ones are novel categories first identified in this study. A statistical analysis of this taxonomy is also carried out. The publication associated with this study is:

> González-Almagro, G., Peralta, D., De Poorter, E., Cano, J. R., & García, S. (2023). Semi-Supervised Constrained Clustering: An In-Depth Overview, Ranked Taxonomy and Future Research Directions. *arXiv preprint arXiv:2303.00522*.

6.2 Study of CC from the point of view of metaheuristics

CC can also be expressed as an optimization problem, which allows for the application of many optimization techniques, although the resolution itself is no trivial task. In this context, metaheuristic algorithms are becoming increasingly popular for finding approximate solutions of sufficient quality. These algorithms work by exploring the solution space using a fitness function and balancing exploration and exploitation. While metaheuristics have been successfully applied to classic clustering, little work has been done on their suitability for CC, particularly in highly-constrained environments. To fulfill the objectives associated with this section, the CC problem is addressed from a metaheuristics point of view, and in two completely different setups: the single-objective optimization paradigm, and the multi-objective optimization paradigm.

6.2.1 CC through single-objective optimization metaheuristics

A widely-known metaheuristic algorithm is the Iterative Local Search (ILS), which is a variant of Local Search (LS) that periodically introduces perturbations to escape local optima. ILS has been successfully applied in a wide range of applications, including the traveling salesman, the quadratic multiple knapsack, and the vehicle routing problem. It is often used in combination with other techniques to achieve good exploration-exploitation trade-offs. In the first study associated with this section, we propose a new variant of ILS called Dual Iterative Local Search (DILS) that combines the exploitation capability of ILS with classic diversity-introducing techniques from genetic algorithms such as recombination and mutation operators. DILS also includes a restarting mechanism to escape from local optima and can optimize two individuals at the same time to manage the exploration of the solution space.

We have developed an application of DILS for CC called DILS$_{CC}$ which uses an integer-based representation scheme and a penalty-style fitness function. We have demonstrated that DILS is a competitive approach for obtaining high-quality solutions to the CC problem, particularly in highly-constrained environments. This is due to the exploitation capability of DILS, which allows for the quality of the results to scale with the amount of constraints. The publication associated with this study is:

> González-Almagro, G., Luengo, J., Cano, J. R., & García, S. (2020). DILS: constrained clustering through dual iterative local search. Computers & Operations Research, 121, 104979. DOI: `https://doi.org/10.1016/j.cor.2020.104979`.

6.2.2 CC through multi-objective optimization metaheuristics

MOEAs are effective at optimizing objective functions that may have conflicting goals, such as CC where traditional clustering objectives create spherical clusters, while CC objectives deviate from this trend.

In the study associated with this section, we propose a memetic elitist version of Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) called Memetic Elitist - MOEA/D (ME-MOEA/D) for CC. Memetic algorithms are a class of optimization algorithms that incorporate both global search strategies and local search procedures in order to improve the quality of solutions found and accelerate the convergence of the underlying genetic algorithm. Unlike MOEA/D, our proposal implements memetic elitism by applying any single-objective optimization procedure to select elite individuals using a dominance-guided sorting mechanism. ME-MOEA/D can also adaptively select the single-objective function to optimize for each individual at any stage of the optimization process, while allowing for user control through parameter settings.

ME-MOEA/D fuses classic multiobjective optimization methods with single-objective procedures such as LS, allowing it to produce high-quality results for the CC problem. We

improve its application to CC by introducing a new initialization method, a biased crossover operator, and an external population limiting method. These genetic components, combined with a memetic elitist version of MOEA/D, have not been studied previously. The publication associated with this study is:

> Gonzalez-Almagro, G., Rosales-Perez, A., Luengo, J., Cano, J. R., & Garcia, S. (2021). ME-MEOA/Dcc: Multiobjective constrained clustering through decomposition-based memetic elitism. Swarm and Evolutionary Computation, 66, 100939. DOI: `https://doi.org/10.1016/j.swevo.2021.100939`.

6.3 Hybrid models for CC

The aim of the study associated with this objective is to explore the use of hybrid models in CC. Most notably, it focuses on the combination of constrained agglomerative HC methods and constrained DML methods. Methods from this paradigm will combine distance-based techniques with clustering-engine adapting techniques to effectively address CC. In addition, we propose the use of weighted constraints to guide the clustering process towards higher-quality solutions. Nonetheless, there is currently limited research on the automatic generation of constraint weights and their integration into distance-based CC methods. Our study aims to fill these gaps by introducing the 3SHACC method. This method can obtain a full dendrogram that captures the intricate manifold structures present in a dataset and incorporates constraints into the process. It accomplishes this through three well-defined stages: (1) the relevance of every constraint is determined and a new metric is built on the basis of the newly weighted constraint set by using a new DML method; (2) the similarities among instances in the dataset are calculated according to the newly computed distance metric and the pairwise reconstruction coefficient; and (3) a dendrogram is obtained by running a classic Agglomerative Hierarchical Clustering (AHC) method with a constraint-biased stepped affinity function integrating the computed similarities and the information contained in the constraint set. Note that, even if these three stages are designed to be applied together, they are independent from each other, and can be incorporated into other CC methods as intermediate steps. The publication associated with this study is:

> Gonzalez-Almagro, G., Suárez, J. L., Luengo, J., Cano, J. R., & García, S. (2022). 3SHACC: Three stages hybrid agglomerative constrained clustering. Neurocomputing, 490, 441-461. DOI: `https://doi.org/10.1016/j.neucom.2021.12.018`.

6.4 Combining multiple types of background knowledge

As stated in Section 6.1, many types of background knowledge can be found in the literature. To fulfill the objective associated with this section, we assume CC methods are not limited to one of them, and that there are applications where more than one type of background

knowledge can be available. Consequently, methods which are capable of handling various type of background knowledge have to be developed. We aim to find an application where pairwise instance-level constraints and other types of constraints are both available, and thus where a model that combines both of them is necessary.

In recent years, unsupervised learning has incorporated a new type of background knowledge from the supervised learning paradigm. Monotonic classification is a type of supervised learning where classification models must follow monotonicity constraints among instances based on their descriptive features. This means that the classes, which are ordered categories, must respect the ordering relationship between instances based on their feature values. For instance, in the context of house pricing, bigger houses are expected to have higher prices than smaller ones in the same neighborhood when other features are similar. This order relationship between instances based on their feature values can be leveraged to produce more accurate predictive models. Monotonicity constraints have been successfully applied in various real-world problems such as fraudulent firm classification, real-time dynamic malware detection, and analysis of learning activities based on student opinion surveys. A methodology to perform clustering in the presence of monotonicity information, known as ordered clustering, has also been introduced within the MCDA framework. This methodology involves defining a distance measure grounded on the concept of preference. The gist of it lies in comparing instances in the dataset by examining the comparative relationships of their features, resulting in a distance measure that produces ordered labeling which follow monotonicity. This approach is similar to the monotonic classification models discussed earlier.

The study associated with this objective aims to combine two types of background knowledge —pairwise constraints and monotonicity constraints—to address real-world problems like the Shanghai Ranking of World Universities (SRWU) dataset partitioning. While previous studies have combined monotonicity constraints with other types of constraints like cluster-size constraints, the combination of monotonicity and pairwise constraints remains unexplored. This study proposes the Monotonic Constrained Clustering (MCC) paradigm, which includes pairwise constraints into the ordered clustering process. To optimize the MCC objective function, an Expectation-Minimization (EM) scheme called Pairwise Constrained K-Means - Monotonic (PCKM-Mono) is proposed. This is the first study to acknowledge and tackle the logical relationship between monotonic classification and ordered clustering, and the hybrid objective function proposed here combines a monotonic distance metric and a penalty term for pairwise constraint violations. The publication associated with this study is:

González-Almagro, G., Suárez, J. L., Sánchez-Bermejo, P., Cano, J. R., & García, S. (2023). Semi-supervised Clustering with Two Types of Background Knowledge: Fusing Pairwise Constraints and Monotonicity Constraints. *arXiv preprint arXiv:2302.14060*.

7 Discussion of Results

With the exception of the first objective, the rest of them involve experimental procedures aimed at proving the capabilities of the proposed methods. A consistent experimental methodology ensures that the research findings are reliable and valid and produce robust evidence to support the research hypotheses. Homogeneity in the experimental procedures guarantees that the same process is followed consistently throughout the research, minimizing the effects of extraneous variables and ensuring that the results are comparable. Statistical validation procedures safeguard against the effects of mere chance and help to ascertain the statistical significance of any patterns or relationships observed in the data. In general, statistical techniques help to ensure that the findings are robust and can be generalized to the wider population. Justifying homogeneity in experimental procedures and statistical validation procedures demonstrates rigor and validity in the research and builds confidence in the findings among peers and the wider academic community.

With these guidelines in mind, all the experimental studies carried out to fulfill the objectives of this thesis follow the same structure and use shared external validity indices and statistical testing procedures. The findings derived from the study that meets the first objective (the survey of the state-of-the-art) are used to particularize experimental studies. The set of benchmarks, external validity indices, and statistical testing procedures are decided according to that study. Particularly, the set of benchmarks found in every experimental study consist of a series of classification datasets, which is as similar to the rest as the characteristics of each proposal reasonably allow. No preprocessing other than standard normalization is applied to these datasets in any case. In all cases, the capabilities of the methods under comparison are tested in three different levels of constraint-based information, and conclusions are always supported with Bayesian statistical testing procedures. Regarding the external quality measures, different studies need to test different capabilities of the compared methods, so the set of measures varies from one to another. However, the ARI is used in all the studies and that can serve as a unifying thread between them, making comparisons possible. Regarding the methods used in the experimental comparisons, the baseline methods found in our survey of the state-of-the-art, such as COPKM and LCVQE, are always present. Nevertheless, given the differences that exist among our proposals, other state-of-the-art methods in the particular area to which each method belongs have to be considered.

This section summarizes the analysis of results obtained to fulfill the objectives of this thesis. It also briefly discusses the particularities related to the experimental setups in each study. Similarly to Section 6, the remainder of this section is organized according to the publications and the objectives introduced in Section 4. Section 7.1 provides the conclusions drawn from the creation of the taxonomy and its statistical analysis. Section 7.2 summarizes the results obtained with the metaheuristic approaches DILS and ME-MOEA/D. The results obtained by our hybrid model 3SHACC are shown in Section 7.3. Finally, Section 7.4 contains the results obtained by PCKM-Mono.

7.1 Creation of a taxonomy and a ranking of CC methods

The study related to this objective produced a systematic review of the field of CC, which is a type of SSL that incorporates background knowledge into the clustering process. It provides an introduction to SSL and discusses the area of semi-supervised clustering, highlighting the types of background knowledge that can be included as constraints. It argues that instance-level pairwise Must-Link (ML) and Cannot-Link (CL) constraints are the most successful types of constraints in CC. It formalizes CC and gives examples of practical applications. Then, advanced CC concepts and structures are described, analyzing the advantages and disadvantages of different approaches. A statistical analysis of the experimental elements present in studies proposing new CC methods serves as the basis to create an objective scoring system to evaluate each approach. This is the system through which 307 CC methods are ranked and split into two major families—constrained partitional and constrained DML. The methods in each family are further divided into more specific categories, and their features and elements are described in detail.

The proposed taxonomy can be used to:

- Decide which type of approach and model is best suited for a new CC problem.
- Compare newly proposed techniques to those in the same family in this taxonomy, so that it can be determined whether the new method represents an improvement over the current state-of-the-art.
- Identify the proposals which best support their conclusions and propose more robust methods, thanks to the scoring system.

As a result of this research, the main flaws and criticism related to the CC area can be identified. Having reviewed 270 studies (proposing 307 methods), we have identified 5 serious shortcomings which affect the vast majority of them. These drawbacks can be summarized as follows: an absence of a unified, general reference; low amount of application studies; a lack of extensive experimental comparisons; unavailability of dedicated, standardized CC-oriented datasets and constraint sets; and statistically unsupported experimental conclusions.

7.2 Study of CC from the point of view of metaheuristics

The two studies related to this objective prove that both single-objective and multi-objective metaheuristics can produce high-quality results in the CC problems. In particular, DILS and ME-MOEA/D have been proven to outperform the state-of-the-art in their respective fields. Sections 7.2.1 and 7.2.2 discuss the particularities and the results obtained with these two methods, respectively.

7.2.1 CC through single-objective optimization metaheuristics

Our single-objective metaheuristic proposal to address CC is called DILS. It is tested on 25 different (unpreprocessed) datasets with three different constraint sets that contain increasing levels of constraint-based information for each one. The comparison includes 6 existing approaches to CC. One of the is the state-of-the-art in single-objective metaheuristics applied to CC (BRKGA+LS) and the rest are baseline CC algorithms (COPKM, LCVQE, RDPM, TVClust and CECM). The external quality measure used to compare results is ARI in this case.

The Bayesian signed rank test provides evidence in favor of DILS when handling highly constrained problems. Nonetheless, it is worth acknowledging that, in some cases, the results obtained by the DILS are not superior in certain datasets to those from a random model. Concerning this caveat, it should be noted that no parameter optimization was performed in this study. Hence, it is possible that different parameter choices could lead to better outcomes in these datasets.

7.2.2 CC through multi-objective optimization metaheuristics

Our multi-objective metaheuristic proposal to address CC is called ME-MOEA/D. The experimental study carried out to test its capabilities is, by far, the most extensive one presented in this thesis. This is because ME-MOEA/D has to be tested in many different environments and from different points of view. First, as it is a CC method, it must be compared with baseline CC methods. Secondly, since it optimizes a single objective function, it must also be compared with the state-of-the-art in single-objective metaheuristics applied to CC. Finally, it is an MOEA, so it must be compared with the state-of-the-art in MOEAs applied to CC. In view of this, ME-MOEA/D is tested on 20 different (unpreprocessed) datasets, again, with three different constraint sets that contain increasing levels of constraint-based information. Regarding the competing methods, 8 existing approaches to CC are considered: four baseline CC algorithms (COPKM, LCVQE, RDPM, and TVClust); one algorithm from the state-of-the-art in single-objective metaheuristics applied to CC (SHADE$_{CC}$); two algorithms from the state-of-the-art in MOEAs applied to CC (MOCK and PESA-II); and lastly the classic MOEA/D, which was the foundation of ME-MOEA/D.

Regarding the external quality measures, two sets of them are employed. Firstly, ARI and Unsat are used to assess the capabilities of the various methods with according to the quality of the partition they obtain and the ability to integrate constraints. Secondly, the Pareto approximations obtained by the MOEAs are compared by means of three classic, well-established, Pareto-related measures: the Pareto size, the Hypervolume, and the ϵ^+-indicator.

The Bayesian signed rank test provides evidence in favor of ME-MOEA/D*CC* consistently outperforming all other compared methods. ME-MOEA/D*CC* also produced Pareto approximations with better individual solutions compared to previous approaches, even if it produced more compact Pareto approximations.

7.3 Hybrid models for CC

Our hybrid CC model is called 3SHACC. It has been tested on 25 (unpreprocessed) datasets and 3 constraint sets with increasing levels of constraint-based information for every one of the datasets. It is compared with 6 previous approaches to CC: one of them is its predecessor 2SHACC, and the rest are CC baseline algorithms (COPKM, LCVQE, RDPM, TVClust and PCSKM). An assessment about the influence of the hyperparameters of 3SHACC on its results is also made in the study related to this objective. The ARI and the Unsat quality indicators are used to compare the results obtained by all these methods. The total number of optimal results is also taken into account to draw conclusions.

Our experimental results demonstrate that 3SHACC consistently outperforms classic approaches to CC and its predecessor 2SHACC in both ARI and Unsat measures. Furthermore, our findings suggest that 3SHACC is capable of scaling the quality of the results with the amount of constraint-based information available to a greater extent than classic approaches. These conclusions are supported by the Bayesian signed-rank test, which assigns a significantly higher probability to our proposal being better on average than any other method studied.

7.4 Combining multiple types of background knowledge

Our method PCKM-Mono combines pairwise instance-level constraints and monotonicity constraints. In this case, the study related to this objective needs to prove not only the capabilities of our proposal, but also the practical relevance of the matter it addresses. For this reason, the experimental setup found in the study differs from the previous ones in some aspects. Firstly, the set of benchmarks is different from the one found in previous studies, as it has to include monotonic datasets to demonstrate the ability of PCKM-Mono to integrate monotonic constraints into the clustering process. Secondly, the set of external quality indicators is extended to include monotonicity-related measures. Unfortunately, there is no standard measure that can take both pairwise instance-level constraints and monotonicity constraints into account, and hence these two aspects have to be analyzed separately. The ARI measure is used to evaluate results from a clustering quality point of view, the Unsat is used to evaluate the ability of the different methods to integrate pairwise instance-level constraints, and the Non-Monotonic Index (NMI) is used to evaluate monotonicity-related capabilities (not to be mistaken with the Normalized Mutual Information).

The set of compared methods must include classic clustering methods (to prove the practical relevance of the problem and the need for novel approaches to tackle it), CC baseline methods, and monotonic clustering methods. To this end, 5 methods are compared: the classic Kmeans, the P2Clust monotonic clustering method, and the COPKM and PCSKM CC methods. All of them use an EM scheme similar to the one employed by PCKM-Mono.

In addition to the traditional set of benchmarks, the dataset of the problem that motivated the creation of PCKM-Mono is considered in the study. A detailed analysis of the Shanghai Ranking of World Universities (SRWU) dataset and the results obtained by all

compared methods has been conducted.

The statistical analysis of the results (through Bayesian statistical testing) confirms the advantage of PCKM-Mono over purely monotonic and purely pairwise CC techniques. While PCKM-Mono achieves similar results to previous methods for specific monotonicity and pairwise constraint satisfaction, it is statistically superior in terms of general clustering quality measures. Additionally, when it comes to a specific problem such as the partitioning of the SRWU dataset, the experimental results continue to provide evidence in favor of PCKM-Mono, as purely CC methods cannot take monotonicity constraints into account, and purely monotonic clustering methods cannot deviate from perfect monotonicity, which SRWU does not feature.

8 Conclusions and Future Work

This section concludes the thesis (Section 8.1), gathers all the relevant studies we have published (Section 8.2), and provides notes on future research lines (Section 8.3).

8.1 Conclusions

This thesis presents an extensive study of CC that provides both a comprehensive view on the work already done in the area and innovation in the form of four new CC methods. The overarching goal of this thesis is to broaden the current knowledge about CC and to address the problem from new perspectives. In order to do so, the most systematic literature review ever produced in the area was carried out, and comprehensive experimental studies were conducted to prove the potential of our proposals to achieve higher standards than the previously published alternatives.

To accomplish the first objective, the most extensive CC literature corpus has been gathered and analyzed, resulting in a new taxonomy of the types of constraints, and in a taxonomy of CC methods which includes newly discovered categories. A statistical analysis of both the experimental setups used to prove the capabilities of the methods gathered in the taxonomy, and the taxonomy itself, provide the necessary knowledge to develop subsequent studies, including the four new proposals described in this thesis. As a conclusion to this objective, we encourage research on the CC area to observe the research guidelines provided in the study associated with it. Through them, researchers can easily identify the unsolved challenges in the area, and focus their efforts on them.

The second objective is the broadest one, since its completion has involved two extensive experimental studies. This objective focuses on CC from a metaheuristics point of view, and aims to approach its resolution through both a single-objective and a multi-objective optimization algorithm. The two methods—DILS and ME-MOEA/D—are designed to realize these two approaches. DILS is a single-objective metaheuristic algorithm based on ILS, and ME-MOEA/D is an MOEA which introduces single-objective-based memetic elitism procedures into the classic MOEA/D method. As a conclusion to this objective, it is reasonable to think that both DILS and ME-MOEA/D overcome the drawbacks that the previous proposals in their respective areas suffered from. DILS constitutes a remarkable improvement over previous single-objective metaheuristic approaches to CC with respect to clustering quality. The same can be said about ME-MOEA/D, which also produces better results than previous proposals in multi-objective-related quality measures.

Hybridizations in the CC area are addressed as part of the third objective of this thesis. The research carried out for this objective proposes the three-staged algorithm 3SHACC. In its three stages it implements an automatic constraint weighting procedure, a constrained DML algorithm capable of handling constraint weights, and an affinity-based hierarchical CC method which produces the output partition. It is clear that 3SHACC is a hybrid CC model, as it combines techniques from the two broadest families of CC methods, namely:

partitional CC, and constrained DML. As a conclusion to this objective, we can highlight the outstanding performance of 3SHACC when compared to CC baseline algorithms. 3SHACC is capable of achieving optimum results much more frequently than any other of its alternatives, and its modularity allows researchers to reuse parts that can benefit their own research.

The fourth and last objective concerns the task of combining multiple types of background knowledge, under the assumption that, in real-world problems, many types of background knowledge can be available to perform learning. The study we proposed to fulfill this objective tackles the combination of pairwise instance-level constraints and monotonicity constraints. To do so, a simple EM style algorithm is designed, called PCKM-Mono. It uses a constraint-based penalty term to include constraints in the monotonic clustering paradigm, thus producing a solution to monotonic CC. The practical relevance of this problem is proved by the existence of datasets such as the SRWU. As a conclusion to this objective, the experimental results in both standard benchmarks and specifically in the SRWU dataset show the superiority of our proposal over classic, purely constrained and purely monotonic clustering methods. These conclusions are supported by Bayesian statistical testing performed on quality measures regarding clustering quality, constraint violations, and monotonicity satisfaction.

Conclusiones

Esta tesis presenta un amplio estudio en CC que proporciona tanto una visión global de los trabajos ya realizados en el área como un enfoque innovador a través de cuatro nuevos métodos de CC. El objetivo general de esta tesis es ampliar el conocimiento actual sobre el CC y estudiarlo desde nuevas perspectivas. Para ello, se ha llevado a cabo la revisión bibliográfica más extensa hasta la fecha y se han hecho estudios experimentales exhaustivos para demostrar el potencial de nuestras propuestas y superar los resultados de las alternativas existentes.

Para lograr el primer objetivo, se ha recopilado y analizado un extenso corpus bibliográfico del CC, que ha conllevado la creación de una nueva taxonomía de tipos de restricciones, así como una taxonomía de métodos de CC que incluye nuevas categorías. Un análisis estadístico tanto de los procedimientos experimentales utilizados para probar las capacidades de los métodos recogidos en la taxonomía como de la propia taxonomía proporciona el conocimiento necesario para desarrollar estudios posteriores, incluyendo las cuatro nuevas propuestas descritas en esta tesis. En las conclusiones del estudio asociado a este apartado, se insta a los investigadores en el área a observar las pautas de investigación que proporciona el estudio. A través de dichas pautas, podrán identificar fácilmente los problemas en las fronteras del área y centrar sus esfuerzos en ellos.

El segundo objetivo es el más amplio, ya que su realización ha implicado dos extensos estudios experimentales. Este objetivo se centra en el CC desde un punto de vista metaheurístico y pretende resolverlo a través de un algoritmo de optimización monoobjetivo y otro multiobjetivo. Los dos métodos—DILS y ME-MOEA/D—están diseñados para implementar estos dos enfoques. DILS es un algoritmo metaheurístico monoobjetivo basado en ILS, mien-

tras que ME-MOEA/D es un MOEA que introduce elitismo memético monoobjetivo en el método clásico MOEA/D. De este objetivo se extrae que es razonable pensar que tanto DILS como ME-MOEA/D superan los inconvenientes de las propuestas anteriores en sus respectivas áreas. DILS constituye una notable mejora respecto a las anteriores aproximaciones metaheurísticas de monoobjetivo al problema CC en lo que se refiere a la calidad del *clustering*. Lo mismo puede decirse de ME-MOEA/D, que también produce mejores resultados que las propuestas anteriores en medidas de calidad multiobjetivo.

Las hibridaciones en el área del CC conforman el tercer objetivo de esta tesis. La investigación llevada a cabo para este objetivo propone el algoritmo 3SHACC. En sus tres etapas, implementa ponderación automática de restricciones, un algoritmo DML con restricciones (capaz de utilizar restricciones ponderadas) y un método de CC jerárquico basado en afinidades que produce la partición de salida. Es evidente que 3SHACC es un modelo CC híbrido, ya que combina técnicas de las dos familias más amplias de métodos CC conocidas: CC particional y DML con restricciones. En cuanto a las conclusiones de este objetivo, destacamos el excelente rendimiento de 3SHACC en comparación con los algoritmos de referencia. 3SHACC es capaz de obtener resultados óptimos con mucha más frecuencia que cualquier alternativa. Además, su modularidad permite a los investigadores seleccionar las partes del mismo que beneficien a su propia investigación.

El cuarto y último objetivo aborda la tarea de combinar múltiples tipos de restricciones bajo el supuesto de que, en problemas reales, se dan dichas combinaciones. El estudio que proponemos para cumplir este objetivo aborda la combinación de restricciones M-L y C-L y restricciones de monotonicidad. Para ello, proponemos un sencillo algoritmo de estilo EM, denominado PCKM-Mono. Este utiliza un término de penalización basado en restricciones para incluirlas en el paradigma de *clustering* monotónico, produciendo así una solución al CC monotónico. La necesidad de este tipo de aproximaciones queda demostrada por la existencia de conjuntos de datos como el SRWU. Este apartado concluye con los resultados experimentales tanto en *benchmarks* estandarizados como en el dataset SRWU. Los resultados muestran la superioridad de nuestra propuesta sobre los métodos de *clustering* clásico, *clustering* puramente restringido y *clustering* puramente monotónico. Estas conclusiones están respaldadas por tests estadísticos bayesianos realizados sobre medidas relativas a la calidad de las particiones, las restricciones incumplidas y el grado de satisfacción de la monotonicidad.

8.2 Publications

This section lists journal, conference and preprint papers published during the PhD study period, ordered by publishing date. The DOI and the number of citations indicated by Google Scholar are given for journal and conference papers.

- **Journal papers**:

  1. González-Almagro, G., Luengo, J., Cano, J. R., & García, S. (2020). DILS: constrained clustering through dual iterative local search. Computers & Operations Research, 121, 104979. DOI: https://doi.org/10.1016/j.cor.2020.104979. CITED BY: 16
  2. González-Almagro, G., Luengo, J., Cano, J. R., & García, S. (2021). Enhancing instance-level constrained clustering through differential evolution. Applied Soft Computing, 108, 107435. DOI: https://doi.org/10.1016/j.asoc.2021.107435. CITED BY: 8
  3. Gonzalez-Almagro, G., Rosales-Perez, A., Luengo, J., Cano, J. R., & Garcia, S. (2021). ME-MEOA/Dcc: Multiobjective constrained clustering through decomposition based memetic elitism. Swarm and Evolutionary Computation, 66, 100939. DOI: https://doi.org/10.1016/j.swevo.2021.100939. CITED BY: 6
  4. Gonzalez-Almagro, G., Suárez, J. L., Luengo, J., Cano, J. R., & García, S. (2022). 3SHACC: Three stages hybrid agglomerative constrained clustering. Neurocomputing, 490, 441-461. DOI: https://doi.org/10.1016/j.neucom.2021.12.018. CITED BY: 1

- **Preprints**:

  1. González-Almagro, G., Peralta, D., De Poorter, E., Cano, J. R., & García, S. (2023). Semi-Supervised Constrained Clustering: An In-Depth Overview, Ranked Taxonomy and Future Research Directions. *arXiv preprint arXiv:2303.00522.* Submitted to ACM Computing Surveys. CITED BY: 0
  2. González-Almagro, G., Suárez, J. L., Sánchez-Bermejo, P., Cano, J. R., & García, S. (2023). Semi-supervised Clustering with Two Types of Background Knowledge: Fusing Pairwise Constraints and Monotonicity Constraints. *arXiv preprint arXiv:2302.14060.* Submitted to Information Fusion. CITED BY: 0

- **Conference papers**:

  1. González-Almagro, G., Rosales-Pérez, A., Luengo, J., Cano, J. R., & García, S. (2020, June). Improving constrained clustering via decomposition-based multiobjective optimization with memetic elitism. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference (pp. 333-341). DOI: https://doi.org/10.1145/3377930.3390187. CITED BY: 5
  2. González-Almagro, G., Suarez, J. L., Luengo, J., Cano, J. R., & García, S. (2020). Agglomerative constrained clustering through similarity and distance recalculation. In Hybrid Artificial Intelligent Systems: 15th International Conference, HAIS 2020, Gijón, Spain, November 11-13, 2020, Proceedings 15 (pp. 424-436). Springer International Publishing. DOI: https://doi.org/10.1007/978-3-030-61705-9_35. CITED BY: 2

3. Wojciechowski, S., González-Almagro, G., García, S., & Woźniak, M. (2022, September). Adapting K-Means Algorithm for Pair-Wise Constrained Clustering of Imbalanced Data Streams. In Hybrid Artificial Intelligent Systems: 17th International Conference, HAIS 2022, Salamanca, Spain, September 5–7, 2022, Proceedings (pp. 153-163). Cham: Springer International Publishing. DOI: `https://doi.org/10.1007/978-3-031-15471-3_14`. CITED BY: 0
4. González-Almagro, G., Bermejo, P. S., Suarez, J. L., Cano, J. R., & García, S. (2022, August). Monotonic Constrained Clustering: A First Approach. In Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence: 35th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2022, Kitakyushu, Japan, July 19–22, 2022, Proceedings (pp. 725-736). Cham: Springer International Publishing. DOI: `https://doi.org/10.1007/978-3-031-08530-7_61`. CITED BY: 2

8.3 Future work

The results of this PhD thesis open up new research lines and contribute to the identification of new challenges in CC. This section presents future work and promising research lines derived from the studies and conclusions gathered in this thesis:

- **Creation of a CC library**: The availability of software is paramount to data science as it directly impacts the efficiency, efficacy, and reproducibility of the research. With the rapid growth of data science, a vast array of software tools have been developed to perform various tasks, ranging from data cleaning and manipulation to ML and statistical analysis. Access to these tools enables researchers to carry out complex data analyses and gain meaningful insights from large and complex datasets. Additionally, the availability of open-source software, such as R and Python libraries, has facilitated collaboration, information sharing and reproducibility of previous work, which is crucial for the advancement of the field. Therefore, the availability of software is an essential factor in enabling the outreach of data science and positive impacts on various fields.

  Unfortunately, very few CC studies make public the software used to produce their results [KWH22]. The creation of an open-access library specialized in CC methods would greatly stimulate research in the area, as it would enable fair and reproducible comparisons for new proposals and grant easy access to working implementations of standard methods for use in new applications.

- **Constraint-based preprocessing**: data preprocessing is a vital part in any data science application [GLH15]. In supervised environments, preprocessing methods involving labels, in addition to the predictors, produce generally better results than completely unsupervised preprocessing methods. However, there is very little work focusing on preprocessing within the SSL paradigm. We argue that constraints, when

available, can prove advantageous in preprocessing procedures, as opposed to the current trend which is to simply dismiss any information that does not fit the mold of classic supervised and unsupervised learning paradigms.

- **Preprocessing the constraint set**: following the same train of thought that motivated the future research line mentioned above, the constraint set can be considered as a dataset itself. This means that it can suffer from the same imperfections as traditional datasets, namely: missing values, noise, redundancies, etc. No methods other than the artificial transitive closure augmentation and our automatic constraint weighting procedures have been proposed to preprocess constraint sets. We argue that general preprocessing methods could increase the quality of the results obtained by CC methods, and therefore their usability and applicability. We consider this to be one of the more promising future research lines concerning CC.

- **Parallelization**: parallelization is highly relevant in genetic algorithms because it can enormously increase the efficiency and speed of the optimization process [DSOM$^+$19, AT$^+$99]. It can drastically reduce the computation time required to find an optimal solution, particularly for scenarios that involve large datasets or complex fitness functions, such as CC. Additionally, parallelization can help overcome issues with local optima, which can trap the optimization process into a suboptimal solution. By exploring multiple regions of the search space simultaneously, parallelization can help the algorithm discover potentially better solutions. Studying the effects of parallelization in the results of genetic algorithms applied to CC remains unaddressed.

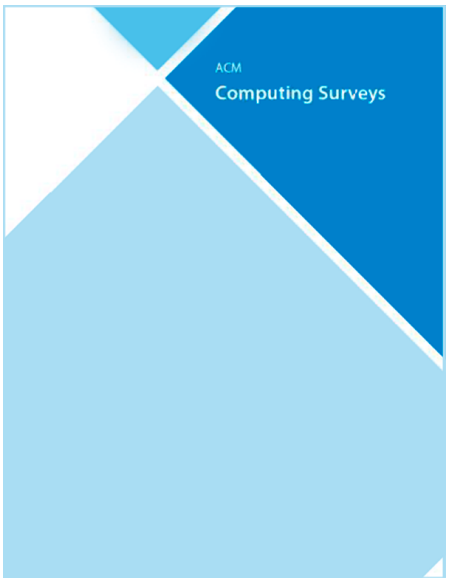
- **New combinations of types of background knowledge**: in the future, more research can be conducted to explore the potential of combining multiple types of background knowledge to develop more robust and accurate models. In this study, we have investigated the combination of pairwise instance-level constraints and monotonicity constraints. However, given the plethora of types of background knowledge that has been found during the elaboration of the taxonomy of CC methods, there is evidence in favor of intensifying research in their combination. One possible direction is to investigate how to integrate multiple types of constraints in a seamless and more efficient way. For example, developing new algorithms or optimization techniques that can handle multiple types of constraints simultaneously can lead to more accurate and robust models. Another potential research direction is to investigate how to identify the best combination of background knowledge for a given problem automatically, thereby keeping human effort and cost low. This can involve developing algorithms or techniques that can identify (1) which types of background knowledge are the most relevant and effective for a given problem, and (2) how to combine them to achieve the best possible outcomes.

Chapter II

Publications

«*HC SVNT DRACONES*».
- Inscription in the Hunt–Lenox Globe.

1 Semi-Supervised Constrained Clustering: An In-Depth Overview, Ranked Taxonomy and Future Research Directions



- **Journal:** ACM Computing Surveys (ACMS)
- **JCR Impact Factor:** 14.324
- **Rank:** 3/110
- **Quartile:** Q1
- **Category:** Computer Science, Theory and Methods
- **Status:** Submitted

Ref.: González-Almagro, G., Peralta, D., De Poorter, E., Cano, J. R., & García, S. (2023). Semi-Supervised Constrained Clustering: An In-Depth Overview, Ranked Taxonomy and Future Research Directions. *arXiv preprint arXiv:2303.00522*.

Semi-Supervised Constrained Clustering: An In-Depth Overview, Ranked Taxonomy and Future Research Directions

Germán González Almagro [*,a,b] **Daniel Peralta** [c] **Eli De Poorter** [c]

José-Ramón Cano [d,b] **Salvador García** [a,b]

[a] *Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada, Granada, Spain*
[b] *Andalusian Institute of Data Science and Computational Intelligence (DaSCI)*
[c] *IDLab, Department of Information Technology, Ghent University - imec, Ghent, Belgium*
[d] *Department of Computer Science, University of Jaén, Jaén, Spain*

Abstract

Clustering has always been a powerful tool in knowledge discovery. Traditionally unsupervised, it has received renewed attention recently as it has shown to produce better results when provided with new types of information, thus leading to a new kind of semi-supervised learning: constrained clustering. This technique is a generalization of traditional clustering that considers additional information encoded by constraints. Constraints can be given in the form of instance-level must-link and cannot-link constraints, which is the focus of this paper. We propose a new metaheuristic algorithm, the Dual Iterative Local Search, and prove its ability to produce quality results for the constrained clustering problem. We compare the results obtained by this proposal to those obtained by the state-of-the-art algorithms on 25 datasets with incremental levels of constraint-based information, supporting our conclusions with the aid of Bayesian statistical tests.

Keywords Constrained Clustering · Instance-level · Must-link · Cannot-link · Dual Iterative Local Search.

* Corresponding Author (germangalmagro@ugr.es)
Email addresses: daniel.peralta@ugent.be (Daniel Peralta), eli.depoorter@ugent.be (Eli De Poorter), jrcano@ujaen.es (José Ramón Cano), salvagl@decsai.ugr.es (Salvador García)

1 Introduction

Two major approaches characterize machine learning: supervised learning and unsupervised learning [1]. In supervised learning, the goal is to build a classifier or regressor that, trained with a set of examples (or instances) X and their corresponding output value Y, can predict the value of unseen inputs. In unsupervised learning, only the set of examples X is available, and no output value is provided. In the latter, the goal is to discover some underlying structure in X. For example, in unsupervised clustering the goal is to infer a mapping from the input to clusters (groups) of similar instances. Generally, the set of examples X is known as the dataset, and the set of output values Y is known as the labels set.

Semi-Supervised Learning (SSL) [2] is the branch of machine learning that tries to combine the benefits of these two approaches. To do so, it makes use of both unlabeled data and labeled data, or other kinds expert knowledge. For example, when considering classification or regression, in addition to a set of labeled data, an additional set of unlabeled data may be available, which can contain valuable information. Similarly, when considering clustering problems, a smaller subset of labeled data (or other types of knowledge about the dataset) may be available. Generally, some kind of information that does not fit within the supervised or unsupervised learning paradigm may be available to perform machine learning tasks. Ignoring or excluding this information does not optimally use all available information, thus the need of SSL [3].

1.1 On the feasibility of semi-supervised learning

With regards to the applicability of SSL, a natural question arises [2]: in comparison with supervised and unsupervised learning, can SSL obtain better results? It could be easily inferred that the answer to this question is "yes", otherwise neither this study nor all the cited before would exist. However, there is an important condition imposed for the answer to be affirmative: the distribution of instances in X must be representative of the true distribution of the data. Formally, the underlying marginal distribution $p(X)$ over the input space must contain information about the posterior distribution $p(Y|X)$. Then, SSL is capable of making use of unlabeled data to obtain information about $p(X)$ and, therefore, about $p(Y|X)$ [3]. Luckily, this condition appears to be fulfilled in most real-world learning problems, as suggested by the wide variety of fields in which SSL is successfully applied. Nonetheless, the way in which $p(X)$ and $p(Y|X)$ are related is not always the same. This gives place to the SSL assumptions, introduced in [2] and formalized in [3]. A brief summary of these assumptions following [3] is presented, please refer to the cited studies for more details.

- **Smoothness assumption**: two instances that are close in the input space should have the same label.
- **Low-density assumption**: decision boundaries should preferably pass through low-density regions in space.

- **Manifold assumption**: in problems in which data can be represented in Euclidean space, instances in the high-dimensional input space are usually gathered along lower-dimensional structures, known as manifolds: locally Euclidean topological spaces.
- **Cluster assumption**: data points which belong to the same cluster also belong to the same class. This assumption can be seen as a generalization of the other three specific assumptions.

As in other machine learning paradigms, the transduction versus induction dichotomy can be made within SSL. Usually, semi-supervised classification methods cope the SSL field, therefore the aforementioned dichotomy is explained in terms of classification as follows:

- **Inductive methods**: inductive methods aim to build a classifier capable of outputting a label for any instance in the input space. Unlabeled data can be used to train the classifier, but the predictions for unseen instances are independent of each other once the training phase is completed. An example of inductive method in supervised learning is linear regression [3].
- **Transductive methods**: transductive methods do not build a classifier for the entire input space. Their predictions are limited to the data used during the training phase. Transductive methods do no have separated training and testing phases. An example of transductive method in unsupervised learning is hierarchical clustering [3].

Classification methods within SSL can be clearly separated following the definitions above. However, when it comes to clustering, this distinction becomes unclear. Clustering methods within the SSL learning paradigm are usually considered to be transductive, as their output is still a set of labels partitioning the dataset and not a classification rule [2]. On the other hand, some authors claim that partitional clustering methods can be considered as inductive methods, because their assignation rule can be used to predict the cluster membership of unseen instances. Hierarchical clustering methods would belong the transductive learning category, as no assignation rule can be derived from them [4]. The differences between partitional and hierarchical clustering will be formalized later in Section 3.

Figure 1 helps us contextualize semi-supervised learning and its derivatives within the overall machine learning landscape. General SSL literature [5, 2, 6] usually divides SSL methods into two categories: semi-supervised classification and semi-supervised clustering. Further dichotomies have been made in later literature. In [3, 7] semi-supervised classification methods are taxonomized taking into account the inductive versus transductive dichotomy. Some of the categories found in these taxonomies have been further studied: [8] proposes a taxonomy for graph-based semi-supervised methods, and [9] does the same for the self-labeling field. Concerning semi-supervised clustering, [10] proposes a high level taxonomy with 4 types of methods, while [11, 12] focus on the specific area of constrained clustering. The supervised and unsupervised learning paradigms are included in Figure 1 for the sake

of contextualization only. Consequently only classic and widely-known tasks belonging to these areas have been included in the diagram.

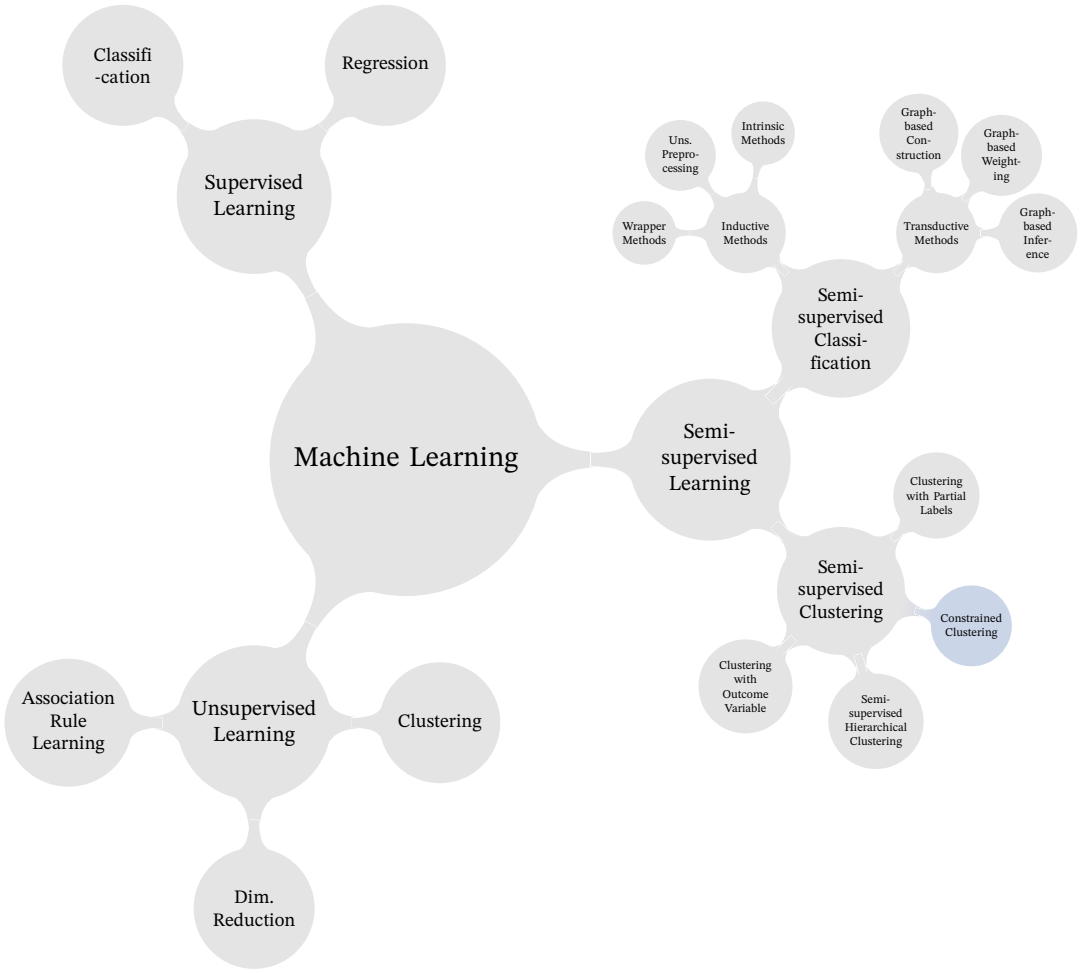


Figure 1: Mindmap of the machine learning overall landscape.

1.2 Related work

The semi-supervised clustering area has been widely studied and successfully applied in many fields since its inception. It can be seen as a generalization of the classic clustering problem which is able to include background knowledge into the clustering process [2]. Many types of background knowledge have been considered in semi-supervised clustering [10], although the most studied one is the instance-level pairwise must-link and cannot-link constraints [12]. This type of background knowledge relates instances indicating if they belong to the same class (must-link) or to different classes (cannot-link). The problem of performing clustering in the presence of this type of background knowledge is referred to in literature as *Constrained Clustering* (CC) (marked in Figure 1 in blue).

This study carries out a comprehensive review of constrained clustering methods. It also proposes an objective scoring system, which addresses the potential and popularity of existing methods, and can be used to produce a sorted ranking for all of them. To the best of our knowledge, no similar study has been published before. Existing literature is either limited to the theoretical background on the topic, very limited in the number of methods reviewed, or outdated due to the rapid advance of the field. The earliest survey including constrained clustering in the reviewed studies can be found in [13], although it is very limited in content. In [11], the first survey focusing specifically on constrained clustering is proposed. It introduces many of the foundational concepts of subsequent studies and provides the first comprehensive reference on the area. However, this study was published in 2007, and even then it was limited to very few methods. The first book fully devoted to constrained clustering was published in [12] (2008). It provides unified formal background within the area and detailed studies on state-of-the-art methods.

1.3 Remainder of this paper

The rest of this study is organized as follows. Section 2 presents a taxonomy of types of background knowledge with which semi-supervised clustering can work, including equivalencies between them in Subsection 2.8. Section 3 formalizes afterwards the constrained clustering problem, starting with basic background on classic clustering (Subsection 3.1) and pairwise constraints (Subsections 3.2 and 3.3), which is followed by a quick note on the history of constrained clustering (Subsection 3.4), and a comprehensive review on the applications of constrained clustering (Subsection 3.5). Subsequently, advanced concepts regarding constrained clustering are introduced in Section 4. A statistical study on the experimental elements used to demonstrate the capabilities of CC methods is proposed in Section 5. This statistical study is used as the basis of the scoring system, which is presented in Section 6 and used in subsequent sections to produce a ranking for all reviewed methods. Section 7 proposes a ranked taxonomic review of constrained clustering methods. A statistical analysis of the taxonomy is presented in Section 8. Finally, Section 9 presents conclusions, criticisms and future research guidelines.

2 Clustering with Background Knowledge

In this section, a comprehensive literature review on the types of background knowledge that have been used by semi-supervised clustering algorithm is carried out. In general terms, 5 families of background knowledge have been identified: partition-level constraints, instance-level constraints, cluster-level constraints, feature-level constraints, and distance constraints. Background information which does not belong to any of the mentioned categories has been placed together in a miscellaneous category. Figure 2 shows a visual representation of this taxonomy. All 5 families are composed by smaller, more specific categories which are detailed below.

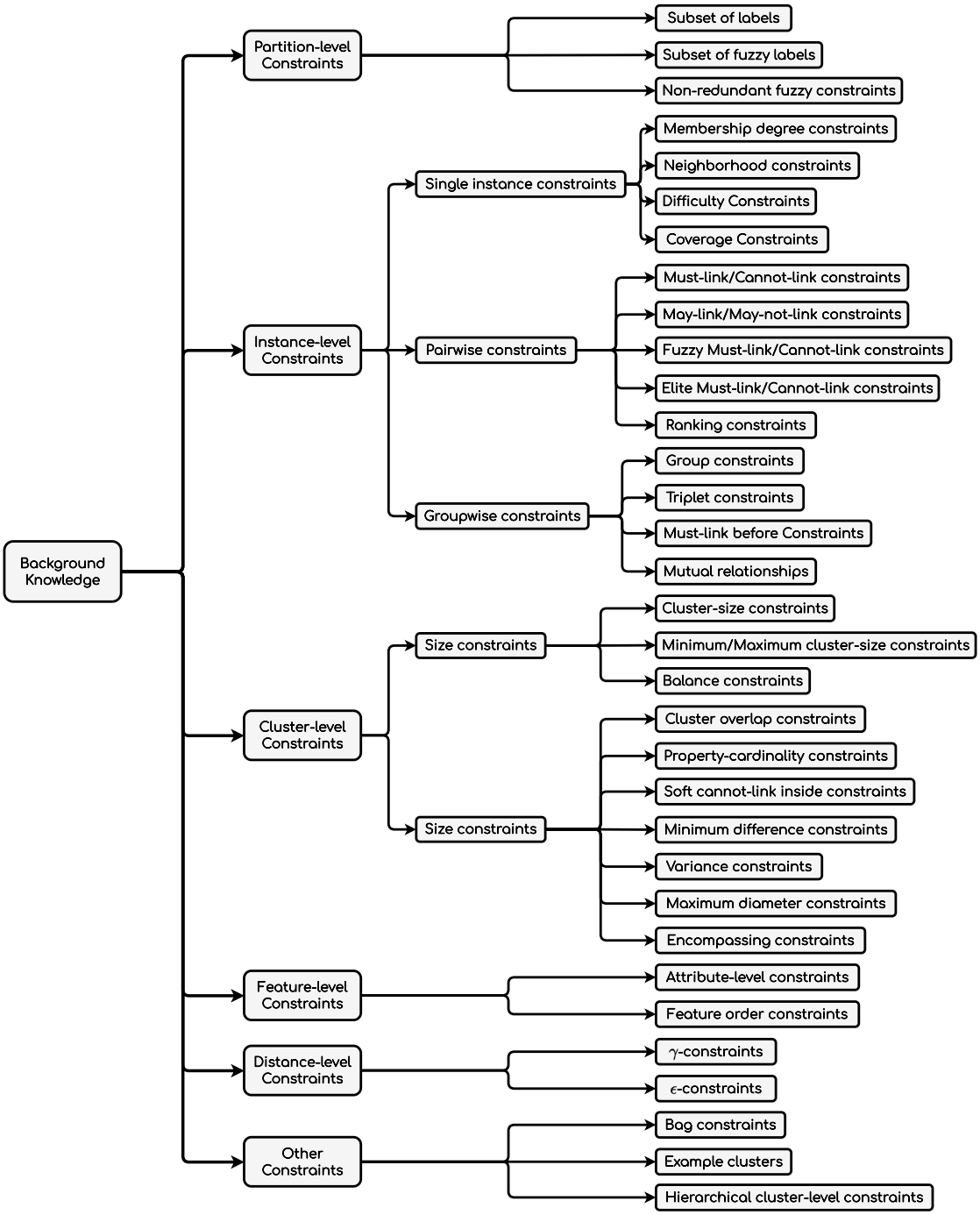


Figure 2: Taxonomy of types of background knowledge

2.1 Partition-level Constraints

Partition-level constraints refer to restrictions imposed on the partition generated by the semi-supervised clustering algorithm [14, 15, 16, 17]. Their most common form is a subset of labeled data, which is often referred simply as "partition-level constraints", although other categories within this type of background knowledge can be found:

- **Subset of labels**: they consist of a subset of instances from the dataset for which labels are available. The resulting partition must be consistent with the given labels [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31].
- **Subset of fuzzy labeled data**: used in fuzzy semi-supervised clustering algorithms. It consists of a subset of instances for which fuzzy labels are provided [32, 33, 34, 35, 36, 37].
- **Non-redundant clusters constraints**: they constraint the output partition so that clusters in it must be orthogonal to each other, therefore maximizing their conditional mutual information and producing non-redundant clusters [38].

2.2 Instance-level Constraints

Instance-level constraints can refer to single instances, pairs of instances or groups of multiple instances. In the case of single instance constraints, they are used to describe particular features of said instances or to restrict the features of the cluster they can belong to:

- **Membership degree constraints**: used in fuzzy semi-supervised clustering algorithm to provide prior membership degrees for some instances [39, 40, 41].
- **Neighborhood constraints**: they link instances to their neighborhood, with the latter being defined differently for every problem [42].
- **Instance difficulty constraints**: they are referred to single instances and specify how hard it is to determine the cluster an instance belongs to, so that the semi-supervised clustering algorithm can focus on easy instances first [43].
- **Coverage constraints**: for clustering algorithms which allow instances to belong to multiple clusters at the same time, this type of constraint limits the number of times an instance can be covered by different clusters [44].

Instance-level pairwise constraints involve pairs of instances and are used to indicate positive or negative relationships. The former refers to features that instances have in common, such as class or relevance, while the latter refers to the opposite case. Even if instance-level must-link and cannot-link are the most common form of instance-level constraints, the latter can be given in multiple ways:

- **Must-link/Cannot-link constraints**: must-links involve pairs of instances that are known to belong to the same class. Therefore they must belong to the same cluster in the output partition. Cannot-link are used to indicate the opposite (the two instances involved in them are known to belong to different classes an thus they need to be placed in different clusters) [11].
- **May-link/May-not-link constraints**: they represent soft must-link and cannot-link constraints respectively. This means that they can be violated in the output partition to some extent. They can be used in combination with the hard must-link and cannot-link constraints [45].
- **Fuzzy Must-link/Cannot-link constraints**: pairwise positive/negative relationships with and associated degree of belief [46].
- **Elite Must-link/Cannot-link constraints**: refined ML and CL constraints. They have the property of being unarguably satisfied in every optimal partition of the dataset [47].
- **Ranking constraints**: in contexts in which output class labels (clusters) can be ordered, ranking constraints are used to indicate whether an instance should be assigned a class label (cluster) higher that the class label of another instance [48].

The last form of instance-level constraints are group constraints, which are used to gather group of instances that are known to share features or to be different to each other in some aspect of their nature. They can also be used to set relative comparisons between a fixed number of constraints. Overall, they can be classified as follows:

- **Group constraints**: also referred to as **grouping information** [49, 50]. They specify the certainty of each or several instances belonging to the same cluster. Note that group constraint cannot be used to specify groups of instances that must not belong to the same cluster [51].
- **Triplet constraints**: also known as **relative constraints** [52, 53, 54]. They involve three instances: an anchor instance a, a positive instance b, and a negative instance c. A triplet constraint indicates that a is more similar to b than c [43, 55].
- **Must-link-before**: these are ML constraints specifically designed to be applied in hierarchical clustering setups. They involve triplets of constraints and their basic idea is to link instances positively not only in the output partition, but also in the hierarchy (dendrogram) produced by hierarchical clustering methods [56].
- **Mutual relationships**: they establish a relation in groups of instances that is not known in advance and is determined during the clustering process. For example, a group of instances in the same mutual relation may be determined to belong to the same cluster during the cluster process, or contrarily they may be determined to not belong to the same cluster. Contrary to ML and CL constraints, mutual relations do

not specify whether the nature of the relation they describe is positive or negative as part of the prior knowledge [57].

2.3 Cluster-level Constraints

Cluster-level constraints are used to restrict a wide variety of features related to clusters without specifying which instances must belong to these clusters. They are considered to be one of the most useful types of background knowledge, as they can convey large amounts of information compared to the amount of expert knowledge available. Size constraints are one of the forms in which cluster-level constraints can be found. They constraint the number of instances that clusters can have in the output partition and can be divided in three categories:

- **Cluster-size constraints**: also called **cardinality constraints** [58]. They specify the number of instances each cluster must have in the output partition. The number of instances in a cluster may vary from a cluster to another [59, 60].
- **Maximum/minimum cluster-size constraints**: they specify the maximum/minimum size a cluster can have in the output partition without specifying the exact size of each cluster [61, 62, 63, 64, 65]. They may also be referred to as **significance constraints** [66].
- **Balance constraints**: also known as **global size constraints** [43] applied in the cluster-level, they try to even the number of instances in every cluster (all cluster should be approximately the same size) [67, 68, 59, 69].

Apart from the size of the cluster, cluster-level constraints can restrict a wide variety of cluster features, ranging from their shape or separation, to the kind of instances they may contain:

- **Cluster-overlap constraints**: they constraint the amount of overlap between clusters [42, 44].
- **Property-cardinality constraints**: they constraint the amount of a specific type of instance a cluster can contain [42].
- **Soft cannot-link inside cluster constraints**: they require that the number of pairs of instances in a cluster which have a cannot-link constraint among them to be bounded [61].
- **Minimum difference constraints**: applied to pair of clusters, they require clusters to be similar or different to some degree [61].
- **Variance constraints**: they impose maximum or minimum values for the variance clusters must feature in the output partition [66].
- **Maximum diameter constraints**: they specify an upper/lower bound on the diameter of the clusters [62].

- **Encompassing constraints**: they determine whether clusters are allowed to encompass each other, i.e., they are allowed to form a hierarchy [44].

2.4 Feature-level Constraints

Feature-level constraints constraint instances by their feature values or directly relate pairs of feature to each other to indicate degrees of importance. Two types of feature-level constraints can be found:

- **Attribute-level constraints**: they constraint the number of possible assignations for instances with specific values for specific features [42].
- **Feature order constraints**: also called **feature order preferences**. They involve pairs of features and determine which one of them is more important. This is, what features need to be paid more attention to when performing comparisons to decide cluster memberships [70].

2.5 Distance Constraints

Distance constraints represent a very particular case of constraint-based information, as they relate pair of instances indirectly and in a global way. That means distance constraints can always be translated to instance-level must-link constraints [11]. Two types of distance constraints are defined in literature:

- γ-**constraints**: also called **minimum margin** [62] or **minimum separation** [71]. They require the distance between two points of different clusters to be superior to a given threshold called γ [11, 62, 72].
- ϵ-**constraints**: they require for each instance to have in its neighborhood of radius ϵ at least another point of the same cluster [11, 62, 72].

2.6 Other Types of Constraints

Finally, authors have proposed forms of background knowledge that do not fit into any of the previous categories:

- **Bag constraints**: specific to the multi-instance multi-label framework, where datasets are given in the form of bags, with each bag containing multiple instances and labels, which provided only at the bag-level. Bag constraints specify similarities between bags [73].
- **Example clusters**: predefined clusters in the dataset given to the clustering algorithm, which is required to output a partition which is consistent with example clusters. This information can be converted to instance-level pairwise constraints [74, 75].

- **Hierarchical cluster-level constraints**: sometimes also referred to as **ranking constraints** [76, 77]. These constraints are designed to be applied only in semi-supervised hierarchical clustering methods. Given pairs of clusters, they specify which action (merge, split, remove, etc.) must be taken over them in successive steps of the clustering process that builds the output dendrogram [78].

2.7 Constraints Usability

After analyzing the wide variety of forms in which constraints can be given, it is reasonable to ask which type of constraint is more effective for general purposes. There is not in fact a unique answer to this question, as it highly depends on the problem or applications and the type of information available to solve it. In [52] an empirical setup that tries to answer this question in a reduced semi-supervised environment is proposed. It only considers instance-level pairwise must-link and cannot-link constraints and subsets of labeled data as available sources of background knowledge. Three questions tried to be answered in the mentioned study, which can be reformulated to include a broader scope as follows:

- Given the same amount of oracle effort, which type of background knowledge is more effective at aiding clustering?
- Which type of constraint is easier to obtain from the oracle?
- Which type of constraint is more reliable?

What it is meant here with an oracle is always understood as the source of background knowledge. This oracle can be a human, an automatic classifier, a crowdsourcing setup to gather information from distributed sources, etc. It is essential for any real-world or in-lab application of semi-supervised clustering to address these three questions.

2.8 Equivalencies Between Types of Background Knowledge

It is well known that some categories of background knowledge are neither isolated nor hermetic. Some types of constraints can be converted to another in a direct manner. Distance constraints can be translated to must-link constraints [11], or a subset of labeled data can always be transformed in a set of must-link and cannot-link constraints [49]. The aim of this section is to provide intuition on all possible transformations without the need of a formal definition/notation for them, as this would require the length of a monography. Previous work on this line has been carried out in [49], although within a much limited scope regarding the types of background knowledge considered. Figure 3 depicts equivalences found between the types of background knowledge introduced in Section 2.

In Figure 3, only conversions without loss of information are considered, e.g.: fuzzy must-link/cannot-link constraints could be converted to must-link/cannot-link constraints by considering only those whose degree of belief is over 50%, although this would involve losing not only constraints, but also the degree of belief information. Even if these kind of transformations are possible, they are not considered here, as they imply losing information.

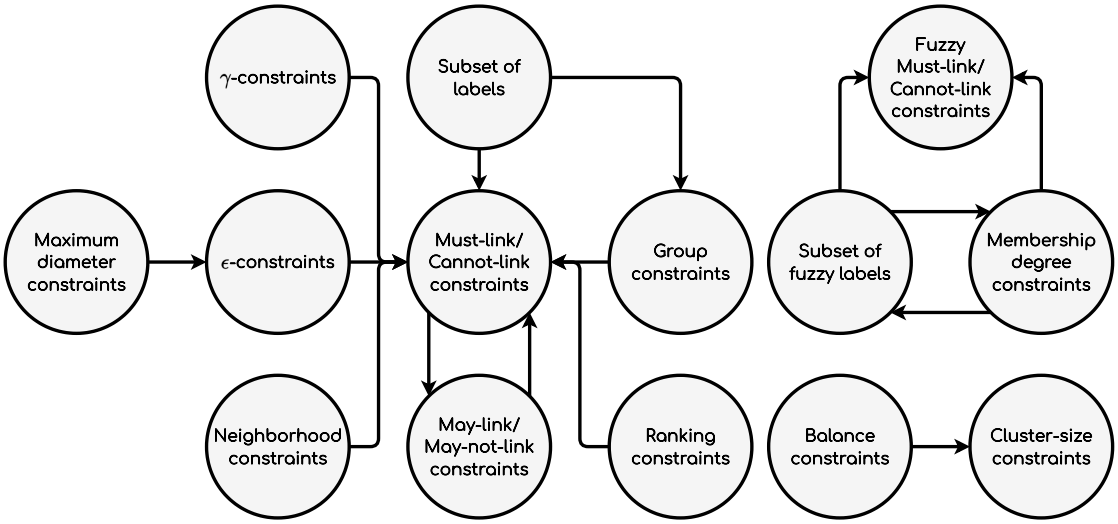


Figure 3: Graphical representation of direct equivalences between types of background knowledge.

3 Instance-Level Pairwise Constrained Clustering

Among all types of background knowledge reviewed in Section 2, pairwise constraints are undoubtedly one of the most studied topics, particularly basic must-link and cannot-link constraints, as it is shown later in this study. From now on, and for the sake of readability, **must-link and cannot-link constraints will be referred to simply as pairwise constraints**. In this section, the basic concepts of classic clustering and semi-supervised partitional and hierarchical clustering under pairwise constraints are introduced. **This problem is known in literature simply as Constrained Clustering (CC)** [11].

3.1 Background on Classic Clustering

Partitional clustering can be defined as the task of grouping the instances of a dataset into K clusters. A dataset X consists of n instances, and each instance is described by u features. More formally, $X = \{x_1, \cdots, x_n\}$, with the ith instance noted as $x_i = (x_{[i,1]}, \cdots, x_{[i,u]})$. A typical clustering algorithm assigns a class label l_i to each instance $x_i \in X$. As a result, we obtain the list of labels $L = [l_1, \cdots, l_n]$, with $l_i \in \{1, \cdots, K\}$, that effectively splits X into K non-overlapping clusters c_i to form a partition called C. The list of labels producing partition C is referred to as L^C. The criterion used to assign an instance to a given cluster is the similarity to the rest of elements in that cluster, and the dissimilarity to the rest of instances of the dataset. This value can be obtained with some kind of distance measurement [79].

Hierarchical clustering methods produce an informative hierarchical structure of clusters called dendrogram. Partitions as described above, with a number of clusters ranging from 1 to n, can always be obtained from a dendrogram by just selecting a level from its hierarchy and partitioning the dataset according to its structure. Typically, agglomerative hierarchical

clustering methods start with a large number of clusters and iteratively merge them according to some affinity criteria until a stopping condition is reached. Every merge produces a new level in the hierarchy of the dendrogram. Formally, given an initial partition with n_c clusters $C = \{c_1, \cdots, c_{n_c}\}$ (usually $n_c = n$), a traditional agglomerative constrained clustering method selects two clusters to merge by applying Equation 1.

$$\{c_i, c_j\} = \underset{c_i, c_j \in C, i \neq j}{\operatorname{argmax}} A(c_i, c_j), \tag{1}$$

with $A(\cdot, \cdot)$ being a function used to determine the affinity between the two clusters given as arguments. This function needs to be carefully chosen for every problem, as it greatly affects the result of the clustering process. Some conventional methods to measure affinity between clusters are worth mentioning, such as single linkage, average linkage and complete linkage [79]. Nevertheless, different measures are used in out-of-lab applications, as the manifold structures usually found in real-world datasets can be hardly captured by the classic affinity measures mentioned above. Typically, classic partitional clustering methods are algorithmically less complex than hierarchical clustering methods, with the former featuring $\mathcal{O}(n)$ complexity and the latter $\mathcal{O}(n^2)$ [11].

3.2 Background on Pairwise Constraints

In most clustering applications, it is common to have some kind of information about the dataset that will be analyzed. In CC this information is given in the form of pairs of instances that must, or must not, be assigned to the same cluster. We can now formalize these two types of constraints:

- Must-link (ML) constraints $C_=(x_i, x_j)$: instances x_i and x_j from X must be placed in the same cluster. The set of ML constraints is referred to as $C_=$.
- Cannot-link (CL) constraints $C_{\neq}(x_i, x_j)$: instances x_i and x_j from X cannot be assigned to the same cluster. The set of CL constraints is referred to as $C_{\neq}$.

The goal of constrained clustering is to find a partition (or clustering) of K clusters $C = \{c_1, \cdots, c_K\}$ of the dataset X that ideally satisfies all constraints in the union of both constraint sets, called $CS = C_= \bigcup C_{\neq}$. As in the original clustering problem, the sum of instances in each cluster c_i is equal to the number of instances in X, which we have defined as $n = |X| = \sum_{i=1}^{K} |c_i|$.

Knowing how a constraint is defined, ML constraints are an example of an equivalence relation; therefore, ML constraints are reflexive, transitive and symmetric. This way, given constraints $C_=(x_a, x_b)$ and $C_=(x_b, x_c)$, then $C_=(x_a, x_c)$ is verified. In addition to this, if $x_a \in c_i$ and $x_b \in c_j$ are related by $C_=(x_a, x_b)$, then $C_=(x_c, x_d)$ is verified for any $x_c \in c_i$ and $x_d \in c_j$ [11].

It can also be proven that CL constraints do not constitute an equivalence relation. However, analogously, given $x_a \in c_i$ and $x_b \in c_j$, and the constraint $C_{\neq}(x_a, x_b)$, then it is also true that $C_{\neq}(x_c, x_d)$ for any $x_c \in c_i$ and $x_d \in c_j$ [11].

Regarding the degree in which constraints need to be met in the output partition/dendrogram of any CC algorithm, a simple dichotomy can be made: hard pairwise constraints must necessarily be satisfied, while soft pairwise constraints can be violated to a variable extent. This distinction is introduced in [11] and adopted by later studies, eventually producing the "may constraints" (may-link/may-not link constraints mentioned in Section 2), which can be seen as the formalization of soft constraints. However, the scientific community still refers to "may constraints" as soft constraints in the majority of the cases and the terms may-link and may-not link are used only in cases in which both soft and hard constraints are mixed and can be considered by the same CC algorithm. The major advantages in favor of soft over hard constraints are found in the resiliency to noise in the constraint set, the flexibility on the design of cost/objective functions, and their optimization procedures. The ability to consider soft, hard, or both types of constraints is a defining element for CC methods.

In [80] two measures designed to characterize the quality of a given constraint set are proposed: **informativeness** (or informativity [11]) is used to determine the amount of information in the constraint set that the CC algorithm could determine on its own, and **coherence**, which measures the amount of agreement between the constraints themselves. These two measures were proposed in early stages of the development of the CC area; however, they have not been used consistently in later studies.

3.3 The Feasibility Problem

Given that CC adds a new element to the clustering problem, we must consider how it affects the complexity of the problem in both of its forms: partitional and hierarchical. Intuitively, the clustering problem goes from its classic formulation "find the best partition for a given dataset" to its constrained form "find the best partition for a given dataset satisfying all constraints in the constraint set". The formalization of this problem is tackled in [81, 11, 82], where the feasibility problems for partitional and hierarchical CC are defined as in 3.1 and 3.2 respectively, where $CS = C_{\neq} \cup C_{=}$ (the joint constraint set). Given these two definitions, we say that **a partition C for a dataset X is feasible when all constraints in CS are satisfied by C**. Note that there exist constraint sets for which a feasible partition can never be found, e.g., no feasible partition exist for $CS_1 = \{C_{=}(x_1, x_2), C_{\neq}(x_1, x_2)\}$ regardless of the value of K. Similarly, the feasibility of partitions such as $CS_2 = \{C_{\neq}(x_1, x_2), C_{\neq}{\neq}(x_2, x_3)C_{\neq}, (x_1x_3)\}$ depends on the value of K. In this case, the feasibility problem for CS_2 can be solved for $K = 3$ but not for $K = 2$.

Definition 3.1 ***Feasibility Problem for Partitional CC**: given a dataset X, a constraint set CS, and the bounds on the number of clusters $k_l \leq K \leq k_u$, is there a partition C of X with K clusters that satisfies all constraints in CS? [81]*

In [81] it is proven that, when $k_l = 1$ and $k_u \geq 3$, the feasibility problem for partitional CC is **NP**-complete, by reducing it from the Graph K-Colorability problem. It is also proven that it is not harder, so both have the same complexity. Table 1 shows the complexity of the feasibility for different types of constraints.

Definition 3.2 ***Feasibility Problem for Hierarchical CC****: given a dataset X, the constraint sets CS, and the symmetric distance measure $D(x_i, x_j) \geq 0$ for each pair of instances: Can X be partitioned into clusters so that all constraints in CS are satisfied? [82]*

Please note that the definition of the feasibility problem for partitional CC (in Definition 3.1) is significantly different from the definition of the feasibility problem for hierarchical CC (in 3.2). Particularly, the formulation for the hierarchical CC does not include any restriction on the number of clusters K, which is equivalent to considering that any level of the dendrogram can be used to produce the partition that satisfies all constraints [82]. In [72] a reduction from the One-in-three 3SAT with positive literals problem (which is **NP**-complete) for the problem in Definition 3.2 is used to prove the complexities presented in Table 1 for the hierarchical CC problem. It is worth mentioning that, for the hierarchical CC problem, the dead-ends problem arises: a hierarchical CC algorithm may find scenarios where no merge/split can be carried out without violating a constraint. Previous solutions based on the transitive closure of the constraint sets have been proposed to this problem, although they imply not generating a full dendrogram [81].

Constraints	Partitional CC	Hierarchical CC	Dead Ends?
ML	**P**	**P**	No
CL	**NP**-complete	**NP**-complete	Yes
ML and CL	**NP**-complete	**NP**-complete	Yes

Table 1: Feasibility problem complexities for partitional and hierarchical CC and dead-ends found in hierarchical CC [81].

Overall, complexity results in Table 1 show that the feasibility problem under CL constraints is intractable, hence constrained clustering is intractable too. This leads to Observation 3.1. For more details on the complexity of constrained clustering please see [81].

Observation 3.1 ***Knowing that a feasible solution exists does not help us find it****. The results from Table 1 imply that the fact that there is a feasible solution for a given set of constraints does not mean it will be easy to find.*

With respect to the dead-ends problem, a full dendrogram considering constraints can be obtained by switching from a hard interpretation of constraints to a soft one. This means that every level in the dendrogram tries to satisfy as many constraints as possible, but constraint violations are allowed in order for the algorithm to never reach a dead-end.

Some interesting results, both positive and negative, about the nature of pairwise constraints are proved and discussed in [11], as well as some workarounds for problems related to the use of constraints in clustering.

3.4 Early History of Constrained Clustering

The Constrained Clustering problem has been rediscovered and renamed throughout years of evolution, firstly in mathematical science, secondly in Computer Science. The first reference to the CC problem was proposed by Harary in [83] as early as in the year 1953. Harary introduced the *signed graph*, which is an undirected graph with +1 or -1 labels on its edges, respectively indicating similarity of dissimilarity between the vertices they connect. This can be directly translated to the ML and CL constraints that shape the CC problem. Besides, Harary introduced the concept of *imbalance* for a 2-way partitioning of such signed graph, which referred to the number of vertices violated by the partitioning. The aim of Harary was to find highly related groups of vertices within a psychological interpretation of the problem: positive edges correspond to pairs of people who like one another, and negative edges to pairs who dislike one another.

It was not until year 2000 that the name Constrained Clustering made its first appearance by the work of K. Wagstaff and C. Cardie in [84], which is a brief paper that introduces later work by the same authors in which the first two CC algorithms in the history of Computer Science are proposed: COP-COBWEB [85] and COP-K-Means [86] in 2000 and 2001 respectively. These two papers set the precedent for a new area in semi-supervised learning known as Constrained Clustering, providing experimental procedures and baselines to compare with.

On the one hand, and following the trend set by K. Wagstaff and C. Cardie, although in separate studies, S. Basu proposes in 2003 the first two soft constrained approaches to CC in [87]: the PCK-Means and MPCK-Means algorithms. Later, in the year 2005, I. Davidson and S.S. Ravi would propose the first hierarchical approaches to the CC problem [72]. On the other hand, E. Xing et al. propose the first distance metric learning based approach to CC with their CSI algorithm [88], also known in literature simply as Xing's algorithm. Finally, in 2008, S. Basu, I. Davidson and K. Wagstaff joined forces to produce the first book fully dedicated to constrained clustering in [12].

Research in CC has followed the general trend in Computer Science ever since. Ranging from well-studied classic clustering approaches, such as fuzzy clustering [89], spectral clustering [90] or non-negative matrix factorization [91], to modern and general optimization models like classic [92] or deep [93] neural networks and evolutive [94] or non-evolutive [95] metaheuristic algorithms.

3.5 Applications of Constrained Clustering

CC has been applied in many fields since its inception. The first applications are gathered in [11], which include clustering of image data, video and sound data, biological data, text

data, web data, and the first application of CC found in [86], which is lane finding for vehicles in GPS data. Figure 4 shows a summary of the overall CC application field.

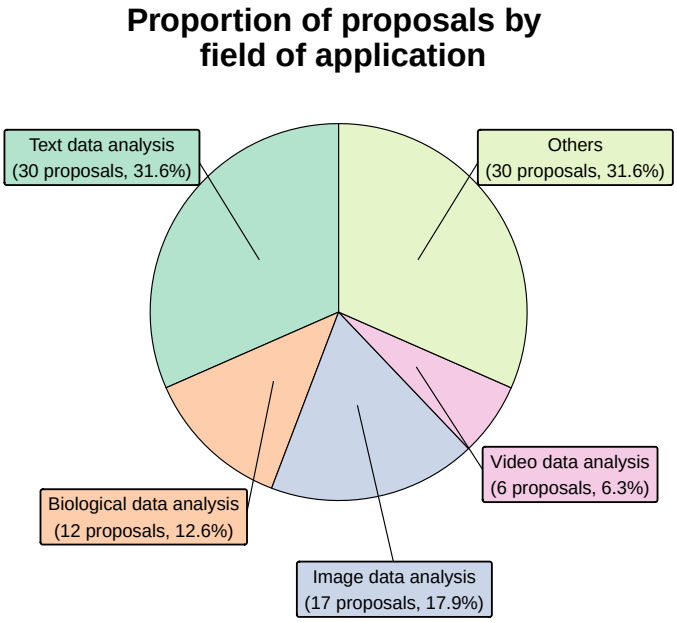


Figure 4: Piechart showing a summary of the overall CC application field.

Table 2 gathers CC applications, sorting them by application field and indicating the specific purpose of every application. The field with the largest number of publications is text data analysis; within it, document clustering has attracted the most publications. Text data clustering is followed by three other wide application fields, which are biological data analysis, image data analysis and video data analysis.

However, some CC applications are very specific and cannot be grouped into wider application fields. Studies which bring forward this kind of applications are listed in Table 3.

4 Constrained Clustering Concepts and Structures

Within the CC research field, some concepts and data structures are repeatedly mentioned and used by researchers. The goal of this section is to provide a formal definition of these concepts, as they will be mentioned later and are necessary for the reader to have a good understanding of the methods described later in Section 7. From now on, and for the sake of readability and ease of writing, **we refer to instances involved in a constraint simply as *constrained instances*, and to ML constraints and CL constraints as simply ML and**

Field of application	No. of studies	Specific application	No. of studies	References
Text data analysis	30	Document clustering	19	[96][97][98][99] [100][101][102][103] [104][105][106][107] [108][109][110][98] [111][112][113]
		Text Clustering	4	[45][114][115][116]
		Verb clustering	1	[117]
		Word disambiguation	1	[118]
		Microblog clustering	3	[119][120][121]
		Document filtering	1	[122]
		Clustering in online forums	1	[123]
Biological data analysis	12	Gene Expression	7	[124][125][126] [127][125][128][129]
		Gene Clustering	1	[130]
		RNA-seq Data Clustering	1	[131]
		Regulatory Module Discovery	1	[132]
		Fiber Segmentation	1	[133]
		Biomolecular Data	1	[134]
Image data analysis	17	Medical image	2	[135][136]
		Image segmentation	3	[137][138][139]
		Image clustering	3	[140][141][142]
		Image categorization	2	[143][144]
		Image annotation	2	[145][146]
		Image indexing	1	[147]
		Point of interest mining	1	[148]
		Multi-target detection	1	[149]
		Face recognition	1	[150]
		Satellite Image Time Series	1	[151]
Video data analysis	6	Extracting moving people	1	[152]
		Web Video Categorization	1	[153]
		Face Clustering	2	[154][155]
		Face Tracking	2	[156][155]

Table 2: Comprehensive listing of applications of CC in wide application fields.

CL, respectively. Instances involved in ML are referred to as ML-constrained instances and instances involved in CL are referred to as CL-constrained instances.

The Constraint Matrix This is one of the most, if not the most, basic and most frequently used data structures to store the information contained in the constraint set. It is a symmetric matrix, with as many rows and columns as instances in the dataset, filled with three values: 0 to indicate no constraint between the instances associated with the row and column in which it is stored, 1 is used for ML and -1 is used for CL. Formally, the Constraint Matrix is a matrix $CM_{n\times n}$ filled as in Equation 2.

Field of Application	References
Identifying speakers in a conversation through audio data	[157]
Clustering of software requirements	[158]
Machinery fault diagnosis	[159]
Patient Segmentation from medical data	[160][161]
Direct marketing applications	[162] [163] [164] [165]
Group extraction from professional social network	[166]
Clustering of cognitive radio sensor networks	[167]
District design	[168][169]
Sentiment analysis	[170][171]
Sketch symbol recognition	[172]
Robot navigation systems	[173]
Terrorist community detection	[174]
Lane finding for vehicles in GPS data	[86]
Optimization of rural ecological endowment industry	[175]
Job-shopping scheduling	[176]
Trace-clustering	[177]
Discovering educational-based life patterns	[178]
Oil price prediction	[179]
Traffic analysis	[180][181]
Vocabulary maintenance policy for CBR systems	[182]
Obstructive sleep apnea analysis	[183]
Internet traffic classification	[184]
Social event detection	[102]

Table 3: Comprehensive listing of particular applications of CC.

$$CM_{[i,j]} = CM_{[j,i]} = \begin{cases} 1 & \text{if } C_{=}(x_i, x_j) \in CS \\ -1 & \text{if } C_{\neq}(x_i, x_j) \in CS \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Please note that, following this definition for the constraint matrix, its diagonal may be assigned to all 1 or all 0. That possibility depends on whether ML with the form $C_{=}(x_i, x_i)$ are included or not in CS, respectively. The inclusion of such constraints may be convenient in some cases. Variants of this matrix are also commonly used. In some cases, the constraint

matrix can store any value in the range $[-1, 1]$, with negative values indicating the weight or degree of belief for CL and positive values doing so for ML.

The Constraint List It is a list, with length equal to the number of constraints, that stores triplets with two values used to specify two instances and a third value used to indicate the type of constraint between them (1 for ML and -1 for CL). Formally, the Constraint List CL contains $|C_=|$ triplets with the form $[i, j, 1]$ for ML such that $C_=(x_i, x_j)$ and $|C_{\neq}|$ triplets with the form $[i, j, -1]$ for CL such that $C_{\neq}(x_i, x_j)$.

The Constraint List is used in methods in which the number of violated constraints needs to be repeatedly computed over fully formed partitions that are not built incrementally. In these cases, the only option is to iterate over the full constraint set and check individually for every constraint whether it is violated by the partition. This task is performed efficiently iterating over CL, which is $\mathcal{O}(|CS|)$, in contrast with CM, which requires an $\mathcal{O}(n^2)$ computation of the number of violated constraints. However, checking for specific constraint violations in iterative partition building processes can be done in $\mathcal{O}(1)$ with CM, as the indexes of the constrained instances are known and matrices support random access. The same task can be performed over CL, but with the much higher computation cost of $\mathcal{O}(|CS|)$.

The Constraint Graph It is a weighted, undirected graph with a one vertex per instance in the dataset and one edge per constraint. An edge connects two instances if they are involved in a constraint, with the weight of the edge indicating the type of constraints, using 1 for ML and -1 for CL. Formally, let an undirected weighted graph $G(V, E, W)$ be a finite set of vertices V, a set of edges E over $V \times V$ and a set of weights W for every edge in E. In the constraint graph $CG(V, E, W)$, V is the set of instances in X, and edges $e(x_i, x_j)$ from E are equivalent to constraints in CS, using the weight of the edge $w_{[i,j]}$ as indicator for the type of constraint, i.e., for edge $e(x_i, x_j)$, if $C_=(x_i, x_j) \in CS$ (ML) then $w_{[i,j]} = 1$, and if $C_{\neq}(x_i, x_j) \in CS$ (CL) then $w_{[i,j]} = -1$ [185].

The Transitive Closure of the constraint set It is an augmented set of constraints which can be obtained on the basis of the information contained in the original constraint set, by applying two of its properties which have been introduced in Section 3.2, and are formally defined here on the basis of the constraint graph as in Properties 4.1 and 4.2. Graphical examples of these two properties are given in Figure 5 and Figure 6, respectively. These two properties can be applied over CG to obtain the transitive closure of the constraint set, which cannot be further augmented without new information.

Property 4.1 ***Transitive inference of ML****: Let cc_1 and cc_2 be two connected components in CG with only positive edges in it (only ML constraints). Then, if there is a constraint $C_=(x_i, x_j)$ with $x_i \in cc_1$ and $x_j \in cc_2$, then the new constraints $C_=(a, b)$ can be inferred for all $a \in cc_1$ and $b \in cc_2$ [12].*

Property 4.2 ***Transitive inference of CL****: Let cc_1 and cc_2 be two connected components in CG with only positive edges in it (only ML constraints). Then, if there is a constraint $C_{\neq}(x_i, x_j)$ with $x_i \in cc_1$ and $x_j \in cc_2$, then the new constraints $C_{\neq}(a, b)$ can be inferred for all $a \in cc_1$ and $b \in cc_2$ [12].*

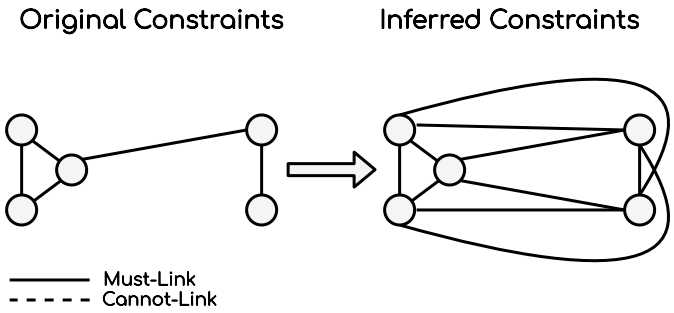


Figure 5: Example of transitive inference of ML constraints.

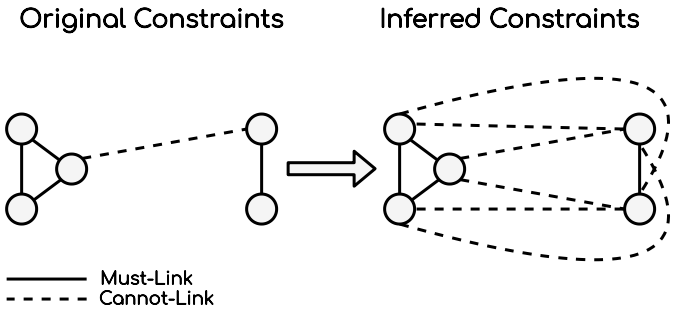


Figure 6: Example of transitive inference of CL constraints.

The Chunklet Graph This graph structure can be derived from the definition of the constraint graph and the concept of chunklet. Chunklets are defined in [186, 187] as "subsets of points that are known to belong to the same, although unknown, class". With this definition, it is clear that ML connected components can be compared to chunklets, thus the chunklet graph can be obtained on the basis of the constraint graph. This is done by replacing ML connected components in CG by a single vertex which is adjacent to all former neighbors of the connected components [188]. In the case that vertices in CG also store position-related information (as in CC), the position of the new vertex is computed as the average of the nodes in the connected component [185].

The Cluster Skeleton It is a reduced constraint set which defines the true basic clustering structure of the data. It is obtained by applying the farthest-first scheme to query an oracle about the constraint relating selected instances from the dataset. This is done within an iterative scheme in which membership neighborhoods are created and the farthest instance from all of them is always selected to be queried against at least one instance from every existing neighborhood. If it is constrained to any of those instances by ML, then it is added to that neighborhood and a new ML is created, whereas if it is constrained by CL to all of them, then a new neighborhood is created and related to the other one by CL. The goal of this procedure is to build a constraint set which defines as many disjoint clusters as possible, aiding later CC algorithms determine the number of clusters a feasible partition must have [189].

The Infeasibility The concept of infeasibility refers to the number of constraints violated by a given partition. It is one of the most used concepts in CC, as many objective/fitness functions include penalty terms that are directly proportional to the number of violated constraints. Given a partition C (an its associated list of labels L^C) and a constraint set CS, the infeasibility can be defined as in Equation 3, with $\mathbb{1}[\![\cdot]\!]$ being the indicator function (returns 1 if the input is true and 0 otherwise) [190].

$$\text{Infs}(C, CS) = \sum_{C_=(x_i,x_j)\in CS} \mathbb{1}[\![l_i^C \neq l_j^C]\!] + \sum_{C_\neq(x_i,x_j)\in CS} \mathbb{1}[\![l_i^C = l_j^C]\!] \tag{3}$$

The k-NN Graph Also called k-NNG. It is not an exclusive concept from CC. It has been widely used in classic clustering literature and k-NN based classification. However, it is a very useful tool for CC research, therefore many CC approaches are built based on its definition. The k-NNG is a weighted undirected graph in which vertices represent instances from the dataset and every vertex is adjacent to at most k vertices. An edge is created between vertices u and v if and only if instances associated to u and v have each other in their k-nearest neighbors set. The weight $w(v, u)$ for the edge connecting u and v is defined as the number of common neighbors shared by u and v: $w(v, u) = |NN(u) \cup NN(v)|$, with $NN(\cdot)$ denoting the set of neighbors of the vertex given as argument [191].

5 Statistical Analysis of Experimental Elements

In this section, a general view on how CC methods are evaluated and compared is presented. Most studies in CC present one or various new methods that need to be evaluated and proved to be competitive with respect to the state-of-the-art at the time they were proposed. In this section, the three experimental elements used to do so are analyzed: the datasets, the validity indices, and the competing methods. Table 4 introduces the 15 most frequently used instances of these elements among all the papers analyzed in this study. All statistics presented in this section have been obtained by analyzing 270 studies, which propose a total of 307 methods. Some studies propose more than one method, and some methods are proposed in more than one study, hence the discordance between the number of papers analyzed and the number of proposed methods. Special cases of the experimental elements have not been taken into consideration to obtain the statistics presented in this section. In other words, if a paper uses the Iris dataset for its experiments but removes one of the three classes in the dataset, it is then considered as a single use of the classic Iris dataset, and not listed as a separate dataset. The same can be said for the other two experimental elements. For example, uses of the Pairwise F-measure (PF-measure) are included in the count of the F-measure, and variations on the initialization methods of COP-K-Means are included in the count of the basic COP-K-Means. This is done to obtain more representative and general statistics.

Sections 5.1, 5.2 and 5.3 dive into the statistics of the frequently used experimental setups regarding datasets, validity indices and competing methods, respectively. Section 5.4 presents the most used procedure to artificially generate constraints for benchmarking purposes, and Section 5.5 gives a quick note on the use of statistical testing to support conclusions in CC literature.

Datasets		Competing Methods		Validity Indices	
Name	No. of Uses	Name	No. of Uses	Name	No. of Uses
Iris	134	COP-K-Means	64	NMI	89
Wine	105	K-Means	57	CE	60
Ionosphere	72	MPCK-Means	34	RI	55
Synthetic	69	SSKK	26	ARI	48
Glass	58	PCK-Means	22	F-measure	38
Breast	51	KKM	21	Time	25
Soybean	47	FFQS	17	Purity	11
Balance	43	Random	14	Unsat	11
Sonar	41	RCA	14	Non Standard (NS)	10
Heart	39	E^2CP	13	JC	8
Digits	35	NCuts	13	Visual	7
Ecoli	28	CSI	13	V-measure	4
MNIST	28	CCSR	12	Precision	4
Protein	28	Constrained EM	12	FMI	3
20Newsgroup	27	HMRF-K-Means	11	CRI	3

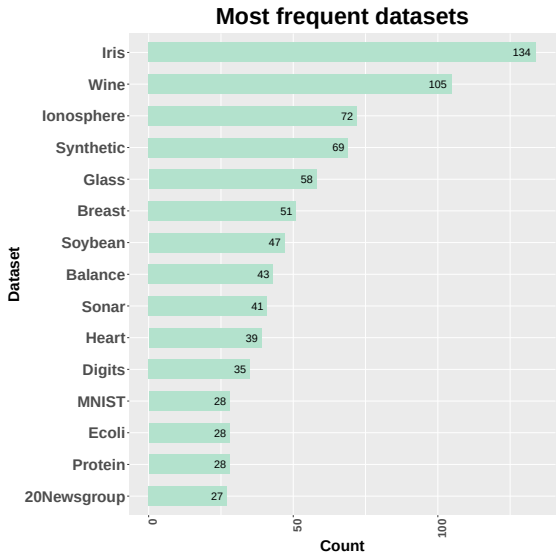
Table 4: Most frequently used dataset in CC experimental setups.

5.1 Analysis of Datasets

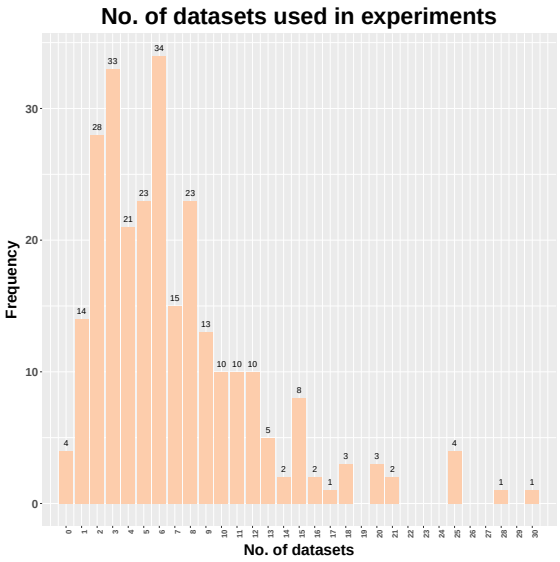
A total of 389 different datasets can be identified in the experimental sections of the literature in CC. Figure 7 displays three different statistical measures about the use of these datasets. Figure 7a depicts the same information contained in Table 4, presenting it visually for the sake of ease of understanding. Figure 7b gives a histogram of the number of datasets used in experiments. Lastly, Figure 7c introduces boxplots featuring the variability on the number of datasets used in different years.

From 7a, it is clear that classification datasets are used as benchmarks for CC methods. The reason for this lies in the lack of specific benchmarks, as very few have been proposed since the inception of the research topic. Classic classification datasets have to be used in order to generate the constraint sets needed by CC methods (see Section 5.4). In these cases, labels are never provided to the CC method, but used as the oracle to generate the constraint sets.

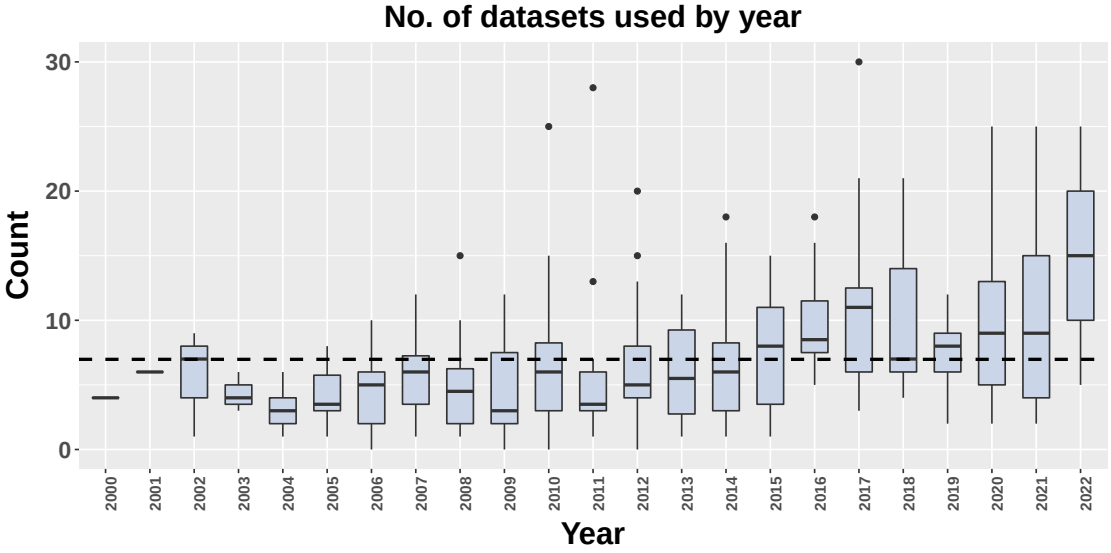
Looking at Figure 7b, it can be concluded that the most frequent number of datasets used in experiment is 6, and the study which uses the most datasets, analyzes up to 30 of them. Most papers use between 1 and 9 datasets. Note how some papers do not use any datasets, therefore they don't carry out any experiments to prove the efficacy of their proposal. Figure 7c shows a consistent increase over the years in the number of datasets used in experiments, probably due to the general growth in computing power, and to the increasing availability of datasets. It also shows how, except for the first few years, there is no consensus on the number of datasets to be used to demonstrate the capabilities of a new method, as boxplots show high variability within each year.



(a) No. of times each dataset in Table 4 is used in experiments



(b) Distribution of the number of datasets used in experiments



(c) Variability of the number of datasets used in every year (the dashed line represents the overall average)

Figure 7: Statistics about the datasets used in the experimental setups of all papers reviewed.

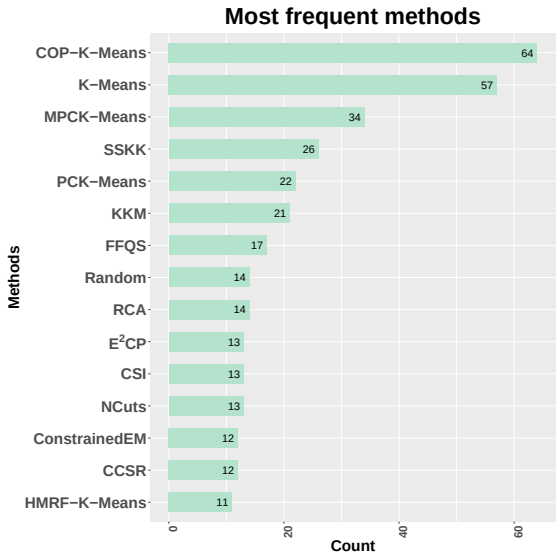
5.2 Analysis of Competing Methods

Any new CC method has to be proven to be competitive with the methods belonging to the state-of-the-art. Methods belonging to this category change over the years. Nonetheless, Figure 8a presents a set of methods which are used very frequently, and can be subsequently understood as baseline methods. In fact, most of them correspond to the first proposals in different CC categories, e.g.: COP-K-Means is the first CC method ever proposed, PCK-Means is the first penalty-based CC method, KKM is the first constrained spectral clustering method, FFQS is the first active constrained clustering method, etc. Section 7 presents all these methods within the context of their specific CC category. Algorithms such as K-means or NCuts stand out as well, as they are not CC algorithms but classic clustering algorithms. When the experiments carried out aim to prove not only the capabilities of a new CC method, but also the viability of CC itself (as in the first proposals) or the viability of any new constraint generation method, then comparing with classic clustering algorithm is justified.

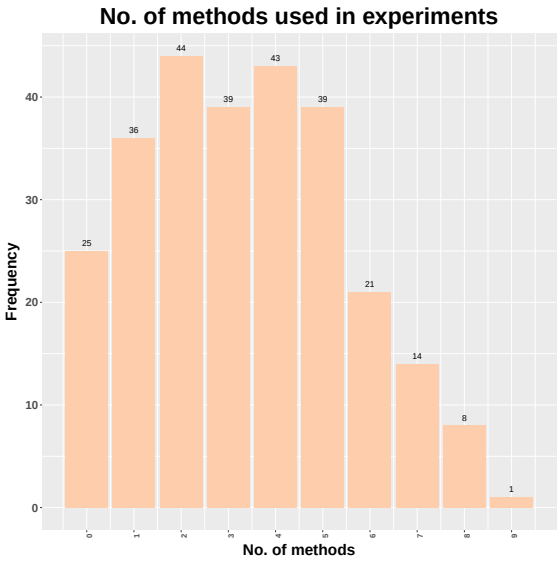
Figure 8b shows the distribution of the amount of methods used in experimental setups in CC literature. The most frequent comparison uses only two methods, which is a very low number taking into account the plethora of methods available to compare with (307 particularly). However, comparisons using between 4 and 6 methods are also reasonably frequent, with said frequency decaying from 6 methods to 9, which are used only in a single study. There is a particular fact that may catch our attention: 25 studies chose not to compare against any previous proposals. Given how well established baselines methods are, this should never be allowed in new CC studies. An increasing tendency can be observed in the number of methods used over the years. Likewise with the number of datasets (see Figure 7c). Accordingly, this can be caused by an increase in computing power over the year and by the increase in the number of available methods to compare against.

Piecharts in Figure 9 show further statistics about the proportions of methods used in experiments in CC studies. For example, it may be interesting to answer the following question: from all methods used in experiments to compare with, how many of them are CC methods? Figure 9b answers this question. From all methods compared with (386), only 38.8% (147) are CC methods. The rest of the methods are not necessarily classic clustering methods, they can belong to other fields of SSL or use different types of constraints. This may seem contradictory with respect to what Figure 7a shows. However, this is not the case. In conjunction, Figures 9 and 7a evidence that the most frequently used methods are CC methods, even if the number of different classic clustering methods used to compare against is higher than the number of different CC methods.

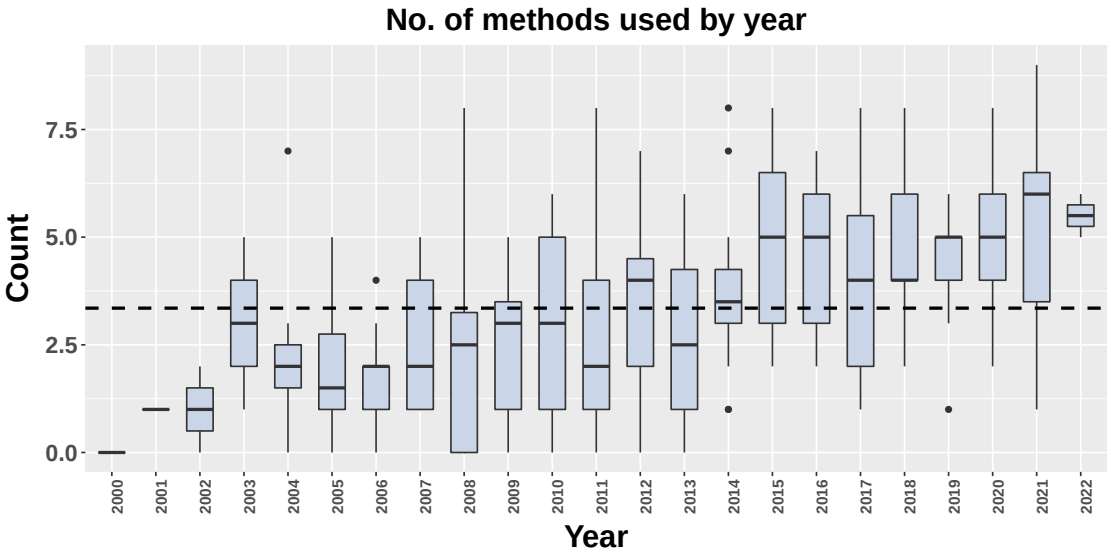
Another interesting question is: from all CC methods proposed over the years, how many of them are used to compare with in later studies? Figure 9a provides now the answer to this question. From all CC methods proposed (307) in the reviewed studies, 48.5% of them (149) are used in the experimental section of other studies. This indicates that more than half of the proposed methods have never been considered to be compared with by other authors. Of course, this statistic does not take into account the number of years any given method has been available to be used, only the absolute number of uses. However, this should not have a great impact in the proportions.



(a) No. of times each method in Table 4 is used in experiments

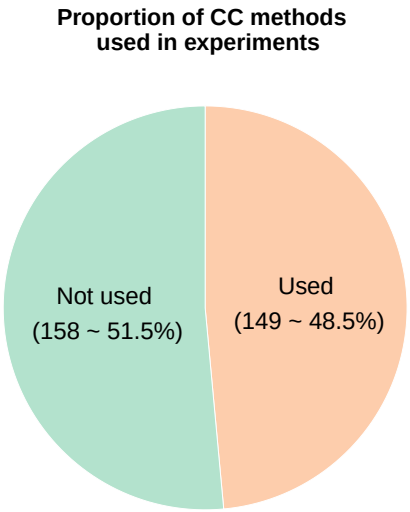


(b) Distribution of the number of methods used in experiments

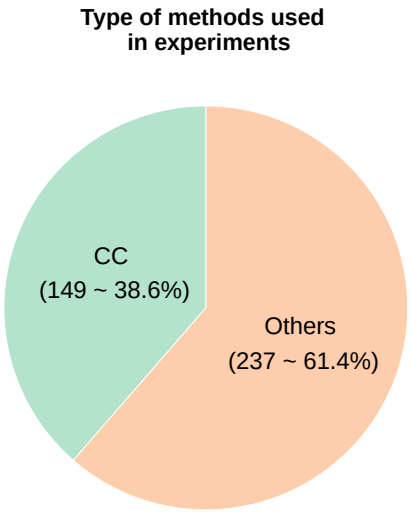


(c) Variability of the number of methods used in every year (the dashed line represents the overall average)

Figure 8: Statistics about the methods used in the experimental setups of all papers reviewed.



(a) Proportion of CC methods that are used in later studies to compare with.



(b) Proportion of methods used in experiments which are CC or other type of clustering methods.

Figure 9: Piecharts depicting the usability of all CC methods reviewed in experimental setups.

5.3 Analysis of Validity Indices

Validity indices are used to objectively evaluate the performance of a given method independently of the benchmarks it is tested in. This means that the output value of the validity indices is independent from the features of the benchmarks datasets, such as their size of their number of features in the case of classification datasets. The same analysis performed over the datasets (Section 5.1) is performed over the validity indices. Figure 10 shows the statistical summary on the usage of validity indices in CC literature (as it was performed for the datasets). From Figure 10a it is clear that the most used validity index is the Normalized Mutual Information (NMI), followed by the Clustering Error (EC), the Rand Index (RI), the Adjusted Rand Index (ARI) and the F-measure. Time is used to compare methods a total of 25 times, which represents a very low percentage over the total number of comparisons. Note how Visual validation makes it to the top 15, despite not being an objective and reliable comparison method. Non Standard (NS) measures are used in 10 studies, meaning that the used measure is proposed specifically in the same paper for that specific case or that it is never referred to again in CC literature. Among the 15 most used measures, there is only one specifically designed to compare CC methods: the Unsat. Unsat measures the proportion of constraints violated by the output partition of any given method, and therefore can be used to measure scalability with respect to the number of constraints. In this study, authors want to draw two validity indices to the attention of the reader: the Constrained Rand Index (CRI), proposed in [85], and the Constrained F-measure (CF-measure), proposed in [192]. These two validity indices are versions of RI and F-measure, respectively, corrected by the

number of constraints available. They assume that the higher the number of constraints available, the easier it is to score a high value in classic clustering validity indices, therefore they correct (lower) those values with the size of the constraint set. These two measures are used in very few cases, while they are specifically designed to benchmark CC methods. This fact is particularly remarkable in the case of the CRI, as it was proposed along with the first CC study ever in [85].

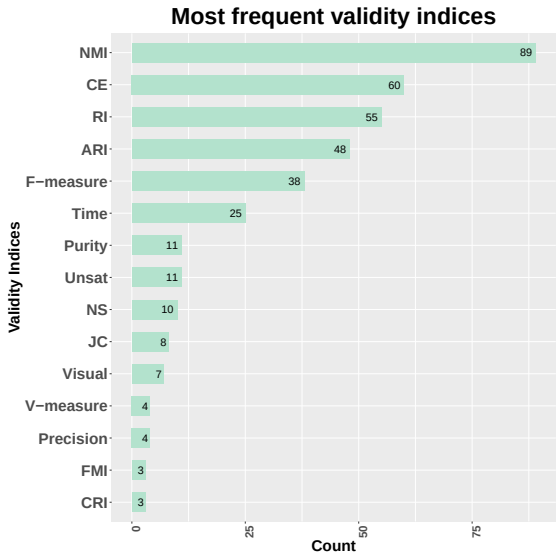
Figure 10b shows that the most common number of validity indices used in CC literature is 1. Using more than one validity index is a healthy practice in any study, as demonstrating the capabilities of a new method in more than one dimension reinforces positive conclusions about it. With respect to Figure 10c, the variability observed in the other cases (Figures 7c and 8c) is not present here, as the number of validity indices used is not related to the computation power, and most of them were proposed before the inception of CC.

Some of the validity indices in Table 4 do not need to be specifically defined, as it is the case of the Time or the Visual indices, which are self-explanatory. Others are incidentally defined, such as the Precision, which is a by-product of the F-measure. Lastly, no general definition can be given for the Non Standard indices. For the rest of them, both a formal and intuitive definition can be found here. From now on, in this section, C refers to the partition generated by any given CC method, and C^* refers to the ground-truth partition. Please note the influence of the use of classification datasets in the selection of validity indices used to evaluate CC methods. All of the validity indices take two partitions as their input, and produce a measure according to their similarity or dissimilarity. Therefore, these validity indices can be used to evaluate the performance of a clustering algorithm only when one of the partitions given as input is the ground-truth partition, which can be obtained for labeled datasets only.

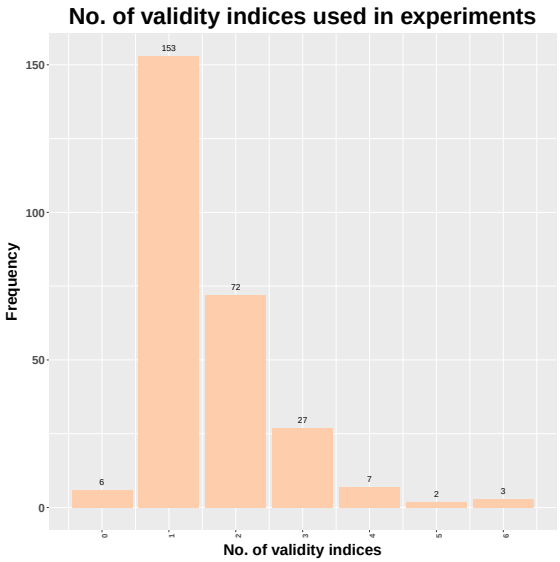
Normalized Mutual Information (NMI) The NMI is an external validity index that estimates the quality of a partition with respect to a given underlying labeling of the data. In other words, NMI measures how closely a clustering algorithm could reconstruct the underlying label distribution. Taking C as the random variable denoting the cluster assignments of instances (the partition), and C^* as the random variable denoting the underlying class labels, the NMI can be formulated in terms of information theory as in Equation 4 [193, 194, 189].

$$\mathrm{NMI} = 2\frac{I(C;C^*)}{H(C)+H(C^*)}, \tag{4}$$

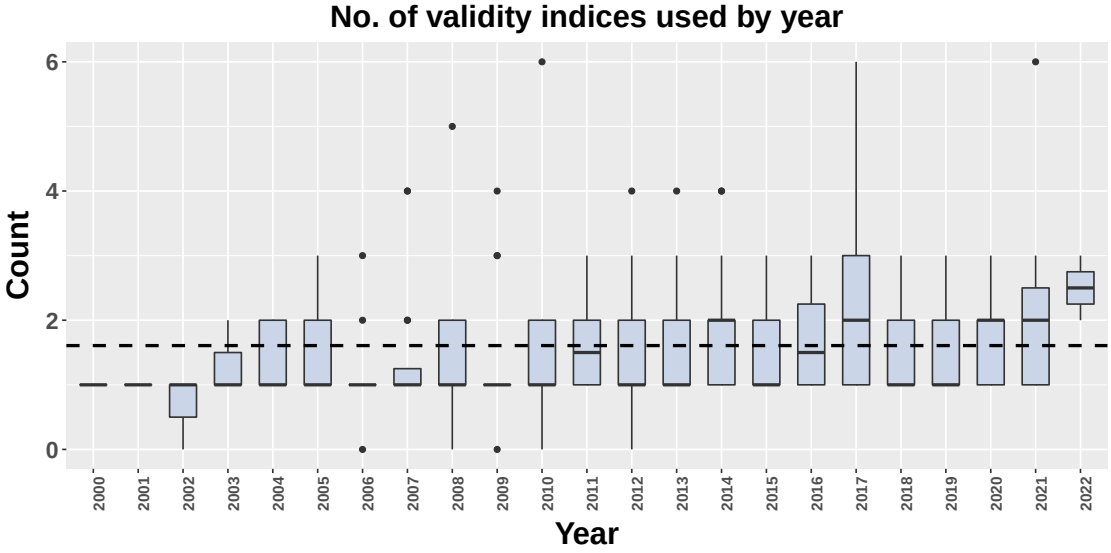
where $I(X;Y) = H(X) - H(X|Y)$ is the mutual information between the random variables X and Y, $H(X)$ is the Shannon entropy of X and $H(X|Y)$ is the conditional entropy of X given Y. For more details on NMI please see [195]. The output value range for the NMI is $[0,1]$, with high values indicating a high level of similarity between the two partitions, and a low value indicating a low level of similarity.



(a) No. of times each validity index in Table 4 is used in experiments



(b) Distribution of the number of validity indices used in experiments



(c) Variability of the number of validity indices used in every year (the dashed line represents the overall average)

Figure 10: Statistics about the validity indices used in the experimental setups of all papers reviewed.

The Clustering Error (CE) The CE is the negative, unsupervised version of the classic classification accuracy. It measures the proportion of correctly clustered instances by best matching the cluster labels to the ground-truth labels. Given the permutation mapping function map(·) over the cluster labels, the CE with respect to map(·) can be computed as in Equation 5 [196, 197, 198]. The best mapping function that permutes clustering labels to match the ground truth labels can be computed by the Kuhn-Munkres algorithm (the Hungarian method) [196, 199]. Please note that the CE validity index is sometimes used in its positive form, which is the clustering accuracy. It can be computed by just changing the condition in the indicator function $\mathbb{1}[\![\cdot]\!]$ to be negative (replace $=$ by $\neq$). The output value range for the CE is $[0, 1]$, with high values indicating a low level of accuracy, and a low value indicating a high level of accuracy.

$$\mathrm{CE} = 1 - \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\![\mathrm{map}(l_i^C) = l_i^{C^*}]\!] \tag{5}$$

The Rand Index (RI) The RI measures the degree of agreement between two partitions. It can be used to measure the quality of a partition obtained by any CC algorithm by giving the ground-truth partition as one of them. Therefore, the two compared partitions are C and C^*. The RI views C and C^* as collections of $n(n-1)/2$ pairwise decisions. For each x_i and x_j in X, they are assigned to the same cluster or to different clusters by a partition. The number of pairings where x_i is in the same cluster as x_j in both C and C^* is taken as a; conversely, b represents the number of pairings where x_i and x_j are in different clusters. The degree of similarity between C and C^* is computed as in Equation 6 [200], where n is the number of instances in X. The output value range for the RI is $[0, 1]$, with high values indicating a high level of agreement between the two partitions, and a low value indicating a low level of agreement.

$$\mathrm{RI} = \frac{a+b}{n(n-1)/2} \tag{6}$$

The RI can be conveniently formulated in terms of the elements of a confusion matrix as well [201]. Equation 7 defines these elements in terms of cluster memberships in a partition, which can be referred to as: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). Equation 8 makes use of these elements to give a new definition for the RI.

$$\begin{aligned}
\mathrm{TP} &= \{(x_i, x_j) | l_i^{C^*} = l_j^{C^*},\ l_i^C = l_j^C,\ i \neq j\} \\
\mathrm{FP} &= \{(x_i, x_j) | l_i^{C^*} = l_j^{C^*},\ l_i^C \neq l_j^C,\ i \neq j\} \\
\mathrm{TN} &= \{(x_i, x_j) | l_i^{C^*} \neq l_j^{C^*},\ l_i^C \neq l_j^C,\ i \neq j\} \\
\mathrm{FN} &= \{(x_i, x_j) | l_i^{C^*} \neq l_j^{C^*},\ l_i^C = l_j^C,\ i \neq j\}
\end{aligned} . \tag{7}$$

$$\text{RI} = \frac{|\text{TP}| + |\text{TN}|}{|\text{TP}| + |\text{FP}| + |\text{TN}| + |\text{FN}|} \tag{8}$$

The Adjusted Rand Index (ARI) The ARI is the corrected-for-chance version of the RI. This correction is done by taking into account the expected similarity of all comparisons between partitions specified by the random model to establish a baseline. This modifies the output value range of the original RI, transforming it into $[-1, 1]$ and slightly changing its interpretation. In ARI, a high output value still means a high level of agreement between the two partitions, and a low value means a low level of agreement. However, a value lower than 0 means that the results obtained are worse than those expected from the average random model. Equation 9 gives the formalization for the ARI [202].

$$\text{ARI} = \frac{\text{RI} - \text{Expected Index}}{\text{Maximum Index} - \text{Expected Index}}, \tag{9}$$

where Expected Index is the degree of similarity with a random model, Maximum Index is assumed to be 1, and RI is the RI value computed for partitions C and C^*.

Pairwise F-measure (PF-measure) The PF-measure is defined as the harmonic mean of pairwise precision and recall, which are classic validity indices adapted to evaluate pairs of instances. For every pair of instances, the decision to cluster this pair into the same or different clusters is considered to be correct if it matches with the underlying class labeling. In other words, the PF-measure gives the matching degree between the obtained partition C and the ground-truth class labels C^*. It can be formalized as in Equation 11 [189, 193], where Precision and Recall are defined as in Equation 10, following the notation introduced in Equation 7 [201]. For more details on the PF-measure please see [203]. The output value range for the PF-measure is $[0, 1]$, with high values indicating a high level of agreement between the two partitions, and a low value indicating a low level of agreement.

$$\text{Precision} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|}, \quad \text{Recall} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|}. \tag{10}$$

$$\text{PF-measure} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2|\text{TP}|}{2|\text{TP}| + |\text{FP}| + |\text{FN}|}. \tag{11}$$

The Constrained Pairwise F-measure (CPF-measure) The CPF-measure is a version of the classic PF-measures that takes constraints into account. It does so by including the number of ML constraints in the computation of the Precision and Recall terms as in Equation 12. This way, the number of correctly clustered instances is penalized by the number of ML constraints. Subsequently, the higher the number of ML constraints available to perform

clustering, the less credit the term TP is given. The final CPF-measure can be computed as in Equation 13. The output value range for the CPF-measure is $[0, 1]$, and the value is interpreted as in the PF-measure.

$$\text{Precision}' = \frac{|\text{TP}| - |C_{=}|}{|\text{TP}| + |\text{FP}| - |C_{=}|}, \quad \text{Recall}' = \frac{|\text{TP}| - |C_{=}|}{|\text{TP}| + |\text{FN}| - |C_{=}|}. \tag{12}$$

$$\text{CPF-measure} = 2\frac{\text{Precision}' \times \text{Recall}'}{\text{Precision}' + \text{Recall}'}. \tag{13}$$

The Purity This is a classic validity index used to evaluate the performance of clustering methods. It measures the homogeneity of the generated partition, i.e.: the extent to which clusters contain a single class [204, 205, 206]. It can be computed by determining the most common class of each cluster c_i (with respect to the true labels C^*), which can be done my determining the greatest intersection with respect to the ground-truth partition. The sum of all intersection is then divided by the total number of instances n in the partition C to obtain the Purity value of said partition. Equation 14 formalizes this concept. The output value range for the Purity is $[0, 1]$, with high values indicating high level of resemblance between the two partitions, and a low value indicating a low level of resemblance.

$$\text{Purity} = \frac{1}{n} \sum_{c_i \in C} \max_{c_i^* \in C^*} |c_i \cap c_i^*| \tag{14}$$

The Unsat The Unsat measures the ability of any given CC method to produce partitions satisfying as many constraints as possible. It is computed as the ratio of satisfied constraints as in Equation 15 [81, 95]. It produces a value in the range $[0, 1]$, with a high value indicating a high number of violated constraint, and a low value indicating the contrary.

$$\text{Unsat} = \frac{\text{Infs}(C, CS)}{|CS|}. \tag{15}$$

The Jaccard Index (JC) The JC measures similarity between finite sample sets. It is defined as the size of the intersection divided by the size of the union of the sample sets. However, this definition is inconvenient when JC is applied to measure the quality of a partition. Subsequently, a more useful definition can be given in terms of Equation 7 as in Equation 16 [207, 208, 209]. Please note that a high value of the CJ in the range $[0, 1]$ indicates high dissimilarity between the two compared partitions, while a low value indicates high similarity.

$$\text{JC} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}| + |\text{FN}|} \tag{16}$$

The V-measure . This measure is closely related to the NMI, as it can be viewed as a version of it that computes the normalization of the denominator in Equation 4 with an arithmetic mean instead of a geometric mean. The V-measure is defined as the harmonic mean of Homogeneity and Completeness, which evaluate a partition in a complementary way [210, 211]. Homogeneity measures the degree to which each cluster contains instances from a single class of C^*. This value can be computed as in Equation 17, where $H(X|Y)$ is the conditional entropy of the class distribution of partition X with respect to partition Y, and $H(X)$ is the Shannon entropy of X. Following the same notation, the Completeness can be defined as in Equation 18. This can be intuitively interpreted as the degree to which each class is contained in a single cluster. Subsequently, the V-measure is computed as in 19 [210]. Please note that another aspect to which the V-measure and the NMI are closely related is that the mutual information between two random variables $I(X;Y)$ can always be expressed in terms of the conditional distribution of said variables $H(X|Y)$ as follows: $I(X;Y) = H(X) - H(X|Y)$. The output value range for the V-measure is $[0,1]$, with high values indicating a high level of similarity between the two partitions, and a low value indicating a low level of similarity.

$$\text{Homogeneity} = 1 - \frac{H(C^*|C)}{H(C^*)}. \tag{17}$$

$$\text{Completeness} = 1 - \frac{H(C|C^*)}{H(C)}. \tag{18}$$

$$\text{V-measure} = 2\frac{\text{Homogeneity} \times \text{Completeness}}{\text{Homogeneity} + \text{Completeness}}. \tag{19}$$

The Folkes-Mallows Index (FMI) The FMI is another classic external validity index used to measure the similarity between two partitions. It is defined as the geometric mean of the Precision and the Recall [209]. It can be formulated as in Equation 20. The output value range for the FMI is $[0,1]$, with high values indicating a high level of agreement between the two partitions, and a low value indicating a low level of agreement.

$$\text{FMI} = \sqrt{\frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|} \times \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|}} = \sqrt{\text{Precision} \times \text{Recall}} \tag{20}$$

The Constrained Rand Index (CRI) The CRI is a revised version of the RI which includes constraints specifically in its definition. It introduces the concept of free decisions, which are defined as decisions not influenced by constraints. The CRI subtracts the number of available constraints from the numerator and the denominator of the classic RI [85, 212]. As a result, it only evaluates the performance of the CC methods in the free decisions. Equation 21 formalizes CRI, following the same notation as Equation 6 (RI). Its results are interpreted as those of RI, but taking into account that the difficulty to obtain values close to 1 increases with the size of the constraint set $|CS|$.

$$\text{CRI} = \frac{a + b - |CS|}{n(n-1)/2 - |CS|} \tag{21}$$

5.4 Constraint Generation Methods

The most frequently used procedure to generate constraints is the one proposed in [86]. It is a simple yet effective method to generate a set of constraints based on a set of labels, hence the generalized use of classification datasets as benchmarks in CC literature. It consists of randomly choosing pairs of instances and setting a constraint between them depending on whether their labels are the same (ML constraint) or different (CL constraint).

The way pairs of instances are chosen from the dataset may differ from one study to another. However, two common trends are observed. One first decides the percentage of labeled data the oracle has access to and then generates the complete constraints graph based on those labels. The other one first decides the size of the constraint set, and then extracts random pairs of instances from the complete dataset. On the one hand, the first method is more realistic in the sense that it has limited access to labeled data, although it may bias the solution towards poor local optima if the selected labeled instances are not representative enough of the whole dataset. On the other hand, the second constraint generation method has virtual access to the complete set of labels, as pairs of instances are randomly chosen, and the constraint set may end up involving all instances in the dataset in at least one constraint, which might not be a realistic scenario. Nevertheless, it is less likely to bias the solution towards local optima.

There is no consensus on how many constraints need to be generated in order to evaluate the capabilities of a given CC method. However, some general guidelines can be given. Based on Observation 5.1, it is clear that proper empiric evaluation of CC methods must include an averaging process on the results obtained for different constraint sets, in order to reduce the effects of specific adverse constraint sets.

Observation 5.1 ***Specific constraint sets can have adverse effects.*** *Even if constraint sets are generated on the basis of the true labels, some constraint sets may decrease accuracy when predicting those very labels [11].*

Given Observation 5.2, testing CC methods should include different levels of constraint-based information. This must be done in order to study the scaling capabilities of the proposed method. If a method does not scale the quality of the solutions with the size of the constraint set, any improvement over the solutions obtained with an empty constraint set may be due to random effects.

Observation 5.2 ***The accuracy of the predictions scales with the amount of constraint-based information.*** *The quality of the solution should scale with the size of the constraint set: the more constraint are available, the better the results obtained are [11].*

5.5 On the Use of Statistical Tests

Statistical testing is a settled practice in Computer Science. It provides objective evidence of the results of a study, supporting its conclusions, either if the used tests are Null Hypothesis Statistical Tests (NHST) [213] or the more recent Bayesian Tests [214]. However, this does not seem to be the case in the CC area. As shown in Figure 11, only 5,6% of the studies (16 out of 270) analyzed in this review use statistical testing to support their conclusions. Authors consider this to be one of the major criticisms of the area of CC. Studies supporting their conclusions on mere average results values for any validity index/indices (as it is the case for most of them) should be encouraged to use statistical testing to further objectively prove their hypotheses.

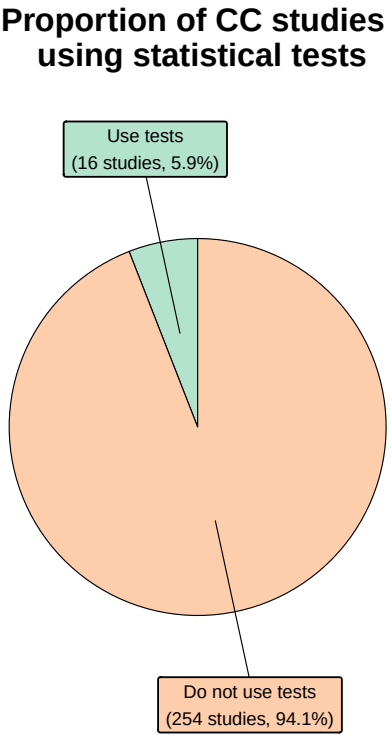


Figure 11: Piechart featuring the proportion of CC studies which use statistical tests.

6 Scoring System

The aim of this study is not only to give a taxonomy of constrained clustering methods, but also to provide researchers with tools to decide which methods to use. This section proposes an scoring system that is designed to indicate the potential and popularity of every reviewed method. This system assigns a numerical value to every CC method, which will be later used to rank all 307 of them. This value can be interpreted as a measure for the quality of

the method. Three semantically different aspects of every method $\mathcal{A}$ are analyzed to decide its score: the quality of the experimental setup they are tested in ($EQ_{\mathcal{A}}$), the confidence in the results obtained in the experiments ($VQ_{\mathcal{A}}$), and the influence of the method in later studies ($I_{\mathcal{A}}$). As for the formalization of these concepts, it is necessary to define the basic quantifiable elements that can be obtained from a method, which are shown in Table 5. All of them are lists that contain a value or a set of values associated to every method. Therefore, the length of these lists is always equal to the number of methods reviewed in this study. These lists can be accessed in a more precise way, by method or by year. For example: $D_{\mathcal{A}}$ is a single value which refers to the number of datasets used to test method $\mathcal{A}$, and D^Y is a list of values referring to the number of datasets used to test methods from year Y. Note that $M_{\mathcal{A}}$ is the list of methods used to compare $\mathcal{A}$, therefore M^Y is a list of lists.

Function	Meaning
Y	The list of publication years for all methods.
M	The lists of sets of methods used in comparisons.
C	The list of number of times a method is used to be compared with in later studies.
T	The list of indicators for the use of statistical tests for every method
D	The list of number of datasets used to test every method.
V	The list of number of validity indices used to evaluate every method.

Table 5: Functions to get basic features of methods.

In this study, authors have decided to evaluate each method within its time context, i.e. the year of publication of the method is taken into account to compute its score. This is done to remove the computational capacity component from the scoring system, as the number of datasets or the number of methods used to test new proposal is highly dependent of said parameter (see Figure 7 and Figure 8). Moreover, publication requirements and standards change over the years, and tend to become more rigid. Not taking the year of publication into account would greatly benefit recent methods, as their studies have to meet harder publication requirements which are usually related to their novelty and their experimental quality.

6.1 Scoring of the Experimental Quality

The quality of the experimental setup EQ used to test a method $\mathcal{A}$ can be computed with information that is fully contained in the study which proposes it. Two of the experimental elements introduced in Section 5 take part in this procedure: the number of datasets used to test the scoring method $\mathcal{A}$ (in list D), and the methods that are used to compare it (in list

M). Equation 22 gathers these two basic measures and gives the expression to compute the experimental quality of a scoring method $EQ_{\mathcal{A}}$.

$$EQ_{\mathcal{A}} = \frac{\alpha_1^{Y_{\mathcal{A}}} D'_{\mathcal{A}} + \alpha_2^{Y_{\mathcal{A}}} MS'_{\mathcal{A}}}{\alpha_1^{Y_{\mathcal{A}}} + \alpha_2^{Y_{\mathcal{A}}}}, \tag{22}$$

where the $MS_{\mathcal{A}}$ term is computed on the basis of $M_{\mathcal{A}}$, but taking the publication year of both the scoring method $\mathcal{A}$ and the compared method m into account, as shown in Formula 23. As a result, every compared method m contributes in an inversely proportional way to MS with respect to the difference between the years of publication of the two methods [1]. This way, methods published in years close to the year of publication of $\mathcal{A}$ contribute more to $MS_{\mathcal{A}}$ that methods published long time before $\mathcal{A}$. In other words: the contribution of every method is proportional to its novelty in the year it is used to make comparisons. Please note that non CC methods are always considered to be published one year before the first CC was published (1999). Subsequently, the contribution of non CC methods to $MS_{\mathcal{A}}$ decays invariably with the years. By doing this, the first CC methods comparing with classic clustering methods are given credit by the comparison, as no CC baseline methods could have been established by that time. However, this is not the case for modern CC methods, which must be compared to other CC methods for said comparison to be meaningful.

$$MS_{\mathcal{A}} = \frac{1}{|M_{\mathcal{A}}|} \sum_{m \in M_{\mathcal{A}}} \frac{1}{Y_{\mathcal{A}} - Y_m}. \tag{23}$$

Both the values of D and MS are normalized within each year following the normalization procedure described in Equation 24 (min-max normalization), which results in D' and MS'. This is done to lessen the effects that the computation capability context can have in $EQ_{\mathcal{A}}$. Please note that, only with respect to the year grouping aspects, methods published in years 2000-2003 are considered to belong to the same time context, hence they are treated as if they were published in the same year. With this in mind, neither Equation 23 nor Equation 28 are affected. This is done to enable withing-groups normalization and comparisons, as only 1 method was published in 2000 and 2001, and only 3 were published in 2002 and 2003. These were the years in which the CC research topic was conceived and it was starting to grow in interest (see Section 3.4). Subsequently, authors consider this exception to be justified.

$$D'_{\mathcal{A}} = \frac{D_{\mathcal{A}} - \min(D^{Y_{\mathcal{A}}})}{\max(D^{Y_{\mathcal{A}}}) - \min(D^{Y_{\mathcal{A}}})}, \quad MS'_{\mathcal{A}} = \frac{MS_{\mathcal{A}} - \min(MS^{Y_{\mathcal{A}}})}{\max(MS^{Y_{\mathcal{A}}}) - \min(MS^{Y_{\mathcal{A}}})}. \tag{24}$$

The last elements to be introduced from Equation 23 are the $\alpha_1^{Y_{\mathcal{A}}}$ and $\alpha_2^{Y_{\mathcal{A}}}$ values, which are different for every year. These values are used to determine the influence of the datasets score and the compared methods score in the computation of the experimental quality score. They are computed as in Equation 25, where $\sigma(\cdot)$ and $\mu(\cdot)$ are functions which return the

[1] This difference is considered to be 1 for methods published in the same year, in order to avoid divisions by 0.

standard deviation and the mean of the list of values given as argument, respectively. Subsequently, $\alpha_1^{Y_\mathcal{A}}$ and $\alpha_2^{Y_\mathcal{A}}$ are directly proportional to the standard deviation of the datasets scores and the compared methods scores, respectively. In other words, $\alpha_1^{Y_\mathcal{A}}$ and $\alpha_2^{Y_\mathcal{A}}$ are used to give more importance to disperse measures, which are usually good discriminators, and therefore are better suited to be used in a scoring system.

$$\alpha_1^{Y_\mathcal{A}} = \frac{\sigma({D'}^{Y_\mathcal{A}})}{\mu({D'}^{Y_\mathcal{A}})}, \quad \alpha_2^{Y_\mathcal{A}} = \frac{\sigma({MS'}^{Y_\mathcal{A}})}{\mu({MS'}^{Y_\mathcal{A}})}. \tag{25}$$

6.2 Scoring of the Validation Procedure Quality

Once again, the information needed to determine the quality of the validation procedures VQ used to evaluate the results obtained with a method $\mathcal{A}$ is fully contained in the study which proposes it. The two experimental elements (introduced in Section 5) that take part in this procedure are: the number of validity indices used to quantify the results obtained by the scoring method $\mathcal{A}$ (in list V), and the indicator of the use of statistical testing procedures (in list T). The list T indicates which methods use statistical tests by giving them a value of 1, whereas the 0 value is assigned to method that do not support their conclusions with statistical tests. Equation 26 shows the expression to compute the validation procedures quality of a scoring method $VQ_\mathcal{A}$.

$$VQ_\mathcal{A} = V'_\mathcal{A} + T_\mathcal{A}, \tag{26}$$

where $V'_\mathcal{A}$ is the normalized value of $V'_\mathcal{A}$, which is computed following formula 27. Please note that in this case the min-max normalization does not take the publication year into account (in contrast to Equation 24), as the number of validity indices used to quantify the results of the proposed methods does not show any tendency with respect to the publication year (see Figure 10). Authors consider studies which use statistical tests to have a significantly higher confidence rate in their results, hence the strength of the second term in Equation 26.

$${V'}_\mathcal{A} = \frac{V_\mathcal{A} - \min(V)}{\max(V) - \min(V)}. \tag{27}$$

6.3 Scoring of the Influence

The influence I of a given method $\mathcal{A}$ cannot be computed with just the information contained in the study which proposes the method. This aspect of the method refers to how influential it has been in later literature, i.e. how many times method $\mathcal{A}$ has been used to make experimental comparisons. This number differs from the total number of times it has been cited, as a citation does not guarantee that the method is being used to make comparisons. In fact, this is one of the hardest aspects to evaluate, and requires experimental comparisons carried out in a corpus of papers to be self-contained. This means that no method referred

in the experimental section of any paper is left out of the corpus. As will be explained in Section 7, authors have made sure that this is the case for the taxonomy presented in this study. However, once this information has been obtained, an index for the influence of any given method can be computed as simply as in Equation 28, where CY refers to the current year, therefore CY = 2022. This is, the number of times a method is used in experimental comparisons divided by the number of years it has been available.

$$I_{\mathcal{A}} = \frac{C_{\mathcal{A}}}{\mathrm{CY} - Y_{\mathcal{A}}}. \tag{28}$$

6.4 Final Scoring

The final scoring S of any given method $\mathcal{A}$ can be computed by normalizing and adding up the three partial scores presented in previous sections, and scaling the output range to $[0, 100]$. Equation 29 gives the expression to compute $S_{\mathcal{A}}$. Please note that none of the partial scores are bounded, hence the need of the min-max normalization step in Equation 30.

$$S_{\mathcal{A}} = \frac{(EQ'_{\mathcal{A}} + VQ'_{\mathcal{A}} + I'_{\mathcal{A}}) \times 100}{3}. \tag{29}$$

$$EQ'_{\mathcal{A}} = \frac{EQ_{\mathcal{A}} - \min(EQ)}{\max(EQ) - \min(EQ)}$$

$$VQ'_{\mathcal{A}} = \frac{VQ_{\mathcal{A}} - \min(VQ)}{\max(VQ) - \min(VQ)} \quad . \tag{30}$$

$$I'_{\mathcal{A}} = \frac{I_{\mathcal{A}} - \min(I)}{\max(I) - \min(I)}$$

Finally, authors want to remark that no hand-tuned parameter is needed to compute $S_{\mathcal{A}}$. Consequently, the probability of introducing any human bias in the scoring system is reduced.

7 Taxonomic Review of Constrained Clustering Methods

In this section, a ranked taxonomic classification for a total of 307 CC methods is presented. The starting point to obtain the corpus of CC studies to be reviewed was to run Query 7.1 in the Scopus scientific database.

Query 7.1 ***Scopus Query****: (TITLE-ABS-KEY ("constrained clustering") OR TITLE-ABS-KEY ("semi-supervised clustering") AND TITLE-ABS-KEY ("constraint" OR "constraints" OR "constrained"))*

This is a very general and wide query, which was conceived to make sure that the most of the CC research area was contained in its output. This search outputted 1162 indexed scientific papers in 24/3/2022. Authors briefly reviewed and evaluated all of these papers to remove those which did not belong to the CC research area. Afterwards, a recursive procedure was used to obtain the final corpus to be reviewed: if a study compares its proposal with a CC proposal not included in the corpus, then the newly identified study is included and applied this procedure over. This is done with the aim of producing a self-contained comparison.

Figure 12 presents a taxonomic tree, organizing the categories in which the CC landscape may be divided. Particular methods are introduced and discussed in Sections 7.1 to 7.17, where tables detailing the features of every method can be found.

As Figure 12 shows, a high-level dichotomy can be made within the CC area: constrained partitional methods versus constrained distance metric learning (DML) methods [11, 12]. The main difference lies in their approach to CC and in their output. In constrained partitional methods, constraints are included into a procedure that progressively builds a partition for the dataset. This is typically done by designing a clustering engine which can deal with constraints or by including constraints in the objective function of a given method, for example, by means of a penalty term. Generally, constrained partitional methods produce a partition of the dataset, which may be accompanied by other by-products of the CC method, such as new constraints or feature weights. On the other hand, constrained DML methods aim to learn a distance metric that reflects the information contained in the constraint set. In general, the learned distances will try to bring ML instances together in the output space, while trying to maximize the distance between CL instances. Generally, constrained DML method do not produce a partition of the dataset, but a new metric, data space or distance matrix. This output can be used to later produce a partition by means of classic clustering algorithms, or even by constrained clustering algorithms. Please note that the difference found between the tree classes of constrained DML methods is merely conceptual, as the results of all the three of them (new metric/data space/distance matrix) can always be derived from each other using classic DML methods. However, the distinction between the three classes is useful from the point of view of CC, as their approach to the problem is different. The vast majority of CC methods are constrained partitional methods. There are hybrid methods, which combine features inherited from both approaches.

Feature tables in Sections 7.1 to 7.17 generally include 8 columns:

- The $S_{\mathcal{A}}$ column gives the quality score assigned to each method. It is computed following the scoring system introduced in Section 6.
- The **Acronym** column provides the acronym of the method. Bearing in mind that some authors do not name their methods, we have decided to refer to these methods by the initials of their authors' names. However, there are exceptions for this rule, such as methods that are not named by their author but are consistently referred by later literature with a given name. In cases in which two methods have the same name, the year it was proposed in is added at the end of the name to differentiate them.

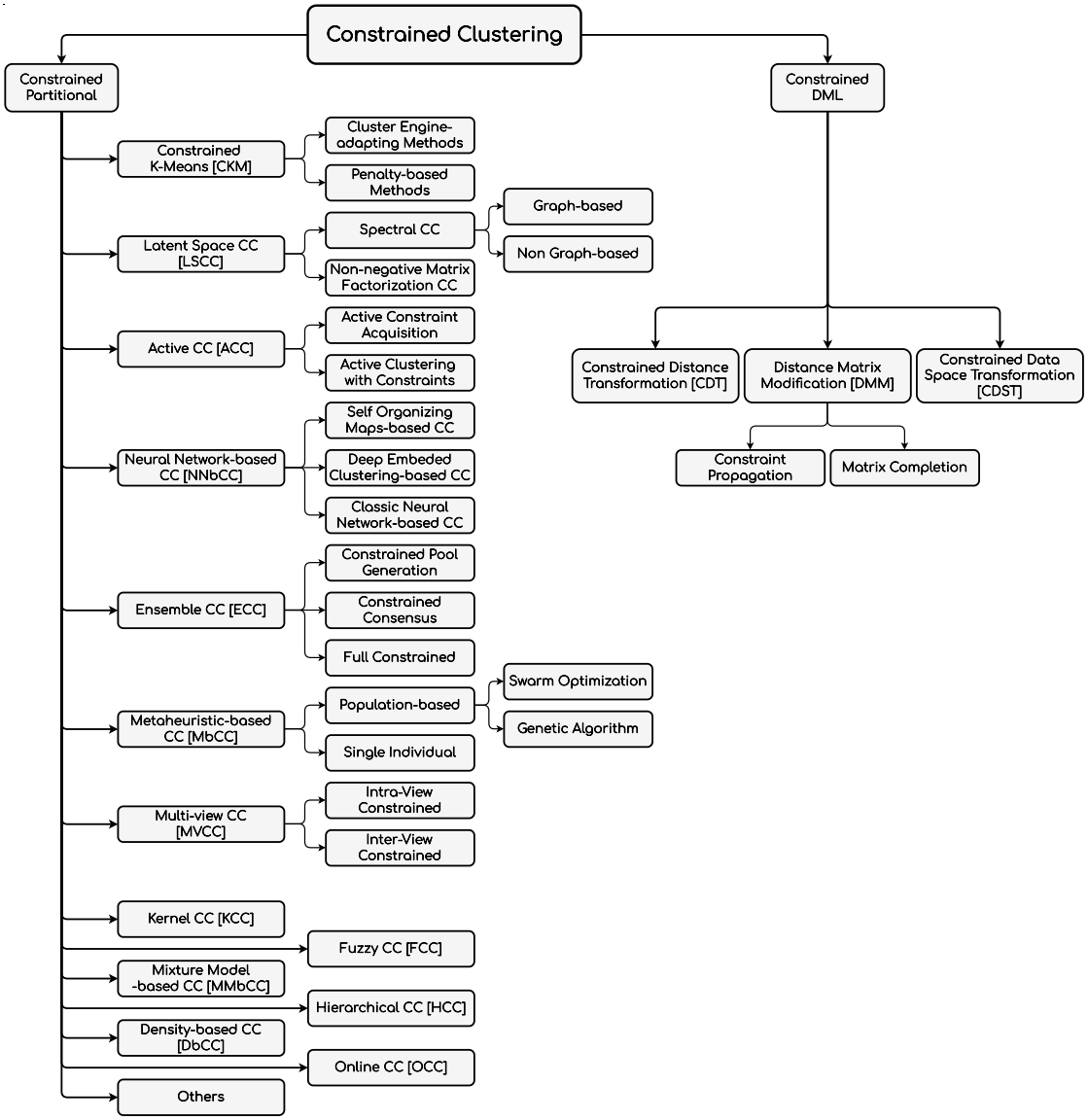


Figure 12: Taxonomic tree for the CC landscape.

- The **ID** column assigns a numeric identifier to each method. This number can be used to find the method in Appendix A, where the full name of all methods are listed, along with its identifier and its acronym. Full names are listed only in said appendix for the sake of readability and visualization.

- The **Penalty** column takes two values: "✓" or "×". This indicates whether constraints are included in the method by means of a penalty term in its objective function ("✓") or by other means ("×").

- The **ML** and **CL** columns refer to the type of constraints the method can handle. **Soft** is used for soft constraints, **Hard** is used for hard constraints, **Hybrid** is used for method that can use both hard and soft constraints. If a method cannot handle ML or CL constraints it is indicated with "-".
- The **Hybrid** column indicates if the method belongs to more than one class, specifying the classes it belongs to. The "-" character is used if the method belongs to only one class.
- The **Year** and **Ref** columns provide the year of publication and the reference of the method respectively.

7.1 Constrained K-Means

The Constrained K-Means (CKM) category gathers methods that can be considered modifications over the classic K-Means algorithm to include constraints. Their common feature is that all of them use an expectation-minimization (EM) optimization scheme. In the expectation step of an EM scheme, instances are assigned to clusters minimizing the error according to an objective function. In the minimization step, centroids are reestimated according to the assignations made in the expectation step. A plethora of objective functions and centroid update rules has emerged to approach the CC problem, although methods belonging to these category can be divided into two main categories.

7.1.1 Cluster Engine-adapting Methods

Cluster engine-adapting methods modify one of the two steps (or both) from the EM scheme in order to include constraints. Methods belonging to this category are presented in Table 6. The first and most basic method performing CC this way is COP-K-Means. It modifies the instance to clusters assignation rule from the expectation step (the clustering engine) so that an instance is assigned to the cluster associated to its closer centroid whose assignation does not violate any constraint. Another popular technique in this category consists of performing clustering over the previously computed chunklet graph (which enforces ML), considering only CL in the expectation step. This is how methods like CLAC, CLWC, PCCK-Means, PCBK-Means or SSKMP perform CC. All of them consider hard ML. They differ from each other in the way in which they build their particular chunklet graph, which may contain weighted chunklets (as in CLAC and CLWC), or may rank chunklets in order for them to be examined more efficiently (as in PCCK-Means). Other methods use basic chunklet graph (like PCBK-Means and SSKMP). Some methods include constraints in EM scheme that are not basic K-Means, like SSKMP, which is a constrained version of the K-Medoids algorithm. Based on the COP-K-Means, methods like CLC-K-Means or ICOP-K-Means are designed to solve the dead-ends problem found in the basic algorithm.

7.1.2 Penalty-based Methods

Penalty-based methods include constraints by means of a penalty term in the objective function of an EM scheme. These methods are presented in Table 7. Some of them simply modify

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
33.33	COP-K-Means	2	×	Hard	Hard	—	2001	[86]
16.94	PCSK-Means	42	×	Hard	Soft	—	2007	[215]
7.62	CLAC	55	×	Hard	Hard	Graph-based	2008	[216]
7.56	MLC-K-Means	57	×	Soft	Soft	—	2008	[217]
12.54	CLWC	66	×	Hard	Soft	—	2008	[188]
7.90	COPGB-K-Means	71	×	Soft	Soft	Graph-based	2008	[218]
9.14	PCCK-Means	72	×	Hard	Soft	—	2008	[219]
3.79	SCK-Means	76	×	Hybrid	Hybrid	—	2009	[45]
5.16	CMSC	90	×	Soft	Soft	—	2009	[220]
6.51	SCKMM	101	×	Soft	Soft	Constrained Distance Transformation	2010	[221]
6.04	PCBK-Means	102	×	Hard	Soft	—	2010	[221]
5.82	ICOP-K-Means	104	×	Hard	Hard	—	2010	[222]
10.24	CLC-K-Means	115	×	Hard	Hard	—	2011	[223]
5.24	SKMS	190	×	Soft	Soft	—	2014	[224]
4.94	BCK-Means	255	×	Hard	Hard	—	2019	[225]
8.73	SSKMP	256	×	Hard	Hard	—	2019	[226]

Table 6: Feature table for CKM - Cluster Engine-adapting Methods.

previous CC or classic clustering algorithms to include a plain penalty term, such as SCOP-K-Means, PCK-Means, S-SCAD. Other methods, like MPCK-Means, also include a metric learning step in the EM scheme, which estimates a cluster-local distance measure for every cluster, and allows them to find clusters with arbitrary shapes. Besides, there are methods which use variable penalty terms, such as HMRF-K-Means, CVQE, LCVQE or CVQE+ that include the distance between constrained instances in it. This is, more relevance is assigned to ML relating distant instances and CL relating close instances. Methods which combine pairwise constraints and other types of constraints have also emerged, like PCS, which includes cluster-size constraints in its EM scheme too. Other methods like GPK-Means use a Gaussian function and the current cluster centroids to infer new constraints in the neighborhood of the original constraints. These new constraints are added to the constraint set and used in subsequent iterations of the EM scheme.

7.2 Latent Space CC

Latent space clustering performs clustering in a space which is different from the input space and which is computed on the basis of the dataset, and also the constraint set in Latent Space CC (LSCC). The input to these algorithms is an adjacency matrix defining the topology of the network (or graph) over which clustering needs to be performed. Each row or column may be regarded as the feature or property representation of the corresponding node. Latent space clustering methods first obtain new property representations in a latent space for each node by optimizing different objective functions, and then clusters nodes in that latent space [240].

S_A	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
0.00	SCOP-K-Means	3	✓	Soft	Soft	—	2002	[227]
16.18	PCK-Means	9	✓	Soft	Soft	—	2003	[87, 189]
20.61	MPCK-Means	11	✓	Soft	Soft	Constrained Distance Transformation	2003	[87, 228]
8.97	HMRF-K-Means	12	✓	Soft	Soft	Constrained Distance Transformation	2004	[194]
13.99	GPK-Means	18	✓	Soft	Soft	—	2005	[229]
8.79	CVQE	19	✓	Soft	Soft	—	2005	[81]
8.92	LCVQE	44	✓	Soft	Soft	—	2007	[230]
6.16	S-SCAD	52	✓	Soft	Soft	—	2007	[231]
4.37	SemiStream	127	✓	Soft	Soft	Online CC	2012	[205]
17.64	PC-HCM-NM	176	✓	Soft	Soft	—	2014	[232]
2.51	AC-CF-tree	185	✓	Soft	Soft	Active Clustering with Constraints	2014	[147]
4.68	PCS	191	✓	Soft	Soft	—	2014	[233]
4.34	TDCK-Means	193	✓	Soft	Soft	Online CC	2014	[234]
12.47	HSCE	207	✓	Soft	Soft	Non Graph-based & Constrained Pool Generation	2015	[235, 236]
2.47	CVQE+	242	✓	Soft	Soft	Active Clustering with Constraints	2018	[237]
3.00	PCSK-Means(21)	276	✓	Soft	Soft	—	2021	[238]
16.80	fssK-Means	279	✓	Soft	Soft	Time Series	2021	[239]

Table 7: Feature table for CKM - Penalty-based Methods.

7.2.1 Spectral CC

Classic spectral clustering algorithms try to obtain the latent space by finding the most meaningful eigenvectors of the adjacency matrix, which are used to define the embedding in which clusters are eventually obtained [240]. A dichotomy can be made within this category: in graph-based methods the input data is always given in the form of a graph, while in non-graph based the input is an adjacency matrix or a regular dataset, which can be transformed into an adjacency matrix. Please note that this distinction only affects the conceptual level of the spectral CC category, as a graph can always be converted to and adjacency matrix and vice versa, all methods from one category may also be applied in the other category. However, the authors have decided to make this distinction, since the terminology and concepts used in the studies referring to each of them differ greatly and can be misleading if interpreted together.

Graph-based spectral clustering In graph-based spectral clustering, the input is assumed to be a graph. The goal is to partition the set of vertices of the graph, taking into account the information contained in the vertices themselves and in the edges of the graph. Edges may carry similarity or dissimilarity information regarding the vertices they connect. Some common strategies to perform graph clustering try to maximize the similarity of vertices within a cluster, normalizing the contribution of each to the objective by the size of the cluster in order to balance the size of the clusters. Other methods try to minimize the total cost of the edges crossing the cluster boundaries [241]. Graph-based methods are particularly suitable to perform CC, as constraints can be naturally represented in their graph form, which is the constraint graph and the chunklet graph (introduced in Section 4).

Table 8 gathers graph-based spectral clustering methods. COP-b-coloring and CLAC exemplify the use of chunklets to enforce ML, while including CL by other means. Other methods modify the input graph to include the information contained in the constraint set. For example, PCOG modifies affinities so that ML instances are always placed in the same connected components and removes edges which connect CL instances. CCHAMELEON modifies affinities between constrained instances, making them larger if instances are related by ML and lower in the case of CL. The all-pairs-shortest-path algorithm is used to propagate changes. PAST-Toss uses a spanning tree based technique to perform CC directly over the constraint graph. SCRAWL is the only non-spectral graph-based CC algorithm, as it does not need pairwise similarity/dissimilarity information to perform CC, but graph-related measures instead.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
4.35	COP-b-coloring	38	×	Hard	Hard	—	2007	[242]
3.56	PAST-Toss	54	×	Soft	Soft	Single Individual	2008	[243]
7.62	CLAC	55	×	Hard	Hard	Cluster Engine-adapting Methods	2008	[216]
7.90	COPGB-K-Means	71	×	Soft	Soft	Cluster Engine-adapting Methods	2008	[218]
8.29	GBSSC	98	×	Hard	Soft	Dimensionality Reduction	2010	[185, 244, 245, 246]
4.54	CCHAMELEON	112	×	Soft	Soft	—	2011	[247]
11.27	SCRAWL	184	×	Soft	Soft	—	2014	[192]
5.37	PCOG	269	×	Hard	Hard	Non Graph-based	2020	[90]

Table 8: Feature table for the LSCC - Spectral CC - Graph-based methods.

Non graph-based spectral clustering In these methods, the input is given in the form of an adjacency matrix, or a dataset whose adjacency matrix can be easily obtained. Two techniques are commonly used to include pairwise constraints in these methods. (1) Modifying similarities/dissimilarities in the original adjacency matrix, computing eigenvectors and eigenvalues to obtain the spectral embedding. (2) Using the constraints to directly modify the embedding, obtained on the basis of the original adjacency matrix. Any classic clustering method can be used to obtain the final partition in the new embedding, which can always be mapped to the original data [240].

Table 9 shows a list of non-graph-based spectral CC methods. The first spectral CC method is found in KKM/SL, known in the literature by these two acronyms (respectively obtained from the name of its authors and the title of the study which proposes it). It is based on HMRF, performing CC by modifying the transition probabilities of the field based on the constraints. KKM/SL and AHMRF constitute the only two HMRF-based approaches to spectral CC. Another common technique in spectral CC is learning a kernel matrix based on the dataset over which spectral clustering is later conducted, as in RSCPC, CCSR, CCSKL, LSE or SSCA. This kernel matrix is usually built taking both pairwise distances and constraints into account. With respect to the methods which modify the original adjacency matrix, two strategies are the most used ones: some methods, such as ACCESS or CSC, simply set entries which relate constrained instances to specific fixed values, while other methods, such

as NSDR-NCuts or LCPN, use constraint propagation techniques to propagate changes in the affinity matrix once it has been modified.

S_A	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
14.40	KKM	6	×	Soft	Soft	—	2003	[248]
11.11	TBJSBM	7	×	Hybrid	Hybrid	—	2004	[249]
12.73	ACCESS	16	×	Soft	Soft	Active Constraint Acquisition	2005	[250]
6.88	CSC	20	×	Soft	Soft	—	2005	[251]
12.47	LCPN	61	×	Soft	Soft	Constraint Propagation	2008	[252]
8.40	S3-K-Means	63	✓	Soft	Soft	—	2008	[109]
3.68	RSCPC	86	×	Soft	Soft	—	2009	[253]
15.39	CCSR	87	✓	Soft	Soft	—	2009	[197]
6.82	SCLC	88	×	Soft	Soft	—	2009	[254]
15.81	CCSKL	89	×	Soft	Soft	Kernel CC	2009	[198]
8.76	CSP	105	×	Hybrid	Hybrid	—	2010	[255]
13.47	ASC	110	×	Any	Any	Active Clustering with Constraints	2010	[256]
8.57	SSC-ESE	152	✓	Soft	Soft	—	2012	[257]
3.90	IU-Red	144	×	Soft	Soft	Active Clustering with Constraints	2012	[258]
5.65	LSE	148	×	Soft	Soft	—	2012	[259]
7.73	NSDR-NCuts	135	×	Soft	Soft	—	2012	[260]
13.38	COSC	138	×	Hybrid	Hybrid	—	2012	[261]
3.62	SSCA	181	×	Soft	Soft	—	2014	[262]
6.78	FHCSC	183	×	Hybrid	Hybrid	—	2014	[263]
6.93	CNP-K-Means	187	×	Soft	Soft	Dimensionality Reduction	2014	[264]
4.46	STSC	188	×	Soft	Soft	Matrix Completion	2014	[265]
5.98	LXDXD	200	×	Soft	Soft	Non-negative Matrix Factorization CC	2015	[240]
12.47	HSCE	207	×	Soft	Soft	Constrained Pool Generation & Penalty-based Methods	2015	[235, 236]
8.82	FAST-GE	218	×	Soft	Soft	—	2016	[266]
6.65	URASC	227	×	Soft	Soft	Active Clustering with Constraints	2017	[267]
16.54	FAST-GE2.0	231	×	Soft	Soft	—	2017	[268]
12.11	TI-APJCF	233	✓	Soft	Soft	—	2017	[49]
12.11	TII-APJCF	234	✓	Soft	Soft	—	2017	[49]
1.89	AHMRF	246	×	Soft	Soft	—	2018	[212]
18.17	MVCSC	261	×	Soft	Soft	Intra-View Constrained	2019	[269]
15.04	SFS^{3}EC	257	×	Soft	Soft	Full Constrained	2019	[270]
5.37	PCOG	269	×	Hard	Hard	Graph-based	2020	[90]

Table 9: Feature table for the LSCC - Spectral CC - Non Graph-based methods.

7.2.2 Non-negative Matrix Factorization CC

Non-negative Matrix Factorization (NMF) clustering algorithms obtain the new representation of the data by factorizing the adjacency matrix into two non-negative matrices [240]. These two matrices can be interpreted as the centroids of the partition and the membership degree of each instances to each cluster. By doing this, all instances can be obtained as a linear combination of each column of the centroids matrix, parameterized by its corresponding membership, found in its associated row from the membership matrix. It can be proven that minimizing the difference between the original dataset matrix and the product matrix (com-

puted usually as the Frobenius norm) is equivalent to performing K-Means clustering over the dataset [271].

Table 10 presents a list of NMF-based CC methods. One of the most common strategies to include constraints into the classic NMF-based methods is to modify its objective function. This can be done by means of a penalty term accounting for the number of violated constraints, as in PNMF, SSCsNMF and SS-NMF(08), or by more complex techniques, as in CPSNMF or NMFS. Another popular strategy is forcing affinities between ML instances to be 0 and affinities between CL instances to be 1, as in NMFCC and SymNMFCC.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
5.68	NMFS	45	×	Soft	Soft	—	2007	[272]
16.28	SS-NMF(08)	73	×	Soft	Soft	—	2008	[273]
4.74	OSS-NMF	96	×	Soft	Soft	Co-Clustering	2010	[98]
11.88	SS-NMF	109	×	Soft	Soft	Co-Clustering	2010	[274]
4.51	PNMF	123	✓	Soft	Soft	—	2011	[116]
6.47	SSCsNMF	129	×	Soft	Soft	—	2012	[275]
5.98	LXDXD	200	×	Soft	Soft	Non Graph-based	2015	[240]
4.45	CPSNMF	211	×	Soft	Soft	Constraint Propagation	2016	[276]
8.20	NMFCC	219	×	Soft	Soft	—	2016	[277]
11.94	SymNMFCC	220	×	Soft	Soft	—	2016	[277]
5.02	PCPSNMF	251	×	Soft	Soft	—	2018	[278]
2.85	CMVNMF	254	×	Soft	Soft	Inter-View Constrained & Active Clustering with Constraints	2018	[91, 271]

Table 10: Feature table for the LSCC - Non-negative Matrix Factorization CC methods.

7.3 Active CC

Active learning is a subfield of machine learning in which algorithms are allowed to choose the data from which they learn. The goal of active learning is to reduce the amount of supervisory information needed to learn, an therefore reduce the human effort and implication in machine learning. In the active learning paradigm, learning methods are provided with an oracle, which is capable of answering a limited number of an specific type of query. For example, in traditional classification, active learning is used to select the best instances to be labeled from a dataset, so the oracle provides the label of the specific queried instance [279]. In Active CC (ACC), the oracle is queried about the type of constraint relating pairs of instances. The key aspect in any active learning algorithm is how to choose the queries to be presented to the oracle.

Active learning is specially useful in CC. In order to have an explanation for this, we compare the complexity of the answers given by oracles involved in active classification and ACC. In active classification, the oracle is queried about the class of a given instance. This query has a virtually infinite number of answers, as the number of classes in the dataset may be unknown. On the other hand, an oracle involved in CC is queried with two instances and asked about the constraint between them (ML or CL), which is the same as asking whether they belong to the same class. There are only three possible answers to this question: “yes”,

"no" or "unknown". It is clear that the oracle in CC carries out a far simpler job than the one in classification. Let us remember that the oracle is just an abstraction of a knowledge source, which is generally a human user. Querying a human about the relation between instances instead of about their class requires less effort from them and leads to less variability and noise in the queries, as the extensive literature in active CC shows.

Two subcategories can be found in Active CC. In active constraint acquisition, constraints are actively generated before performing constrained clustering, while in active clustering with constraints, both clustering and active constraint generation are performed iteratively at the same time. This way, in active constraint acquisition queries are generated on the basis of the dataset and the current state of the constraint set, while in active clustering with constraints information about the current partition can also be used.

7.3.1 Active Constraint Acquisition

The immediate result of active constraint acquisition methods is a set of constraints, rather than a partition of the dataset. However, the partition can be obtained using any other CC method by just feeding the generated constraints into it, along with the dataset used to generate the constraints. Only the dataset, the constraint set generated so far, and an initial unconstrained partition are available to perform active constraint learning in the active constraint acquisition paradigm. No CC algorithm is involved in the constraint acquisition step. Table 11 shows a list of active constraint acquisition methods. Columns indicating the type of constraint these methods can handle have been removed, as they are not relevant here. Column "CC Method" has been added, indicating the CC method used to produce a partition based on the constraint generated by every active constraint acquisition method in the experimental section of the studies that propose them.

Many strategies to select the best pair of instances to query to the oracle have been proposed. Some methods start by dividing the dataset into preliminary groups and then use the oracle to query constraints which consolidate that information, such as FFQS, MMFFQS, SSL-EC or LCML. Other methods focus on finding the boundaries in the dataset to select pairs of instances from them, such as ACCESS, ASC(10) or SACS. Besides, there are methods that use classic clustering to obtain preliminary information from the dataset, such as the co-association of instances or the compactness of clusters. DGPC, JDFD, WAKL, MICS, AAA(19), AIPC, ALPCS or ASCENT are some of the methods which use this strategy. Other methods focus on specific features from the constraint themselves in order to evaluate them and select the more informative ones, such as AAVV or KAKB. More complex approaches can be found in AAA(18), which solves the active constraint acquisition problem as an instance of the uncapacited k-facility location problem, or RWACS, which uses the commute time from graph theory to select queries.

$S_{\mathcal{A}}$	Acronym	ID	CC Methods	Hybrid	Year	Ref.
16.24	FFQS	10	—	—	2004	[189]
12.73	ACCESS	16	—	Non Graph-based	2005	[250]
7.00	DGPC	289	PCK-Means	—	2007	[280]
13.19	MMFFQS	69	MPCK-Means	—	2008	[281]
9.42	ASC(10)	106	AHCC, MPCK-Means	—	2010	[282, 191, 283]
7.24	KAKB	290	S3OM	—	2011	[284]
4.63	SSL-EC	131	—	Constraint Propagation	2012	[285]
5.74	Cons-DBSCAN	137	—	Hierarchical CC & Density-based CC	2012	[111]
8.62	JDFD	292	—	—	2013	[286]
8.64	SACS	186	Xiang's, RCA, MPCK-Means	—	2014	[287]
6.34	WAKL	293	MPCK-Means	—	2014	[288]
4.27	MICS	294	RCA+K-Means	—	2015	[289]
11.85	CCCPYL	203	DBSCAN, COP-K-Means, KKM, CSI, MPCK-Means	—	2015	[290]
6.63	LCML	300	MPCK-Means	—	2016	[291]
7.12	AAA(18)	296	MPCK-Means, RCA	—	2018	[292]
5.88	RWACS	297	MPCK-Means, RCA	—	2018	[293]
9.91	AAA(19)	298	MPCK-Means, RCA	—	2019	[294]
18.34	AIPC	301	PCK-Means	—	2019	[201]
5.65	AAVV	302	MPCK-Means, RCA	—	2020	[295]
7.36	ALPCS	299	—	—	2020	[296]
3.93	ASCENT	307	MPCK-Means	—	2020	[193]

Table 11: Feature table for the ACC - Active Constraint Acquisition methods.

7.3.2 Active Clustering with Constraints

In active clustering with constraints, a CC procedure and a constraint generation method are applied alternately. The immediate result of these methods are both a partition of the dataset and a constraint set. These methods usually start by computing an unconstrained partition of the dataset. After this, some criteria are applied to select pairs of instances to query the oracle on the basis of the obtained partition. The answers to these queries are used to generate and save new constraints, which are later used to generate a new partition of the dataset by means of a CC method. Active clustering with constraints methods iterate these steps to produce the final constraint set and the partition. The active constraint generation method can be dependent of the CC method used to produce partitions, in which case they cannot be used separately. On the other hand, some active constraint generation methods are designed to be paired with any CC algorithm.

Table 12 gathers a list of active clustering with constraints methods. A major trend in this category is found in the use of the uncertainty of instances to rank and pair them to select the more uncertain ones and query them to the oracle. The uncertainty is always computed based in the current partition and is usually defined as the probability of an instance belonging to different known clusters. Some methods in this category are: RHWL, IU-Red, ALC-SSC, CMKIPCM, AAA, URASC, A-COBS, ADP and ADPE. There are as well other criteria to select pairs of instances to query, such as the utility maximization (SRBR), the maximum expected error reduction (ASC), the partition change maximization (Active-HACC), the ensemble consensus (PT), or the classic informativeness and coherence (A-ITML-K-Means). Cluster-related criteria can also be used to select queries, such as the size and distance

between the clusters (CAC, COBRA) or how well defined the frontiers are between them (AFCC, CVQE+). Paradigm-specific criteria are used by some methods, such as CMVNMF, which performs multi-view clustering and selects pairs of instances to query, based on intra-view and inter-view criteria, or AC-CF-tree and COBRAS, which use queries to determine the best cluster merge to perform in hierarchical CC. Similarly, the family of active FIECE-EM use concepts related to the population of individuals it maintains to select the best instances to query.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
7.14	RHWL	43	×	Soft	Soft	Probabilistic Clustering	2007	[96]
7.62	PT	56	×	Soft	Soft	Constrained Pool Generation	2008	[216]
9.50	AFCC	59	×	Soft	Soft	Fuzzy CC	2008	[297]
13.47	ASC	110	—	Any	Any	Non Graph-based	2010	[256]
4.17	CAC1	113	—	Hard	Soft	Hierarchical CC	2011	[140]
3.90	IU-Red	144	×	Soft	Soft	Non Graph-based	2012	[258]
7.86	SRBR	155	×	Soft	Soft	—	2013	[298]
5.62	A-ITML-K-Means	156	×	Soft	Soft	Constraint Propagation	2013	[299]
9.09	ALCSSC	180	×	Soft	Soft	—	2014	[300]
2.51	AC-CF-tree	185	✓	Soft	Soft	Penalty-based Methods	2014	[147]
3.57	Active-HACC	189	×	Soft	Soft	Hierarchical CC	2014	[142]
16.77	CMKIPCM	202	×	Soft	Soft	Fuzzy CC	2015	[301]
21.19	AAA	221	×	Soft	Soft	Fuzzy CC	2016	[302]
4.20	COBRA	225	×	Soft	Soft	Hierarchical CC	2017	[108]
6.65	URASC	227	×	Soft	Soft	Non Graph-based	2017	[267]
8.76	A-COBS	241	×	Soft	Soft	Constrained Consensus	2017	[303]
2.47	CVQE+	242	×	Soft	Soft	Penalty-based Methods	2018	[237]
19.57	COBRAS	244	×	Soft	Soft	Hierarchical CC	2018	[304]
2.85	CMVNMF	254	×	Soft	Soft	Non-negative Matrix Factorization CC & Inter-View Constrained	2018	[91, 271]
15.56	FIECE-EM+BFCU	303	×	Hard	Hard	Genetic Algorithm & Mixture Model-based CC	2020	[305, 306]
15.56	FIECE-EM+FCU	304	×	Hard	Hard	Genetic Algorithm & Mixture Model-based CC	2020	[305, 306]
15.56	FIECE-EM+DVO	305	×	Hard	Hard	Genetic Algorithm & Mixture Model-based CC	2020	[305, 306]
15.56	FIECE-EM+LUC	306	×	Hard	Hard	Genetic Algorithm & Mixture Model-based CC	2020	[305, 306]
17.22	ADPE	281	×	Soft	Soft	Density-based CC & Constrained Pool Generation	2021	[307]
17.84	ADP	282	×	Soft	Soft	Density-based CC	2021	[307]

Table 12: Feature table for the ACC - Active Clustering with Constraints methods.

7.4 Neural Network-based CC

Neural Networks (NN) are universal approximators which have been applied in many machine learning tasks, and CC is not an exception. Neural Network-based CC (NNbCC) tackles the CC from the NN perspective in three different ways: through self organizing maps, through deep-embeded clustering and through classic neural networks architectures.

7.4.1 Self Organizing Maps-based CC

Self organizing maps are NN (usually with fixed topology) whose neurons modify their position in the solution space to organize themselves according to the shape of the clusters. The result is a net whose neurons are grouped in clusters, which can be used to determine the

cluster every instance belongs to. Constraints can be included into this process in different ways, as Table 13 shows. Some on them, like SS-FKCN, simply use a penalty term accounting for violated constraints in a classic SOM variant. Others such as PrTM and SSGSOM, reformulate the classic SOM problem and use multiple neuron layers, forcing instances to flow through these to layers to be ultimately assigned to the appropiate cluster. In order to do so, PrTM uses constraint-influenced probabilities to decide how the position of the neurons changes, while SSGSOM adjusts the between-layer weights and the number of nodes of the first layer to dynamically correct the violation of constraints. Simpler methods like the S3OM, modify classic SOM for it to carry out only assignations without violating any constraints, similarly to COP-K-Means.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
5.74	SS-FKCN	36	✓	Soft	Soft	—	2006	[127]
10.13	PrTM	77	×	Soft	Soft	—	2009	[308]
7.70	S3OM	111	×	Hard	Hard	—	2011	[284]
10.74	CS2GS	201	×	Soft	Soft	Online CC	2015	[309]

Table 13: Feature table for the NNbCC - Self Organizing Maps-based CC methods.

The main difference between SOM-based approaches to CC and the other two approaches (deep embedded clustering and classic neural networks) is that the primary goal of the former is to produce a partition of the dataset, while the latter's is to cast predictions over unseen instances regarding the cluster they belong to. Please note that a partition can be obtained with deep embedded clustering and classic neural networks by feeding the training instances to the trained model.

7.4.2 Deep Embedded Clustering-based CC

In deep embeded clustering-based CC constraints are included into the classic Deep Embeded Clustering (DEC) model. Table 14 gathers methods which use this approach. SDEC includes constraints into the classic DEC model by using constraints to influence its distance learning step, DCC does so by simply modifying the loss function of DEC with a penalty term. CDEC uses DEC to initialize its encoder, which is finally retrained to finally assign instances to clusters and satisfy the constraints.

7.4.3 Classic Neural Network-based CC

Lastly, classic neural network-based CC methods are presented in Table 15. Some of them, such as S^3C^2 and CDC, use the siamese neural networks, as they are known, to solve the CC problems in two steps. In the case of S^3C^2, the siamese neural network is used to solve the two steps in which the CC is decomposed into simpler binary problems, while CDC uses the siamese neural network to perform unsupervised clustering and a triple NN to perform

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
8.32	SDEC	260	×	Soft	Soft	—	2019	[310]
10.31	DCC	265	×	Soft	Soft	—	2020	[43, 311]
2.44	CDEC	275	×	Soft	Soft	—	2021	[93]

Table 14: Feature table for the NNbCC - Deep Embeded Clustering-based CC methods.

CC. NN-EVCLUS simply implements EVCLUS in an NN setup and uses a penalty term in its loss function to include constraints.

SNNs consist of a NN model designed to learn non-linear similarity measures from pairwise constraints and to generalize the learned criterion to new data pairs. A SNN is a feedforward multi-layer perceptron whose learning set is defined as triplets composed of two instances and the constraint set between them, using 1 for ML and 0 for CL. In other words, ML instances have an associated target equal to 1, while CL instances have an associated target equal to 0. From the architectural point of view, the SNN has an input layer which accepts pairs of instances, a single hidden layer which contains an even number of units, and an output neuron with sigmoidal activation. The training of the SNN can be performed using the standard backpropagation scheme. Since the metric learned by an SNN cannot be straightforwardly used by a K-Means style algorithm, as the centroids do not necessarily have to be found in the dataset, centroids computation can be embedded in the SNN by using a classic K-Means minimization scheme based on backpropagation. This scheme keeps the weights and biases of the trained SNN fixed and varies the centroid coordinates (seen as free parameters). This is equivalent to redefining the original SNN model by adding a new layer to the network structure whose neuron activation functions correspond to the identity mapping.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
7.01	SNN	136	×	Soft	Soft	Constrained Distance Transformation	2012	[312]
15.54	S^3C^2	268	×	Soft	Soft	—	2020	[313]
10.09	CDC	274	×	Soft	Soft	—	2021	[314]
3.56	NN-EVCLUS	284	✓	Soft	Soft	—	2021	[92]

Table 15: Feature table for the NNbCC - Classic Neural Network-based CC methods.

7.5 Ensemble CC

Ensemble clustering methods usually perform clustering in two steps. (1) generating a pool of solutions, whose diversity depends on the method (or methods) used for the generation. (2) taking the pool of solutions as input and producing a single final solution by merging or selecting solutions from the pool. The function in charge of this procedure is called the consensus function. The application of ensemble-based clustering methods on the constrained clustering problem gives place to a new distinction within this category, which classifies Ensemble CC (ECC) methods depending on the step (or steps) in which they consider constraints.

7.5.1 Constrained Pool Generation

These ensemble methods use constraints in the pool generation step (the first step), i.e.: the partitions in the pool of solutions are generated with CC methods. Table 16 presents the list of methods belonging to this category. The consensus functions used by these methods do not take constraints into account. Therefore they are not considered in any distinction made within this category. The most commonly used consensus functions are majority voting, NCuts and CSPA.

Many methods use the subspace technique, which consists of performing clustering in a new space with a lower number of dimensions than the original space. The technique used to produce different subspaces introduces variability on the pool of solutions. The most common procedure used to generate the subspaces is simply a random sampling of the original features, such as in SCSC, ISSCE, RSSCE, CESCP, DCECP or ADPE. However, there are subspace generation methods specifically designed for certain algorithms. An example of this is SMCE, which uses the CSI method to project instances and constraints into multiple low-dimensional subspaces and then learning positive semi-definite matrices therein.

Other methods simply use any previous CC algorithm to produce the pool. The most common way to introduce diversity in the pool is by applying different CC methods to produce different partitions. Methods that use this strategy are SCEV, MVSCE, E^{2}CPE, HSCE or FQH. Another (and less used) method to generate diversity is varying the hyperparameters of a single CC method, as in Samarah.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
5.33	SMCE	37	×	Soft	Soft	—	2006	[315]
7.62	PT	56	×	Soft	Soft	Active Clustering with Constraints	2008	[216]
10.81	Samarah	99	×	Soft	Soft	—	2010	[316]
0.00	SCEV	146	×	Soft	Soft	—	2012	[317]
11.05	MVSCE	164	×	Soft	Soft	—	2013	[318]
1.25	E^{2}CPE	170	×	Soft	Soft	—	2013	[319]
5.48	SCSC	206	×	Soft	Soft	Online CC	2015	[320]
12.47	HSCE	207	×	Soft	Soft	Non Graph-based & Penalty-based Methods	2015	[235, 236]
23.05	ISSCE	214	×	Soft	Soft	Constraint Propagation	2016	[321]
20.25	RSSCE	215	×	Soft	Soft	Constraint Propagation	2016	[321]
6.09	FQH	232	×	Soft	Soft	—	2017	[322]
7.83	CESCP	252	×	Soft	Soft	Constraint Propagation	2018	[323]
10.64	DCECP	253	×	Soft	Soft	Constraint Propagation	2018	[323]
17.22	ADPE	281	×	Soft	Soft	Density-based CC & Active Clustering with Constraints	2021	[307]

Table 16: Feature table for the ECC - Constrained Pool Generation methods.

7.5.2 Constrained Consensus

In constrained consensus ensemble methods, constraints are used only in the consensus function to produce a final partition meeting as much constraints as possible. Table 17 gathers the four methods which belong to this category. All of these methods generate the

partitions in the pool by means of classic clustering algorithms. This is why the consensus functions used by these methods usually measure the quality of the generated solutions with respect to the constraints by means of a quality index and select the best ones to be finally merged. For example COBS and A-COBS use the infeasibility to select the best partition in the pool, which is generated by any classic clustering method. WECR K-Means runs classic K-Means multiple times with different hyperparameters to generate the pool, then a weighting procedure is used to automatically assign a weight to every partition depending on their local and global quality, which includes the infeasibility. A weighted co-association matrix based consensus approach is then applied to achieve a final partition. Semi-MultiCons builds a tree-like pool or partitions and then applies a normalized score which measures constraint satisfaction if any given merge or split operation between clusters is performed in the tree.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
7.00	COBS	240	×	Soft	Soft	—	2017	[303]
8.76	A-COBS	241	×	Soft	Soft	Active Clustering with Constraints	2017	[303]
6.68	WECR K-Means	278	×	Soft	Soft	—	2021	[324]
5.39	Semi-MultiCons	288	×	Soft	Soft	—	2022	[325]

Table 17: Feature table for the ECC - Constrained Consensus methods.

7.5.3 Full Constrained

These methods (in Table 18) include constraints in both the pool generation and the consensus steps. They make use of the formulas described before and combine them. On the one hand, SFS^{3}EC, ARSCE and RSEMICE make use of the subspace technique, although they differ in the consensus function. SFS^{3}EC merges partitions in the pool by building a hypergraph which takes partitions and constraints into account and running METIS over this graph to get the final partition. ARSCE computes the affinity graph for every solution in the pool, and uses regularized ensemble diffusion to fuse the similarity information. Finally, RSEMICE assigns a confidence factor to each solution in the pool to build a consensus matrix, which can be interpreted as a graph over which the NCut algorithm is applied (used as the consensus function). On the other hand, COP-SOM-E and Cop-EAC-SL use previous CC methods (ICOP-K-Means and COP-K-Means, respectively) to generate their pool. COP-SOM-E uses a hard constrained version of SOM as the consensus matrix, and Cop-EAC-SL runs the constrained single-link algorithm over a co-association matrix which counts how many times pairs of instances are placed in the same cluster in different partitions.

7.6 Metaheuristics-based CC

Metaheuristics-based CC (MbCC) use metaheuristic algorithms to approach the CC problem. Many distinctions can be made within the metaheuristic algorithms field, in this study the trajectory-based methods versus population-based methods is used to produce to subcategories of CC approaches, as it is the one which results in the more consistent dichotomy.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
6.36	Cop-EAC-SL	75	×	Soft	Soft	—	2009	[326]
3.00	En-Ant	133	×	Soft	Soft	Swarm Optimization	2012	[327]
7.54	COP-SOM-E	143	×	Hard	Hard	—	2012	[328]
10.77	RSEMICE	237	✓	Soft	Soft	—	2017	[329]
15.04	SFS^{3}EC	257	×	Soft	Soft	Non Graph-based	2019	[270]
6.86	ARSCE	263	×	Soft	Soft	—	2020	[330]

Table 18: Feature table for the ECC - Full Constrained methods.

7.6.1 Population-based

A plethora of metaheuristic methods has been applied to the CC problem. Particularly, population-based methods have shown remarkable success, with evolutive algorithms being the most used ones. A further distinction can be made within these methods: swarm optimization algorithms and genetic algorithms. In swarm optimization algorithms, a population of individuals is used to mimic the behavior of a colony of insects in its natural environment, while in genetic algorithms, the population is evolved according to the rules of natural selection, expecting them to generate the best possible individual (solution).

Swarm Optimization Table 19 gathers swarm optimization algorithms which tackle the CC problem. All of them are based on ant colonies behavior, with the main differences found in the scheme used to include constraints. MCLA, MELA and CELA are all based in the Leader Ant algorithm and use the same integration scheme. They modify the ant-nest assignment rule so that only feasible assignments are taken into account. MCLA and MELA ensure that they do not violate any ML constraints by using chunklets. They only differ in the type of constraints they can handle. CAC is based on the RWAC algorithm, which tries to simulate the behavior of ants in their environment trying to find a place to sleep. Constraints are included in this scheme by modifying attractive and repulsive forces between ants associated to constrained instances. The En-Ant algorithm uses three instances of the Semi-Ant algorithm as the partition generation algorithm in an ensemble setup. Constraints are used both in the generation of the partition pool and in the consensus function. Please note how none of these algorithms modify the fitness function of the ant colony algorithm on which they are based, instead they include constraints by modifying other aspects of the algorithms.

Genetic Algorithm Genetic strategies used to tackle the CC problems are presented in Table 20. A low-level dichotomy can be made within these methods, they can be either single-objective or multi-objective genetic algorithms. On the one hand, multi-objective algorithms optimize a set of fitness functions all at the same time. Constraints can be naturally included in this paradigm by simply adding the infeasibility as one of the functions to be optimized. This is the case for MOCK (which includes constraints into the classic PESA-II

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
6.26	MCLA	91	×	Hard	Hard	—	2009	[331]
12.66	MELA	92	×	Hard	—	—	2009	[331]
12.66	CELA	93	×	—	Hard	—	2009	[331]
7.71	CAC	132	×	Soft	Soft	—	2012	[206, 332]
3.00	En-Ant	133	×	Soft	Soft	Full Constrained	2012	[327]

Table 19: Feature table for the MbCC - Population-based - Swarm Optimization methods.

algorithm), PCS (which is based on NSGAII) and ME-MOEA/D_{CC} (which modifies classic MOEA/D). These three proposals also modify a basic aspect of the algorithm they are based on for it to fit better to the CC problem. MOCK implements a constraint-oriented initialization scheme. PCS features an improving procedure applied to the population after the classic operators have been applied. Lastly, ME-MOEA/D_{CC} uses memetic elitism with controlled feedback. On the other hand, single objective genetic algorithms usually optimize a combination of any classic clustering related measure and the infeasibility included as a penalty term. Methods such as COP-HGA, BRKGA+LS or SHADE$_{CC}$ use this strategy. Other proposals, such as Cop-CGA and FIECE-EM, evolve separate populations or subpopulations which have individuals with different solutions qualities and make them interact to generate new individuals. FIECE-EM+BFCU, FIECE-EM+FCU, FIECE-EM+DVO and FIECE-EM+LUC are all active variants of FIECE-EM.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
15.92	MOCK	30	×	Soft	Soft	—	2006	[333]
9.10	COP-CGA	62	×	Soft	Soft	—	2008	[334]
10.79	COP-HGA	65	✓	Soft	Soft	—	2008	[335]
4.31	PSC	126	×	Soft	Soft	—	2012	[336]
4.38	CEAC	212	×	Soft	Soft	—	2016	[337]
13.75	BRKGA+LS	235	✓	Soft	Soft	—	2017	[338]
13.97	FIECE-EM	247	×	Hard	Hard	Mixture Model-based CC	2018	[339]
15.56	FIECE-EM+BFCU	303	×	Hard	Hard	Active Clustering with Constraints & Mixture Model-based CC	2020	[305, 306]
15.56	FIECE-EM+FCU	304	×	Hard	Hard	Active Clustering with Constraints & Mixture Model-based CC	2020	[305, 306]
15.56	FIECE-EM+DVO	305	×	Hard	Hard	Active Clustering with Constraints & Mixture Model-based CC	2020	[305, 306]
15.56	FIECE-EM+LUC	306	×	Hard	Hard	Active Clustering with Constraints & Mixture Model-based CC	2020	[305, 306]
29.32	SHADE$_{CC}$	285	✓	Soft	Soft	—	2021	[340]
28.22	ME-MOEA/D_{CC}	286	✓	Soft	Soft	—	2021	[94]

Table 20: Feature table for the MbCC - Population-based - Genetic Algorithm methods.

7.6.2 Single Individual

Single individual methods focus on modifying and improving a single candidate solution. They start with a single individual which is improved with respect to the fitness function. Simulated annealing, local search, iterated local search or guided local search are examples of single solution metaheuristics. Table 21 lists methods which belong to this category.

CCLS and $DILS_{CC}$ use both variants of the classic LS algorithm to find solutions for the CC problem. They use a combination of the intra-cluster mean distance and the infeasibility to build their fitness function. The SemiSync algorithm is a nature-inspired non-evolutive based on regarding instances as a set of constrained phase oscillators, whose dynamics can be simulated to build a partition. The local interaction of every oscillator with respect to its neighborhood can be computed over time. Therefore similar instances will synchronize together in groups that can be interpreted as clusters. ML and CL are included by introducing an additional global interaction term.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
3.56	PAST-Toss	54	×	Soft	Soft	Graph-based	2008	[243]
3.99	CCLS	223	✓	Soft	Soft	—	2016	[341]
11.50	CG+PR+LS	236	×	Soft	Soft	Column Generation	2017	[338]
14.10	SemiSync	258	×	Soft	Soft	—	2019	[342]
20.09	$DILS_{CC}$	270	✓	Soft	Soft	—	2020	[95]

Table 21: Feature table for the MbCC - Single individual methods.

7.7 Multi-View CC

In many applications, data is collected from different sources in diverse domains, usually involving multiple feature collectors. This data refers to the same reality, although it exhibits heterogeneous properties, which translates into every instance being described by different sets of features. These are the called views, and the problem of performing clustering over instances described by different sets of features is known as multi-view (or multi-source) clustering [343]. In multi-view clustering, different sources of the data are used to produce a single partition. Constraints can be included into multi-view clustering in different ways and levels, giving place to Multi-View Constrained Clustering (MVCC). With respecto to the level in which constraints can be used, two options are available: intra-view constraints and inter-view constraints. Intra-view constraints relate instances which belong to the same view of the data (similarly to constraints in any non MVCC algorithm), while inter-view constraints relate instances which belong to different views, hence encouraging collaboration between the clustering processes applied to them.

7.7.1 Intra-View Constrained

Intra-view CC usually performs clustering separately in each view and then tries to find a consensus between the obtained partitions (similarly to what ensemble clustering does with the consensus function). Methods which belong to this category are gathered in Table 22. SMVC models clustering views via multivariate Bayesian mixture distributions located in subspace projections. It includes constraints in the Bayesian learning processes. TVClust and RDPM are both very particular methods, as they view the dataset and the constraint set

as different sources of information for the same data, thus performing multi-view clustering with only two views. The dataset is modeled by a Dirichlet Process Mixture model and the constraint set is modeled by a random graph. They aggregate information from the two views through a Bayesian framework and they reach a consensus about the cluster structure though a Gibbs sampler. MVMC independently builds a pairwise similarity matrix for every view and casts the clustering task into a matrix completion problem based on the constraints and the feature information from multiple views. The final pairwise similarity matrix is built iteratively by approaching the independent pairwise similarity matrices in different views to each other. The final partition is obtained by performing spectral clustering of the final similarity matrix. SSCARD is based on classic CARD, which is able to combine multiple weighted sources of relational information (some may be more relevant than others) to produce a partition of the dataset. SSCARD simply includes the PCCA penalty term (see Section 7.11) into the classic CARD objective function. Lastly, MVCC is the only intra-view CC method that performs clustering in the different views in a collaborative way. It performs constrained clustering in each view separately, inferring new constraints and transferring them between views using partial mapping. For constraint inference and transfer, a variant of the co-EM algorithm [344] is used, which is an iterative EM based algorithm that learns a model from multiple views of the data.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
7.09	SSCARD	49	✓	Soft	Soft	Fuzzy CC	2007	[345, 346]
3.45	MVCC	182	×	Soft	Soft	—	2014	[347]
3.62	SMVC	195	×	Soft	Soft	—	2014	[348]
19.21	TVClust	209	×	Soft	Soft	—	2015	[349]
20.02	RDPM	210	×	Soft	Soft	—	2015	[349]
11.04	MVMC	226	×	Soft	Soft	Matrix Completion & Spectral CC	2017	[350]
18.17	MVCSC	261	×	Soft	Soft	Non Graph-based	2019	[269]

Table 22: Feature table for the MVCC - Intra-View Constrained methods.

7.7.2 Inter-View Constrained

Inter-View CC methods can handle both intra-view and inter-view constraints. Methods which belong to this category are presented in Table 23. UCP uses the results of intra-view constraint propagation to adjust the similarity matrix of each view, and then performs inter-view constraint propagation with the adjusted similarity matrices. Its main drawback is that it is limited to two views. CMVNMF minimizes the loss function of NMF in each view, as well as the disagreement between each pair of views. The disagreement is defined as the difference between feature vectors associated to the same instance in the same view. It should be high if they are CL instances and low if they are ML instances. MSCP can propagate constraints across different data sources by dividing the problem into a series of two-source constraint propagation subproblems, which can be transformed into solving a Sylvester matrix equation, viewed as a generalization of the Lyapunov matrix equation. MCPCP uses a low-

rank relation matrix to represent the pairwise constraints between instances from different views. Afterwards it learns the full relation matrix by using a matrix completion algorithm and derives an indicator matrix from it with an iterative optimization process.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
5.27	UCP	167	×	Soft	Soft	Constraint Propagation	2013	[351]
4.03	MSCP	169	×	Soft	Soft	Constraint Propagation	2013	[319]
8.95	MCPCP	198	×	Soft	Soft	Matrix Completion	2015	[352]
2.85	CMVNMF	254	×	Soft	Soft	Non-negative Matrix Factorization CC & Active Clustering with Constraints	2018	[91, 271]

Table 23: Feature table for the MVCC - Inter-View Constrained methods.

7.8 Kernel CC

Kernel methods perform clustering by mapping the data from the original input space to a new feature space, which is usually of higher dimensionality. The key aspect of kernel-based methods is the avoidance of an explicit knowledge of the mapping function, which is achieved by computing dot products in the feature space via a kernel function. A critical aspect for kernel-based methods is the selection of the optimal kernel and its parameters. The basic classic kernel-based clustering method is the Kernel-K-Means algorithm. It performs clustering directly in the feature space by computing pairwise distances and updating centroids, using dot products and the kernel trick [241]. The goal of Kernel CC (KCC) is to learn a kernel function that maps ML instances close to each other in feature space, while mapping CL instances far apart, an then perform clustering in the feature space.

Table 24 shows a list of KCC methods. The first KCC method can be found in SSKK [353, 354], which is built on the basis of the HMRF-K-Means method. Kernel CC methods are all very similar to each other, with one of the few differences being the way in which they include the kernel parameters in clustering process. Some methods, such as ASSKK or SFFA include this parameter in the optimization process. Therefore they do not need to be specified by the user, not withstanding the higher computational cost. Additionally, other methods use more than one kernel, such as TRAGEK and ENPAKL. They learn multiple kernels which are later combined in a single one to obtain the global kernel matrix. Some minor differences can be found in ssFS, for example, which is specifically designed to cluster sets of graphs.

7.9 Fuzzy CC

Fuzzy classic clustering represents a hard dichotomy within the clustering area. In fuzzy clustering, instances are allowed to belong to more than one cluster, with a list of probabilities that indicate the likelihood of said instance to belong to every cluster. The set of these lists is called a fuzzy partition, in contrast to crisp (or hard) partitions obtained by non-fuzzy clustering algorithms, where instances are assumed to belong to a single cluster with a 100% probability [362]. Fuzzy clustering algorithms are usually applied over relational information, meaning that the feature vector describing the instances in the dataset are not needed, only the pairwise relations between them, which can be computed as pairwise dis-

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
20.98	SSKK	23	×	Soft	Soft	—	2005	[353, 354]
6.24	ASSKK	27	×	Soft	Soft	—	2006	[355]
9.90	BoostCluster	40	×	Soft	Soft	—	2007	[356]
15.81	CCSKL	89	×	Soft	Soft	Non Graph-based	2009	[198]
2.37	SFFA	125	×	Soft	Soft	—	2012	[357]
6.97	TRAGEK	141	×	Soft	Soft	—	2012	[358]
6.97	ENPAKL	142	×	Soft	Soft	—	2012	[358]
4.15	ssFS	291	✓	Soft	Soft	—	2012	[359]
5.82	SSKSRM	157	×	Hard	Hard	—	2013	[360]
5.82	SSSeKRM	158	×	Hard	Hard	—	2013	[360]
12.08	SKML	194	×	Soft	Soft	—	2014	[361]

Table 24: Feature table for KCC methods.

tances [363]. This makes the fuzzy clustering paradigm specially suitable to be extended in order to include constraints, as constraints are a natural type of relational information.

A list of Fuzzy CC (FCC) methods is proposed in Table 25. Even if this is one of the largest categories in clustering-based CC methods, authors have found that no further significant categorizations can be performed over it. The vast majority of methods in this category include constraints by means of a penalty term, which can be computed with different confidence degrees [125]. The confidence degree of the penalty refers to the number of possible assignations in which constraints violations are checked. Some methods examine all possible assignations for all constraints, such as SSCARD or AFCC, while others only examine the most probable assignation, like SSFCA, which is equivalent to perform violations checks in the crisp partition. Many methods are based on the PCCA method (see Section 7.11), such as AFCC, ACC, SS-CARD, PCsFCM and SS-CLAMP. These methods modify the objective function of PCCA to make it fuzzy or borrow its objective function to use it in a new optimization scheme. Besides, it is worth noting that some fuzzy methods are not relational but evidential. The evidential clustering framework is built around the concept of credal partition, which extends the concepts of crisp and fuzzy partitions and makes it possible to represent not only uncertainty, but also imprecision with respect to the class membership of an instance. Methods such as CECM, CEVCLUS and k-CEVCLUS include constraints into the evidential clustering paradigm, with CEVCLUS and k-CEVCLUS combining both relational en evidential fuzzy clustering features.

7.10 Mixture Model-based CC

Mixture models are parametric statistical models which assume that a dataset originates from a weighted sum of several statistical sources. These sources can typically be Gaussian distributions, originating the Gaussian Mixture Model (GMM). However, other statistical

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
7.09	SSCARD	49	✓	Soft	Soft	Intra-View Constrained	2007	[345, 346]
9.50	AFCC	59	×	Soft	Soft	Active Clustering with Constraints	2008	[297]
0.00	ACC	60	✓	Soft	Soft	—	2008	[144]
0.00	PCsFCM	78	×	Soft	Soft	—	2009	[364]
0.00	PCeFCM	79	×	Soft	Soft	—	2009	[364]
6.17	SCAP	124	✓	Soft	Soft	—	2011	[365]
4.82	SSFCA	134	×	Soft	Soft	—	2012	[125]
7.43	CECM	139	×	Soft	Soft	—	2012	[366]
6.41	SS-CLAMP	168	✓	Soft	Soft	—	2013	[367]
12.45	SS-FCC	163	✓	Soft	Soft	Co-Clustering	2013	[112]
13.78	SSeFCMCT	162	×	Soft	Soft	—	2013	[368]
13.78	SSsFCMCT	161	×	Soft	Soft	—	2013	[368]
17.64	PC-eFCM-NM	177	✓	Soft	Soft	—	2014	[232]
17.64	PC-sFCM-NM	178	✓	Soft	Soft	—	2014	[232]
5.81	CEVCLUS	192	×	Soft	Soft	—	2014	[369]
16.77	CMKIPCM	202	×	Soft	Soft	Active Clustering with Constraints	2015	[301]
6.38	SFFD	295	✓	Soft	Soft	—	2015	[370]
21.19	AAA	221	×	Soft	Soft	Active Clustering with Constraints	2016	[302]
6.84	k-CEVCLUS	245	×	Soft	Soft	—	2018	[89]

Table 25: Feature table for FCC methods.

distributions (like the Dirichlet distribution) can be used in the mixture models paradigm. In GMM-based clustering, each cluster is associated to a parameterized Gaussian distribution. These parameters are optimized for the final distribution to explain its associated cluster. The result of GMM-based clustering is not a crisp partition, but the probability for each instance to be generated by each of the available optimized Gaussian distributions. EM schemes are one of the most widely used methods to optimize the parameters of the distribution [371]. Table 26 gathers Mixture Model-based CC (MMbCC) methods. The most common way to include constraints in GMM-based clustering is by discarding unwanted distributions. For the case of hard CC, this is done by removing all distributions which violate any constraints from the addition of Gaussians, as in Constrained EM or DPMM. In the case of soft CC, the summation of distributions is modified to take these distributions into account to a higher or lower extent depending on the number of violated constraints and on the relevance of constraints themselves. This can be achieved by means of a penalty-style objective function, as in sRLe-GDM-FFS, MCGMM, SCGMM, or by assigning a level of confidence to each constraint, as in MAA or PPC. Some methods use different statistical models (not Gaussian), like DPMM, which uses Diritchlet Processes, or they allow a single cluster to be represented by more than one distribution, as in MCGMM. SPGP is not based on Gaussian mixture models but on Gaussian process classifiers (GPC). Given its similitude with GMM, authors have decided to include it in this category. The main difference between these two paradigms is that GMM are generative models, while GPC are discriminative models.

S_A	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
5.45	MAA	8	×	Hybrid	Hybrid	—	2004	[51]
17.70	Constrained EM	14	×	Hard	Hard	—	2004	[372]
4.67	PPC	21	×	Hybrid	Hybrid	—	2005	[373, 374]
20.58	MCGMM	24	×	Soft	Soft	—	2005	[204]
20.25	SCGMM	25	×	Soft	Soft	—	2005	[204]
9.01	SPGP	48	×	Hybrid	Hybrid	—	2007	[375]
1.45	CDPMM08	58	×	Hard	Hard	—	2008	[211]
7.30	sRLe-GDM-FFS	147	×	Soft	Soft	—	2012	[145]
4.71	C_4S	238	×	Soft	Soft	—	2017	[376]
13.97	FIECE-EM	247	×	Hard	Hard	Genetic Algorithm	2018	[339]
8.18	JDG	249	×	Soft	Soft	—	2018	[339]
15.56	FIECE-EM+BFCU	303	×	Hard	Hard	Active Clustering with Constraints & Genetic Algorithm	2020	[305, 306]
15.56	FIECE-EM+FCU	304	×	Hard	Hard	Active Clustering with Constraints & Genetic Algorithm	2020	[305, 306]
15.56	FIECE-EM+DVO	305	×	Hard	Hard	Active Clustering with Constraints & Genetic Algorithm	2020	[305, 306]
15.56	FIECE-EM+LUC	306	×	Hard	Hard	Active Clustering with Constraints & Genetic Algorithm	2020	[305, 306]

Table 26: Feature table for MMbCC methods.

7.11 Hierarchical CC

Details in classic hierarchical clustering and hierarchical CC have been already introduced in Section 3.1. Let us remember that hierarchical clustering methods produce a dendrogram, instead of a partition. Affinity criteria are used to determine cluster merges in every lever of the dendrogram.

Table 27 presents a list of Hierarchical CC (HCC) methods. Many strategies designed to include constraints into hierarchical clustering can be found in this category. Some on them modify the clustering engine of existing methods to include constraints in the process of selecting the clusters to merge, such as COP-COBWEB, C-DBSCAN, CAC1, Cons-DBSCAN or SDHCC. Others transform the dataset in some way for it to include the information contained in the constraint set, such as COBRA, COBRAS, C-DenStream. AHC-CTP includes constraints in the computation of the dissimilarities without using a penalty term, while methods such as 2SHACC and PCCA have to use one. The only divisive hierarchical CC method found by the authors is SDHCC, all of the rest perform hierarchical agglomerative CC.

7.12 Density-based CC

In density-based classic clustering, a cluster is considered to be a set of instances spread in the data space over a contiguous region with high density of instances. Density-based methods separate clusters by identifying regions in the input space with low density of instances, which are usually considered as noise or outliers [386].

Table 28 presents a list of Density-based CC (DbCC) methods. Two main strategies are used to include constraints into density-based clustering methods. The first one consists of modifying the assignation rule for instances to cluster, taking constraints into account, similarly to the way in which it is done in cluster engine-adapting methods. This strategy is used in

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
4.78	COP-COBWEB	1	×	Hard	Hard	—	2000	[85]
19.30	IDSSR	17	×	Hard	Hard	—	2005	[72, 82]
4.44	PCCA	22	×	Soft	Soft	—	2005	[377, 143]
3.87	C-DBSCAN	39	×	Hard	Hard	Density-based CC	2007	[378]
4.48	C-DenStream	80	×	Hard	Hard	Density-based CC & Online CC	2009	[379]
0.00	AHC-CTP	94	×	Soft	Soft	—	2010	[380, 381]
4.17	CAC1	113	×	Hard	Soft	Active Clustering with Constraints	2011	[140]
5.07	SDHCC	116	×	Hard	Soft	—	2011	[382]
3.44	AHCP	117	×	Hard	Soft	—	2011	[383]
3.29	SGID	121	×	Hard	Hard	SAT	2011	[384]
5.74	Cons-DBSCAN	137	×	Hard	Hard	Density-based CC & Active Constraint Acquisition	2012	[111]
3.57	Active-HACC	189	×	Soft	Soft	Active Clustering with Constraints	2014	[142]
4.20	COBRA	225	×	Soft	Soft	Active Clustering with Constraints	2017	[108]
19.57	COBRAS	244	×	Soft	Soft	Active Clustering with Constraints	2018	[304]
16.51	2SHACC	262	×	Soft	Soft	Constraint Propagation	2020	[385]
22.52	3SHACC	287	✓	Soft	Soft	Constrained Distance Transformation	2022	[190]

Table 27: Feature table for HCC methods.

methods like C-DBSCAN, C-DenStream, Cons-DBSCAN, SDenPeak or SSDC. On the other hand, some methods use constraints to redefine the density computation method to take them into account, such as in SemiDen or YZWD. In addition, there are methods that simply use constraints to modify the similarity measure or the dataset on the basis of the constraints, which run classic density-based clustering algorithm over them afterwards, such as SSDPC or fssDBSCAN.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
3.87	C-DBSCAN	39	×	Hard	Hard	Hierarchical CC	2007	[378]
4.48	C-DenStream	80	×	Hard	Hard	Hierarchical CC & Online CC	2009	[379]
5.74	Cons-DBSCAN	137	×	Hard	Hard	Hierarchical CC & Active Constraint Acquisition	2012	[111]
3.83	SDenPeak	213	×	Soft	Soft	—	2016	[387]
3.34	SemiDen	224	×	Hard	Hard	—	2017	[388]
4.33	YZWD	239	×	Soft	Soft	—	2017	[389]
5.37	SSDC	243	×	Soft	Hard	—	2018	[390]
15.94	SSDPC	272	×	Soft	Soft	—	2020	[391]
16.80	fssDBSCAN	280	×	Soft	Soft	Time Series	2021	[239]
17.22	ADPE	281	×	Soft	Soft	Active Clustering with Constraints & Constrained Pool Generation	2021	[307]
17.84	ADP	282	×	Soft	Soft	Active Clustering with Constraints	2021	[307]

Table 28: Feature table for DbCC methods.

7.13 Online CC

Online clustering methods perform clustering over data which varies over time. This is called a data stream. In classic online clustering, new data instances arrive (in the form of chunks or single instances) over time and the goal of the clustering algorithm is to produce

a partition of the current set of instances, usually taking into account information obtained from past instances. In online constrained clustering, not only new instances are provided to the method over time, but also constraints and, in some cases, only constraints. Methods performing Online CC (OCC) are gathered in Table 29.

The CME algorithm is an online wrapper for the COP-K-Means algorithms. It gradually forgets past constraints, lowering their effect on the current partition as new data from the data stream arrive. TDCK-Means is built on the basis of classic K-Means and also uses a decay term to handle online constraints. It addresses the temporal nature of the data by adapting the Euclidean distance to take into account both the distance in the multidimensional space and in the temporal space. A penalty term is then added to include constraints, which is more severe for instances closer in time and whose magnitude decays over time. CS2GS also uses constraint weight decay to perform online CC and is based on SOM. The architecture of the neural network used by CS2GS features two layers. The between-layer weights and the number of nodes of the first layer are adapted to dynamically correct the violation of constraints. Using the metrics included in these layers as a reference, the violation of constraints is quantified as network's error. Weights are modified over time based on the error to obtain the new weights. The weight update procedure tries to satisfy the currently violated constraints while keeping the new weights close to the old ones to avoid breaking the old constraints.

O-LCVQE and C-RPCL are both online competitive learners. Competitive learning algorithms are characterized by competition among k neurons, which compete to learn instances. This is known as the winner-take-all (WTA) approach. Competitive learning can be seen as performing cluster in the input space, viewing the neurons as centroids and using the Euclidean distance as the competition score. The O-LCVQE is a WTA approach that can only deal with CL constraints by defining the score as in LCVQE. Therefore the winner centroid is computed with regards to the objective function of LCVQE modified to consider only CL constraints. Similarly, the RPCL algorithm [392] can be modified to include only CL constraints within the WTA, resulting in C-RPCL. The intuition behind this modified algorithm is that, if a CL constraint gets violated by assigning the instance to a given centroid, C-RPCL searches for the nearest rival which does not cause any constraint violations. This nearest rival becomes the winner, and the previous winner prototype is moved away from that instance.

C-DenStream is based on the density-based clustering DenStream, which is the online version of DBSCAN. DenStream performs density-based clustering. it uses the micro-cluster density, which is based on weighting areas of instances in a neighborhood as a result of an exponential decay function over time. C-DenStream includes constraints into the DenStream clustering process by translating instance-level constraints into micro-cluster-level constraints using the micro-cluster membership of each instance in each timestamp. SemiStream builds an initial partition using MPCK-Means and updates it as new chunks of data arrive. The update process consist of performing clustering assigning pairs of constrained instances to clusters, minimizing the cost of said assignment. The SCSC algorithm is the only online cc algorithm that keeps the dataset constant over time and consider the

time dimension only over the constraint set. It consists of two main components, an offline procedure to build a convex hull, and an online procedure to update the clustering results when new pairwise constraints are received.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
0.35	CME	34	✓	Hard	Hard	—	2006	[393]
4.48	C-DenStream	80	×	Hard	Hard	Hierarchical CC & Density-based CC	2009	[379]
4.37	SemiStream	127	✓	Soft	Soft	Penalty-based Methods	2012	[205]
12.20	O-LCVQE	159	✓	—	Soft	Neural Networks-based CC	2013	[394]
12.20	C-RPCL	160	×	—	Soft	Neural Networks-based CC	2013	[394]
4.34	TDCK-Means	193	✓	Soft	Soft	Penalty-based Methods	2014	[234]
10.74	CS2GS	201	×	Soft	Soft	Self Organizing Maps-based CC	2015	[309]
5.48	SCSC	206	×	Soft	Soft	Constrained Pool Generation	2015	[320]

Table 29: Feature table for OCC methods.

7.14 Others

This section gathers minor CC categories. These categories, shown in Table 30, are considered to be minor because of the number of methods belonging to them (5 or less), or because of the restricted applicability or specificity of said methods. A total of 15 minor CC categories are briefly introduced, for a total of 40 methods.

7.14.1 Constrained Co-clustering

Co-clustering methods perform clustering on the column and the rows of a given dataset at the same time, considering them as closely related different sources of information. OSS-NMF extends the classic NMF by introducing constraints and performing clustering by solving a constrained optimization problem. This method is specifically proposed to solve the document clustering task, and does so by performing co-clustering in words and documents simultaneously, and considering both word-level and document-level constraints. RJFM is based on the meta-algorithm called Bregman Co-clustering, which can optimize a large class of objective functions belonging to the Bregman divergences. Its principle is simple: it alternatively refines row and column clusters, while optimizing an objective function that takes both partitions into account. It includes constraints in both columns and rows clustering the same way: it never performs rows or columns assignation breaking CL constraints, and always assigns full cliques of ML constraints. SS-NMF learns a new metric by applying simultaneously distance metric learning and modality selection. The new metric is used to derive distance matrices over which clustering is finally performed. SS-FCC formulates the CC problem as an optimization problem with an objective function built on the basis of the competitive agglomeration cost with fuzzy terms and constraint-based penalties. It introduces cooperation into the co-clustering process by including two fuzzy memberships, one of them related to columns and other related to rows, which are expected to be highly correlated. The amount of cooperation is delivered by the degree of aggregation, which should be maximized among clusters to accomplish the clustering task.

$S_{\mathcal{A}}$	Acronym	ID	Category	Penalty	ML	CL	Hybrid	Year	Ref.
5.49	SSAP	85	Affinity Propagation	×	Soft	Soft	—	2009	[395]
9.13	COALA	28	Alternative Clustering	×	Any	Any	—	2006	[207]
9.13	COALAcat	29	Alternative Clustering	×	Any	Any	—	2006	[207]
4.34	ADFT	67	Alternative Clustering	×	Soft	Soft	—	2008	[396]
8.56	ClusILC	46	Clustering Trees	×	Soft	Soft	—	2007	[397]
12.45	SS-FCC	163	Co-Clustering	✓	Soft	Soft	Fuzzy CC	2013	[112]
11.88	SS-NMF	109	Co-Clustering	×	Soft	Soft	Non-negative Matrix Factorization CC	2010	[274]
2.06	RJFM	97	Co-Clustering	×	Hard	Hard	—	2010	[398]
4.74	OSS-NMF	96	Co-Clustering	×	Soft	Soft	Non-negative Matrix Factorization CC	2010	[98]
5.08	CCG	175	Column Generation	×	Hard	Hard	—	2014	[61]
11.50	CG+PR+LS	236	Column Generation	×	Soft	Soft	Single Individual	2017	[338]
6.31	TKC(17)	229	Constraint Programming	×	Hard	Hard	—	2017	[399]
8.03	3CP	197	Constraint Programming	×	Hard	Hard	—	2015	[400]
2.38	BBMSC	205	Constraint Programming	✓	Soft	Soft	—	2015	[196]
15.45	CMSSCCP	199	Constraint Programming	×	Soft	Soft	—	2015	[401]
7.58	CCPP	264	Constraint Programming	×	Hard	Hard	—	2020	[42]
16.39	cut	84	Constraint Programming	×	Soft	Soft	—	2009	[402]
9.06	TKC	154	Constraint Programming	×	Hard	Hard	—	2013	[62]
4.12	YLYM	70	Feature Selection	×	Soft	Soft	—	2008	[403]
7.53	SCAN	259	HIN	×	Soft	Soft	—	2019	[404]
2.66	SCHAIN	230	HIN	×	Soft	Soft	—	2017	[405]
7.02	EICC	53	Incremental CC	×	Hard	Hard	—	2007	[406]
4.42	3SMIC	196	Information Maximization	×	Soft	Soft	—	2014	[400]
5.55	NLPPC	271	Label Propagation	✓	Soft	Soft	—	2020	[407]
4.88	PCKMMR	114	MapReduce	×	Soft	Soft	—	2011	[408]
11.04	MVMC	226	Matrix Completion	×	Soft	Soft	Intra-View Constrained & Spectral CC	2017	[350]
11.83	PMMC	140	Maximum Margin Clustering	✓	Soft	Soft	—	2012	[409]
17.11	CMMC	68	Maximum Margin Clustering	×	Soft	Soft	—	2008	[410]
13.16	TwoClaCMMC	171	Maximum Margin Clustering	×	Soft	Soft	—	2013	[411]
8.73	DCPR	222	Probabilistic Clustering	×	Hybrid	Hybrid	—	2016	[52]
7.14	RHWL	43	Probabilistic Clustering	×	Soft	Soft	Active Clustering with Constraints	2007	[96]
13.76	d-graph	250	Probabilistic Clustering	×	Soft	Soft	—	2018	[412]
2.98	JPBMS	130	SAT	×	Hard	Hard		2012	[413]
3.29	SGID	121	SAT	×	Hard	Hard	Hierarchical CC	2011	[384]
4.02	ISL	103	SAT	×	Hard	Hard	—	2010	[71]
3.20	JBMJ	228	SAT	×	Hard	Hard	—	2017	[414]
12.97	CPSC*-PS	151	Spatial CC	×	Soft	Soft	—	2012	[168]
12.97	CPSC*	150	Spatial CC	×	Soft	Soft	—	2012	[168]
3.54	CPSC	149	Spatial CC	×	Hard	Hard	—	2012	[168]
16.80	fssK-Means	279	Time Series	✓	Soft	Soft	Penalty-based Methods	2021	[239]
16.80	fssDBSCAN	280	Time Series	×	Soft	Soft	Density-based CC	2021	[239]

Table 30: Feature table for CC algorithms belonging to minor categories (Others).

7.14.2 Alternative Clustering based on constraints

Constrains can have multiple uses, other than serving as hints for the clustering process. In alternative clustering, constraints are used to produce different partitions of a single datasets. Alternative clustering methods are not strictly CC methods, as they do not use constraints generated from a side source of information (an oracle), but from the current state of a partition in an iterative clustering process. The COALA takes a partition of a dataset as input, and aims to find a different high-quality partition using constraints. In order to make the

obtained partition different from the one provided, CL constraints are created between the instances assigned to the same cluster in the original partition. COALA applies agglomerative hierarchical clustering considering two possible merges in each step, one involves the two closest instances and the other involves the two closest instances that do not violate a CL constraint. Which merge is performed depends on a parameter controlling the trade-off between quality and dissimilarity (with respect to the base partition) of the new partition. COALAcat is the categorical version of COALA. Contrary to COALA, ADFT does not take a partition of a dataset as input. The overall ADFT method can be summarized in five steps. In the first step, a classic clustering method like K-Means is applied to partition the dataset. The second step characterizes this partition by means of ML and CL constraints and obtains a new distance metric based on them (using the CSI method). In the third step, this distance metric is taken as basis to compute an alternative distance measure, by obtaining singular values decomposition of the matrix that defines the metric is obtained, and by computing the Moore-Penrose pseudo-inverse of the stretcher matrix. This effectively flips the stretching and compressing dimensions. After that, the metric matrix is recomposed, multiplying its decomposition. In the fourth step, the newly learned metric is applied on the dataset to transform it, and in the fifth step, the classic clustering algorithm run to obtain the original partition is re-run to obtain a new (and different) one.

7.14.3 CC based on clustering trees

Methods which belong to this category perform clustering by using decision trees. ClusILC uses the Top-Down Induction (TDI) approach to build the clustering tree. While most TDI approaches are based on heuristics local to the node that is being built, ClusILC employs a global heuristic which measures the quality of the entire tree and which takes all instances of the dataset into account. ClusILC's heuristic measures the average variance in the leafs of the tree (normalized by the overall dataset variance) and the proportion of overall violated constraints. ClusILC greedily searches for a tree that minimizes this heuristic using an iterative process that refines the current tree in every iteration by replacing one of its leaves with a subtree consisting of a new test node and two new leaves. This subtree is selected among a set of candidates generated procedurally based on the heuristic described above.

7.14.4 Maximum Margin CC

Maximum Margin Clustering (MMC) uses the maximum margin principle adopted in the supervised learning paradigm. It tries to find the hyperplanes that partition the data into different clusters with the largest margins between them. CMMC includes constraints into MMC by adding concave-convex restrictions to the original MMC optimization problem. These restrictions behave as penalties for constraint violation. PMMC introduces a set of loss functions, featuring a strong penalty to partitions violating constraints while operating under the maximum margin principle at the same time. In order to do so, the classical definition of score used in MMC to determine cluster membership is modified to include high penalties for assignations violating constraints. Both CMMC and PMMC use the constrained concave-convex procedure (proposed in [409]) to solve the non-convex optimiza-

tion problem they set to address the CC problem. TwoClaCMMC is proposed to overcome some shortcomings of the CMMC algorithm, although it is limited to two-class problems. It modifies the MMC objetive function with a penalty term which accounts for constraints violations. TwoClaCMMC differs from CMMC in the formulation of the penalty term. In TwoClaCMMC, the position of the hyperplane with respect to the instances involved in a violated constraint is taken into account, weighting the cost of violating such constraints with respect to said position.

7.14.5 Feature Selection

Clustering can be used to perform feature selection over high-dimensionality datasets. YLYM is a clustering method designed to perform feature selection making use of a constraint set. It is composed of three steps. The first two steps consist of the classic expectation and maximization steps from an EM optimization scheme, in which feature saliencies are computed in a completely unsupervised way. The third step is called the tuning step (T-step), which refines saliencies to minimize the feature-wise constraint violation measure, which is computed based on the Jensen-Shannon divergence. The three steps (expectation, maximization and tuning) are performed iteratively until they reach convergence. The proposed method outputs a partition of the dataset and the saliency of every feature.

7.14.6 CC through MapReduce

The MapReduce paradigm is used to address problems in the context of Big Data. The MapReduce paradigm consists of dividing the computational load associated with processing a dataset among multiple processing nodes, in order to decrease the time required to obtain results. MapReduce is based on two operations: (1) The Map operation processes inputs in the form of key-value pairs and generates intermediate key/value pairs received by the Reduce operation (2) The Reduce operation processes all intermediate values associated with the same intermediate key generated by Map. PCKMMR constitutes the first MapReduce approach to CC. It applies the MapReduce approach on the COP-K-Means algorithm. In order to do so, authors propose a Map function which calculates the distance of each instance to each centroid and assigns it to the one that minimizes this measure and does not violate any constraint. To avoid interdependence between mappers, the constraints are generated locally to each mapper in each call. The map function returns the centroid/instance pair. The Reduce operation takes as input all instances associated to a centroid and updates the value for that centroid by computing the average of those instances.

7.14.7 SAT-based CC

SAT-based CC approaches formulate the CC problems in terms of logical clauses in conjunctive normal form. They apply general SAT solvers to find a solution, which can find solutions for any problems formulated as a satisfiability problem. This approaches to CC problem can include hard constraints in a very natural way, as well as they usually can handle many types of the constraints described in Section 2. ISL is limited to clauses of 2 literals. Several types of problems within the CC framework can be expressed in the form of sets

of formulas in closed normal form (CNF), implying that they can be approached with ISL and therefore solved optimally. It is worth noting that ISL is limited to two-class problems (k=2), although it can find a solution to these problems in polynomial times (if such solutions exists). SGID performs agglomerative hierarchical clustering given a dataset and a set of constraints formulated in terms of logical clauses in its Horn's normal form. In order to produce a dendrogram, the clauses modeling its properties must also be given to SGID, which allows them to vary in the features of the dendrogram it produces. In JPBMS the declarative modeling principle of constrained programming is used to define a CC problem taking into account the constraint set, the description of the clusters and the clustering process itself. Traditionally, clustering algorithms proceed by iteratively refining queries until a satisfactory solution is found. JPBMS includes the stepwise refinement process in a natural way to focus on more interesting clustering solutions. JBMJ performs CC under the correlational clustering paradigm, where a labeled weighted undirected graph is given to perform clustering. The objective function of correlation clustering clusters the nodes of the graph in a way that minimizes the number of positive edges between different clusters and negative edges within clusters. JBMF formulates this problems in terms of clause satisfiability and applies MaxSAT to obtain an optimal solution. It is able to include several types of constraints by modifying the graph structure and applying specific SAT-translation procedure for some of them. Particularly, instance-level constraints are included by just setting the weight of the edges which connect ML related instances to ∞ and CL related instances to $-\infty$, there is no need to generate specific clauses.

7.14.8 CC through constraint programming

The CC problem can be addressed from the constraint programming (CP) point of view when the classic requirements of a clustering problem formulated as restrictions for a CP are extended with the restrictions regarding ML and CL constraints. TCK does this exactly: it models the classic clustering problem requirements, the constraints, and the clustering optimization criteria, including the within-cluster sums of squares (WCSS), as constraints to be solved by a general constraint programming solver. 3CP extends TCK in the sense that it is more general and it does not need the number of clusters to be specified, only the boundaries of the interval it lies in. Additionally, 3CP is capable of optimizing more than one clustering criteria at the same time, finding the minimal set of nondominated Pareto solutions. TKC(17) also extends TCK, differing from it in two key aspects. Firstly, It does not need the number of clusters to be specified, only bounds need to be given (as in 3CP). Secondly, three optimization criteria are modeled as CP constraints in TKC(17): minimizing the maximal diameter, maximizing the split between clusters, and minimizing WCSS. Besides, CMSSCCP optimizes the WCSS, but it does so via a global optimization constraint. A lower bound for this criterion is computed using dynamic programming, and a filtering algorithm is proposed to filter objective variables as well as decision variables. As usual, instance-level constraints are modeled in terms of constraint programming, along with the classic clustering problem requirements. JPBMS (also in Section 7.14.7) uses the declarative modeling principles of CP to define the CC problem as a SAT problem, which can be solved with a general SAT solver. BBMSC formulates the CC problem in terms of an integer pro-

gramming problem, rather than in terms of a constraint programming problem (which can be considered as a subtype of the former). It employs the same procedures as in constraint programming, incorporating ML and CL constraints via a weighted penalty term added to the base classic clustering restrictions, alongside with a regularization term to favor smoothness.

7.14.9 CC through Column Generation

In Column Generation (CG), the minimum sum-of-squares (MSS) problem for clustering is solved optimally. This is done by formulating the problem in terms of an integer linear programming problem. In it, a boolean matrix encoding all possible partitions in its columns is explored to find an optimal solution with respect to a cost function (the MSS in this case) that is applied to the boolean matrix by columns. In practice, the boolean matrix is too large to be computed, therefore it is incrementally built when searching for the optimal solution. The column generation approach derives a master problem from a reduced set of restrictions and iterates between two steps: solving the master problem and adding one or multiple candidate columns to the boolean matrix. A column is a candidate to be included in the restricted master problem if its addition improves the objective function. If no such column can be found, one is certain that the optimal solution of the restricted master problem is also the optimal solution of the full master problem. CCG includes constraints into this framework by modifying the MSS formulation. This way a new restriction which enforces all constraints to be satisfied is added to is classic form. Effectively, this is translated into the clustering process by removing all partitions violating any amount of constraints from the boolean matrix. This way, they are discarded from candidate solutions. The CG+PR+LS solves CC similarly to CCG, however it includes two extra steps: path-relinking algorithms are used to intensify and diversify the search in a group of solutions, and a LS procedure is used to locally improve the final solution.

7.14.10 Information-maximization CC

Information-maximization clustering techniques address the lack of objective model selection and parameter optimization strategies from which most other clustering techniques suffer. In it, a probabilistic classifier is learned so that some information measure between instances and clusters assignments is maximized. 3SMIC includes constraints into the information-maximization clustering algorithm SMIC. SMIC tries to learn the class-posterior probability in an unsupervised manner so that the mutual information between instances and their class labels (in the final partition) are maximized. Constraints are included into SMIC by modifying the SMI approximator, so that the inner product of the probabilities of constrained instances which belong to the same cluster is maximized in the case of ML and minimized in the case of CL. Furthermore 3SMIC includes a procedure to apply the transitive property of ML efficiently.

7.14.11 CC through Heterogeneous Information Networks

Heterogeneous Information Networks (HINs) are graphs which model real world entities and their relationships with objects and links, where objects can be of different types and whose links represent different kinds of relationships. HINs in which objects are described by attributes (features) are called attributed HIN or AHIN abbreviated. The challenge in AHINs is to perform clustering based not only on attribute similarity, but also based on link similarity. The former can be measured with a conventional distance measure, while the latter is measured with graph-oriented distance measures, such as shortest-path length and random-walk-based, although meta-path are commonly used as well. A meta-path is a sequence of node types that expresses a relation between two objects in an AHIN. SCAN includes constraints in AHIN by means of a penalty term in its similarity function. It computes the similarity of every node pair based on their attribute similarity and the connectedness of the nodes network. The former is obtained by an attribute similarity measure which considers coupling relationship among attributes, while the latter is derived based on the meta-paths connecting the object pair. This similarity is then penalized proportionally to the number of violated constraints. SCHAIN includes constraints in AHINs by first composing a similarity matrix that measures the similarity of every object pair based on the attribute similarity and the network connectedness. SCHAIN assigns a weight to each object attribute and meta-path in composing the similarity matrix. To take constraints into account, SCHAIN uses a penalty function which involves the generated weights and cluster memberships. It employs an iterative, staggered 2-step learning process to determine the optimal weights and cluster assignment as output.

7.14.12 Incremental CC

In incremental CC, a fixed set of instances and a variable constraint set are provided to perform clustering. Modifications over the constraint sets are given to incremental CC algorithms over time. These modifications include the addition and removal of constraints. The goal of Incremental CC is to efficiently update a the current partition, without running the base clustering algorithm used to generate the initial partition. EICC is the only proposal belonging to this category. This method takes as input a single constraint at a time, and depending on the properties of the constraint, it will attempt to greedily optimize a given (not specified) objective function. If the constraint does not result in a significant improvement in the objective function, it is passed over and a new one is chosen (by the user). This algorithm only works when a set of preconditions are met, otherwise it will not produce a partition of the dataset, therefore it cannot be used to address a general constrained clustering problem.

7.14.13 CC through affinity propagation

In Affinity Propagation (AP) a binary grid factor-graph is used to perform clustering and to determine the most representative instances, which are called exemplars and can be compared with the notion of centroid. In order to do so, a number of hidden variables equal to the number of pairwise dissimilarities is defined. Afterwards, an iterative procedure is per-

formed over the hidden variables, updating their value on the basis of their neighborhood variables and hyperparameters. The final value of the hidden variables determines the exemplar instances and the membership of the rest of instances with respect to the exemplars. SSAP includes constraints in AP by introducing a fictitious meta-points for every chunklet in the transitive closure and for every CL instance which is not part of a chunklet. The meta-points allow explicitly enforcing ML constraints and CL constraints, while they also propagate them.

7.14.14 Spatial CC

Spatial clustering is a variant of classical clustering in which instances are not points, but polygons. Classic clustering methods do not work well when applied to spatial clustering because they represent the polygons as points which summarize their features; which are not sufficiently representative of the polygons to obtain a good result. To overcome this problem, the CPSC algorithm is proposed, a spatial clustering algorithm based on the A* algorithm which is able to consider both instance-level constraints (ML and CL) and cluster-level constraints. In order to include the constraints into the clustering process, the heuristic function used by the A* algorithm is designed based on them. CPSC starts by selecting k seeds from the data set (polygons), which will be the initial clusters and will grow through the iterative process. The seeds must be selected in such a way that each of them violates all ML constraints with respect to other seeds, thus ensuring that they will not be grouped in the same cluster. In addition to this, the seeds must satisfy all CL constraints between them. The best k seeds are selected among those that meet these conditions. After that, the A* algorithm starts. The seeds are considered as the initial state, and the target clusters as the goal state. Each cluster is increased by adding polygons to its initial state one by one until it reaches its goal state. At each iteration, the best cluster (with respect to the heuristic) to be augmented and the best polygon to be augmented are selected. This is done to ensure that all clusters grow in parallel and not sequentially, which would affect compactness. The process continues until all polygons have been assigned to a cluster or until a deadlock state is reached, which can occur when two clusters compete for the same polygon and in which case there may be polygons which are not assigned to any cluster in the final partition. To overcome this problem, two other algorithms are proposed: CPSC*, which allows the user to relax the constraints to ensure that all polygons are assigned to a cluster, ensuring convergence, and CPSC*-PS (polygon split), which also allows polygons to be split when strictly necessary.

7.14.15 Probabilistic CC

In probabilistic clustering a probabilistic model is used to describe relationships between instances and their cluster memberships. Constraints are included into this framework by also describing them in terms of probabilities. RHWL is an active CC method which uses a basic probabilistic CC procedure as the clustering algorithm. It takes advantage of the probabilities computed by this procedure to later use them to decide the pair of instances to query to the oracle. DCPR uses an objective function maximizing the likelihood of the observed constraint labels composed of two terms: (1) the first is the conditional entropy of

the empirical cluster label distribution, which is maximized when the cluster labels are uniformly distributed (the clusters are balanced). (2) the second term is the conditional entropy of instance cluster labels for the unlabeled instances, which is minimized when the formed clusters have large separation margin and high confidence for the cluster memberships of unconstrained instances. A variational EM optimization scheme is used to optimized the proposed objective function.

7.15 Constrained Distance Transformation

Distance transformation methods usually take a standard distance measure as their basis (like as the Euclidean distance) and parameterize it. By modifying this parameters, the distance measure is transformed for it to be adapted to the data and the constraints. Generally, the goal of these methods is to learn a new distance metric bringing ML instances together and setting CL instances apart. Constrained Distance Transformation (CDT) methods are presented in Table 31.

Methods such as CSI and Xiang's learn the weights of a matrix by parameterizing a family of Mahalanobis distances. RCA also learns a Mahalanobis metric. In order to do so it changes the feature space used for data representation by assigning high weights to relevant dimensions and low weights to irrelevant dimensions by means of a global linear transformation. The relevant dimensions are estimated using chunklets. ERCA extends RCA to include CL. It does so by computing a matrix that optimizes the between-class scatter and combining it with the matrix optimizing the within-class scatter. DCA is another extension over RCA to include CL. It does so by looking for a linear transformation which results in an optimal distance metric by maximizing the variance between chunklets and minimizing the variance between instances in the same chunklet. KDCA uses the kernel trick to learn a nonlinear metric distance under the same principles of DCA. MSSB is a modification over RCA that uses a data-dependent regularizer term to avoid the drawbacks that weighting discrimination brings to RCA. From all methods using Mahalanobis distances parameterization, the ITML and A-ITML-K-Means approaches are the only ones using Information Theoretic Metric Learning as its base framework. ITML learns a constrained distance metric by learning a positive-definite matrix that parameterizes a Mahalanobis distance. This matrix is regularized to be as close as possible to a given Mahalanobis distance function, parameterized by another auxiliary matrix. The distance between these two matrices can be quantified via an information-theoretic approach, so it can be computed as the relative entropy between their corresponding multivariate Gaussians. Instance-level constraints are included as linear constraints to the optimization problem, which results in a particular case of the Bregman divergence. Therefore, it can be optimized by the Bregman's method. A-ITML-K-Means is an active clustering with constraints setup for the ITML approach that uses K-means as its clustering algorithm.

It uses a reduced set of selected constraints to perform ITML and applies classic clustering (K-Means) with the newly learned metric in an active clustering setup. ITML learns a Mahalanobis distance metric under a given set of constraints and instances by minimizing the

LogDet divergence between the original distance matrix and an objective distance matrix that in built on the basis of constraints.

HMRF-K-Means approaches the CC problem from a hybrid probabilistic framework based on Hidden Markov Random Fields (HMRF), which is developed to find an EM-optimizable objective function derived from the posterior energy, which is defined by the HMRF. This objective function combines an adaptive distance measure, such as the Bregman divergence or directional similarity measures, and a constraint-violation penalty term. The later is controlled by a scaling function that assign more relevance to ML constraints relating distant instances and CL constraints relating close instances. Comraf uses a combinatorial MRF (an MRF in which at least one node is a combinatorial random variable). The Comraf model can be applied to classic clustering by searching for cliques in the Comraf graph and using the mutual information as a potential function. This graph is built on the basis of the interactions between the combinatorial random variables. Comraf can be extended to constrained clustering by incorporating weighted constraints as a penalty term in the objective function that Comraf optimizes.

Other methods such as MPCK-Means iteratively compute cluster-local weights for every feature, effectively creating a new distance metric for every cluster, which can be applied and updated during the clustering process in an EM scheme. LLMA also performs metric learning through locally linear transformations, achieving global consistency via interactions between adjacent local neighborhoods.

SMR builds the graph Laplacian matrix based on pairwise similarities. This matrix is then used to regularize a non-parametric kernel learning procedure in which the learned kernel matrix is forced to be consistent with both pairwise similarities and the constraint sets by minimizing the regularizer that is based on the Laplacian graph. MSBSBS is similar to SMR. However it uses the Karush-Kuhn-Tucker conditions to learn the non-parametric kernel. MSBSBS(10) is an improvement over MSBSBS which combines both the constraint set and the topological structure of the data to learn a non-linear metric. CDJPBY uses a weighted sum to combine multiple kernels. It includes an optimization criterion that allows it to automatically estimate the optimal parameter of the composite Gaussian kernels directly from the data and the constraints.

RDF transforms the metric learning problem into a binary class classification problem and employs random forests as the underlying representation. Constraints are included by replacing the original distance function with a feature map function which transforms each constrained instance, changing their location to reflect the information contained in the constraint set. The transformed data is used as training data for a random forest that evaluates each instance pair. Each tree from the random forest independently classifies the pair as similar or dissimilar, based on the leaf node at which the instance-pair arrives. SSMMHF uses a random forest-based strategy as well. It first builds a model of the data by computing a forest of semi-random cluster hierarchies. Each tree is generated applying a semi-randomized binary semi-supervised maximum-margin clustering (MCC) algorithm iteratively. This way, each tree encodes a particular model of the full semantic structure of the data, so the full structure of the tree can be considered as a weak metric. A final metric model can be pro-

duced by merging the output of the forest described above. Constraints are included by modifying the MMC procedure, producing Semi-Supervised MMC. This results in a method that seeks to simultaneously maximize the cluster assignment margin of each point (as in unsupervised MMC) and an additional set of margin terms reflecting the satisfaction of each pairwise constraint.

SCKMM firstly estimates the optimal value for the parameter of a Gaussian kernel by ascending gradient and obtains an initial partition using PCBKM, which is a modification of the classic K-Means algorithm to include constraints. After this, a distance measure is obtained using the assignments of the last partition, which is used by PCBKM to produce a new partition. SCKMM iterates between steps two and three until it converges.

3SHACC includes a metric learning step in 2SHACC. In first place, it determines the relevance of every constraint in a completely unsupervised manner. Said relevances are used as constraint weights by the Weighted-Learning from Side Information (WLSI) DML method, which is a weighted version of CSI.

7.16 Distance Matrix Modification

In Distance Matrix Modification (DMM) methods, the CC problem is approached from the DML point of view. Nevertheless these methods work directly with the distance matrix. Their goal is to modify the entries of the distance matrix for it to reflect the information contained in the constraint set, once again resulting in ML instances being brought closer together and CL instances being set apart. These methods do not provide neither a new distance metric nor a new data space, although these two can be obtained from said distance matrix with classic DML techniques.

7.16.1 Constraint Propagation

In constraint propagation methods, entries in the distance matrix which correspond to constrained instances are usually first modified. Then, those changes are propagated to the rest of the matrix to a variable extent and using different strategies. Table 32 gathers methods which use constraint propagation to perform CC.

The most simple approach in constraint propagation consists of simply setting distances between ML instances to 0 and to a high value for CL instances in the distance matrix. Afterwards, the all-pairs-shortest-path algorithm is run to propagate the changes to the rest of entries. Methods such as CCL, 2SHACC and CAF use this approach.

E^2CP propagates constraints in the distance matrix by taking each of its columns as the initial configuration of a two-class semi-supervised learning problem with respect to the instance associated to the column. The positive class contains the examples which should appear in the same cluster as the instance associated to the column, and the negative class contains the examples that should not. In this way, the constraint propagation problem can be decomposed into a number of subproblems equal to the size of the dataset. After that, the same process is repeated, this time taking rows instead of columns. MSCP extends

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
19.17	CSI	5	×	Soft	Soft	—	2002	[88]
20.61	MPCK-Means	11	✓	Soft	Soft	Penalty-based Methods	2003	[87, 228]
8.97	HMRF-K-Means	12	✓	Soft	Soft	Penalty-based Methods	2004	[194]
15.99	DistBoost	13	×	Soft	Soft	—	2004	[415]
13.18	LLMA	15	×	Soft	Soft	—	2004	[416]
16.01	RCA	26	×	Soft	—	Constrained Data Space Transformation	2005	[187]
26.59	ERCA	31	×	Soft	Soft	—	2006	[417]
19.24	DCA	32	×	Soft	Soft	—	2006	[418]
16.08	KDCA	33	×	Soft	Soft	—	2006	[418]
5.68	Comraf	35	—	Soft	Soft	—	2006	[419]
19.96	DYYHC	47	×	Soft	—	—	2007	[420]
6.62	ITML	50	×	Soft	Soft	—	2007	[421]
11.01	SMR	51	×	Soft	Soft	—	2007	[422]
15.05	Xiang's	74	×	Soft	Soft	—	2008	[423]
11.82	MSSB	83	×	Soft	Soft	—	2009	[424]
16.97	MSBSBS	100	×	Soft	Soft	—	2010	[425]
6.51	SCKMM	101	×	Soft	Soft	Cluster Engine-adapting Methods	2010	[221]
17.78	MSBSBS(10)	107	×	Soft	Soft	—	2010	[426]
5.01	LRML	108	×	Soft	Soft	—	2010	[141]
18.63	LRKL	118	×	Soft	Soft	—	2011	[427]
7.28	CDJPBY	119	×	Soft	Soft	—	2011	[241]
7.01	SNN	136	×	Soft	Soft	Classic Neural Network-based CC	2012	[312]
7.83	RFD	145	×	Soft	Soft	—	2012	[428]
5.62	A-ITML-K-Means	156	×	Soft	Soft	Active Clustering with Constraints	2013	[299]
7.07	LSCP	204	×	Soft	Soft	Constraint Propagation	2015	[429]
10.40	SSMMHF	216	×	Soft	Soft	—	2016	[430]
15.34	AMH-L	266	×	Soft	Soft	—	2020	[431]
15.34	AMH-NL	267	×	Soft	Soft	—	2020	[431]
22.52	3SHACC	287	×	Soft	Soft	Hierarchical CC	2022	[190]

Table 31: Feature table for CDT methods.

E^2CP to consider multi-source data. It decomposes the problem into a series of two-source constraint propagation subproblems, which can be transformed into solving a Sylvester matrix equation, viewed as a generalization of the Lyapunov matrix equation. SRCP also uses decomposition to propagate constraints, although it decomposes the problem taking into account the full dataset, not only columns or rows. It exploits the symmetric structure of pairwise constraints to develop a constraint propagation approach based on symmetric graph regularization. MMCP propagates constraints in rows and columns separately without decomposition using multi-graph propagation methods.

LCP propagates the influence of the constraints to the unconstrained instances in the dataset proportionally to their similarity with the constrained data. LCP first determines the proportion in which constrained instances influence unconstrained instances using a previously

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
19.16	CCL	4	×	Soft	Soft	—	2002	[432]
12.47	LCPN	61	×	Soft	Soft	Non Graph-based	2008	[252]
8.22	Lo-NC	82	×	Soft	Soft	—	2009	[433]
11.96	E^2CP	95	×	Soft	Soft	—	2010	[434, 319]
5.32	SRCP	120	×	Soft	Soft	—	2011	[435]
6.94	MMCP	122	×	Soft	Soft	—	2011	[436]
5.74	LCP	128	×	Soft	Soft	—	2012	[437]
4.63	SSL-EC	131	×	Soft	Soft	Active Constraint Acquisition	2012	[285]
5.27	UCP	167	×	Soft	Soft	Inter-View Constrained	2013	[351]
4.03	MSCP	169	×	Soft	Soft	Inter-View Constrained	2013	[319]
5.62	A-ITML-K-Means	156	×	Soft	Soft	Active Clustering with Constraints	2013	[299]
10.31	CAF	166	×	Soft	Soft	Constrained Data Space Transformation	2013	[438]
4.04	ACC(14)	179	×	Soft	Soft	—	2014	[439]
7.07	LSCP	204	×	Soft	Soft	Constrained Distance Transformation	2015	[429]
4.45	CPSNMF	211	×	Soft	Soft	Non-negative Matrix Factorization CC	2016	[276]
23.05	ISSCE	214	×	Soft	Soft	Constrained Pool Generation	2016	[321]
20.25	RSSCE	215	×	Soft	Soft	Constrained Pool Generation	2016	[321]
7.73	C^3	217	×	Soft	Soft	—	2016	[440]
7.83	CESCP	252	×	Soft	Soft	Constrained Pool Generation	2018	[323]
10.64	DCECP	253	×	Soft	Soft	Constrained Pool Generation	2018	[323]
5.21	PCPDAMR	273	×	Soft	Soft	—	2020	[441]
16.51	2SHACC	262	×	Soft	Soft	Hierarchical CC	2020	[385]
4.16	ILMCP	277	×	Soft	Soft	—	2021	[442]

Table 32: Feature table for DMM - Constraint Propagation methods.

proposed label propagation procedure. Then, intermediate structures, called constrained communities, are defined to include the factional instances that are affected by a constrained instance (including itself). These structures are used to find the range of influence of each constraints without any parameter estimation. ACC(14) performs constraint propagation the same way as LCP, the only difference being that ACC(14) allows overlapping between constrained communities. C^3 propagates constraints proportionally in constrained communities as well, although directly performing clustering on them through their indicator matrix.

PCPDAMR uses both the similarity and the dissimilarity (distance) matrix to perform constraint propagation. It does so to emphasize the difference between ML and CL constraints, which are usually encoded in the same matrix. The constraint propagation is carried out via manifold embedding, in which the inherent manifold structure among the data instances is mapped to their similarity/dissimilarity codings. A regularization term to consider adversarial relations between the two matrices is used to enhance the discriminability of propagated constraints.

Many methods simply use the E^{2}CP as an intermediate step performed between other CC-related operations. For example, RSSCE, CESCP and DCECP are all ensemble clustering method which use E^{2}CP as the CC method to generate different partitions using the random subspace technique. CPSNMF simply performs E^{2}CP and a penalty-based version of NMF afterwards.

More exotic approaches can be found in LCPN or Lo-NC. LCPN takes the affinity matrix as the covariance matrix of a parameterized Gaussian process with mean 0, effectively connecting the spatial locations of constrained instances and propagating a positive or negative affinity value (depending on whether it is a ML or CL constraint) in those locations. Lo-NC performs a space-level generalization of pairwise constraints by locally propagating the information contained in the constraint set.

7.16.2 Matrix Completion

Matrix completion techniques, gathered in Table 33, are used to fill gaps in relational matrices. Its key concept is found in how to build the matrix over which matrix completion is applied. The reason is that matrix completion algorithm itself does not need to be specifically designed for CC, but it only need a matrix with gaps to be filled, so low-rank matrices are preferred.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
6.79	MCCC	165	×	Soft	Soft	—	2013	[443]
4.46	STSC	188	×	Soft	Soft	Non Graph-based	2014	[265]
8.95	MCPCP	198	×	Soft	Soft	Inter-View Constrained	2015	[352]
3.49	LMRPCP	208	×	Soft	Soft	—	2015	[444]
11.04	MVMC	226	×	Soft	Soft	Intra-View Constrained & Spectral CC	2017	[350]

Table 33: Feature table for DMM - Matrix Completion methods.

In CC, the constraint matrix is usually a low-rank matrix, perfectly suitable for constraint propagation. STSC takes advantage of this and proposes the first approach to CC from the Self-taught learning paradigm, where side information is generated by the same algorithm that will make use of it later without the need of an oracle (or human). STSC can augment the set of constraints taking advantage of the low-rank nature of the constraint matrix via matrix completion. Matrix completion methods are able to recover low-rank matrices with high probability by using only a small number of observed entries. STSC performs self-taught constraint augmentation and constrained spectral clustering in an iterative manner. Constraint augmentation is performed via matrix completion over a combination of the low-rank constraint matrix and the affinities in the affinity graph. MCPCP also performs matrix completion over the constraint matrix, with each entry in the matrix being a real number that represents the relevance of the two corresponding instances. MCPCP aims to learn the full relation matrix by using a matrix completion algorithm and derives an indicator matrix from it.

Other methods aim to reconstruct and artificially made low-rank similarity matrix. This is the case of MCCC, which assigns similarity 1 for any pair of instances in the same cluster and 0 otherwise, based on the given constraints and the dataset. It can be proven that this is equivalent to finding the best data partition. A convex optimization problem, whose global solution can be efficiently obtained, can be used to solve this problem. Please note that the aim of MCCC is reconstructing the similarity matrix taking constraints into account, not the constraint matrix.

Lastly, LMRPCP performs matrix completion within a transductive learning framework. These approaches make the data matrix and the label matrix jointly low-rank and simultaneously apply a matrix completion algorithms to them. LMRPCP assumes that the data matrix is a clean low-rank matrix, while the constraint matrix is considered to be noisy and low-rank. The resulting problem is a matrix completion problem which can be solved with an augmented Lagrangian multiplier algorithm. The generated constraints are used to adjust pairwise similarities, over which classic spectral clustering is performed to obtain a partition.

7.17 Constrained Data Space Transformation

Constrained Data Space Transformation (CDST) techniques (gathered in Table 34) seek to transform the space in which the data is embeded so that the new space can include the information contained in the constraint set. This usually involves reducing or augmenting the number of dimensions of said space. The majority of methods which belong to this category seek to reduce the number of dimensions, thus summarizing (and sometimes losing) information from the original data space. Other methods augment the number of dimensions based on the constraints and without any loss of information, although they produce a larger dataset which is usually harder to process by partitional methods.

$S_{\mathcal{A}}$	Acronym	ID	Penalty	ML	CL	Hybrid	Year	Ref.
16.01	RCA	26	×	Soft	—	Constrained Distance Transformation	2005	[187]
17.68	SCREEN	41	×	Hard	Soft	—	2007	[215]
16.07	PCP	64	×	Soft	Soft	—	2008	[445]
8.22	RLC-NC	81	×	Soft	Soft	—	2009	[433]
8.29	GBSSC	98	×	Hard	Soft	Graph-based	2010	[185, 244, 245, 246]
17.73	CPSSAP	153	×	Soft	Soft	—	2012	[446]
10.31	CAF	166	×	Soft	Soft	Constraint Propagation	2013	[438]
11.55	MPHS-Linear	172	×	Soft	Soft	—	2013	[447]
11.55	MPHS-Gauss	173	×	Soft	Soft	—	2013	[447]
11.55	MPHS-PCP	174	×	Soft	Soft	—	2013	[447]
6.93	CNP-K-Means	187	×	Soft	Soft	Non Graph-based	2014	[264]
4.43	CCC-GLPCA	248	×	Soft	Soft	—	2018	[448]
7.74	DP-GLPCA	283	×	Soft	Soft	—	2021	[449]

Table 34: Feature table for CDST methods.

Most methods in this category perform dimensionality reduction over the original dataset. Some of them are based on the classic PCA algorithm: RCA, CCC-GLPCA and DP-GLPCA are some examples of this. RCA is an ML constrained version of the classic PCA algorithm. It seeks to identify and down-scale global unwanted variability within the data. In order to do so, it changes the feature space used for data representation by assigning high weights to relevant dimensions and low weights to irrelevant dimensions by means of a global linear transformation. The relevant dimensions are estimated using chunklets. CCC-GLPCA includes a regularized term in the objective function of GLPCA (a Graph-Laplacian variant of PCA) to include constraints. This regularizes the similarity between the multiple low-dimensional representations used by GLPCA. DP-GLPCA simply performs classic GLPCA over the modified distance matrix to set distances between instances related by ML and CL to 0 or 1, respectively. Besides, it includes a dissimilarity regularizer which emphasizes CL to expand their influence.

Other methods use diverse procedures to perform dimensionality reduction taking constraints into account. SCREEN includes a step where a constraint-guided feature projection method (called SCREEN$_{PROJ}$) is used to project the original data in a low-dimensional space. RLC-NC seeks a low-dimensional representation of the data through orthogonal factorizations in which the clustering structure defined by the prior knowledge is strengthened. GBSSC first obtains the chunklet graph, over which a Laplacian process is applied for the graph to reflect CL constraints. The entire resulting graph is then projected onto a lower-dimensional space. CPSSAP uses the constraint projection methods to produce a faithful representation of the constraint set in a lower-dimensional space. The affinity matrix is computed on the basis of the new data space and a classic affinity propagation algorithm is used to produce a partition of the dataset. CNP-K-Means seeks to project the original dataset into a lower-dimensional space, preserving the information contained in the constraint set. In order to do so, it defines a neighborhood for every instance based on a parameterized radius value, used to propagate the influence of constraints from constrained instances without augmenting the constraint set.

The three variants of MPHS perform dimensionality reduction in a more exotic constraint-oriented way. It is inspired by the maximum margin hyperplane of SVM. Intuitively, CL constraints can be used to define hyperplanes between pairs of instances. Given two instances with related by CL constraint, we can compute its mid-perpendicular hyperplane which is perpendicular to the line across the two instances, which is also the maximum margin hyperplane. Since there is more than one CL constraint in the constraint set, multiple mid-perpendicular hypeprlanes can be obtained. MPHS contains three main steps. Firstly, it learns a new data representation using the mid-perpendicular hyperplane corresponding to each cannot-link constraint, which can also be regarded as dimensionality reduction. Secondly, it learns individual similarity matrix according to the new data representation corresponding to each CL. In the end, individual similarity matrices are aggregated into a similarity matrix and then perform kernel k-means. Three variants of MPHS are proposed in [447]: MPHS-linear (for simple and well-structured data), which is performed on original data space, MPHS-Gauss (for complex data) which is performed on Gaussian-kernel

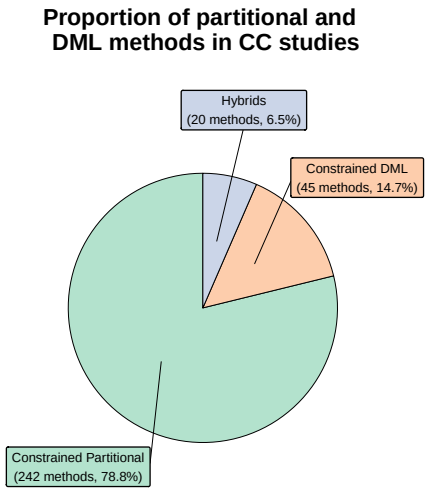
induced feature space, and MPHS-PCP which first learns a data-dependent kernel similarity (PCP-kernel [445]) and performs MPHS in PCP-kernel induced feature space.

PCP is one of the few methods which projects the original data into a higher-dimensional space to include constraints. To do so, it learns a mapping over the data graph and maps the data onto a unit hypersphere, where instances related by ML are mapped into the same point and instances related by CL are mapped to be orthogonal. This can be achieved using the kernel trick via semidefinite programming, and has to be done in a high dimensional space, as implementing it in the input space is hard if not unfeasible. Another method which projects the original data into higher dimensions is CAF. It augments the initial space with additional dimensions derived from CL constraints. The instances are augmented with additional features, each of which is defined by one of the given CL constraints. ML constraints are included by modifying the initial distance matrix so that the distance between instances related by ML is the lowest among all pairwise distances (greater than 0) and restoring metricity and the triangle inequality afterwards. Pairwise distances between instances in the augmented space combines both the distances in the original space (modified to include ML) and the distances of instances according to each CL constraint. The distance derivation for the new dimension is based on diffusion maps. The actual clustering is then performed by any classic clustering technique.

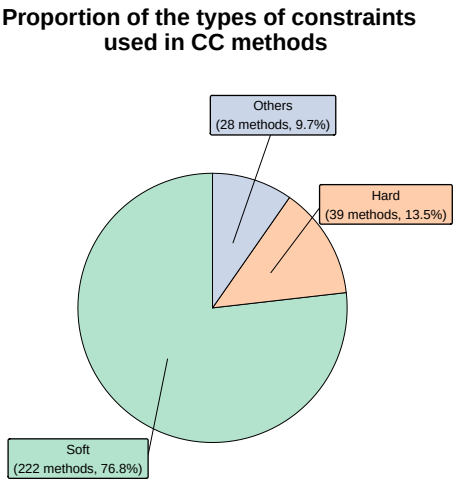
8 Statistical Analysis of the Taxonomy

This section presents relevant statistics on the ranked taxonomy proposed in Section 7. The `UpSetR` package provides the perfect tool to obtain a visualization of the overall taxonomy in the form of a statistical summary, presented in Figure 14. The left histogram represents the number of methods which belong to every category, while the top histogram considers all hybridizations found between said categories. The categories involved in hybridizations are indicated by the central dot matrix. For example, the left histogram shows that a total of 12 methods belong to the KCC category, and the top histogram shows that 11 of them are purely KCC methods, with a single hybrid method, which the central dot matrix indicates that belongs to both KCC and LSCC category thanks to (which is consistent with the information provided in Table 24).

Figure 14 shows that the ACC category is the most prominent one, while also being the one which represents the most hybridizations. The rationale behind this is that all methods which belong to the active clustering with constraints category (Table 12) always use a CC method from another category as their core CC method. In addition to this, it can be observed that the more successful hybridization are found in MMbCC+MbCC, HCC+ACC, and LSCC+ACC, with up to five hybrids methods in each combination. Please note that, for Constrained DML categories (CDT, CDST and DMM) hybridization never refer to the method used to eventually obtain a partition from their outputs, as these method are never considered to determine their category memberships. The proportion of methods which belong to the classes of the highest level dichotomy in the taxonomy (constrained partitional versus constrained DML) is presented in Figure 13a, which is introduced next to Figure 13b,



(a) Proportion of Constrained Clustering and Constrained DML methods



(b) Proportion of constraint types used in CC methods

Figure 13: Piecharts on the proportions of methods in the highest dichotomy of the taxonomy and on the types of constraints.

depicting the proportion of the types of constraints used by these methods. From these figures is clear that the vast majority of methods belong to the constrained partitional category and that the use of soft constraints is greatly preferred over any other type of constraints. The “Others” portion in Figure 13b gathers methods that consider any combination of constraints that are not purely soft nor hard. For example, methods considering only one type of constraints, or using hard ML and soft CL are included in said portion.

Another interesting statistic is presented Figure 15, which shows a histogram for the number of publications in the CC area sorted by year (please note that only three months of the year 2022 are included in this figure). It is clear that this number increases consistently from 2001 to 2008, when the proficiency of the area becomes inconsistent. Authors firmly believe that this is due to the lack of a solid, general reference in the area, which may help new researchers get a general understanding on the problems and the points of view proposed to tackle them.

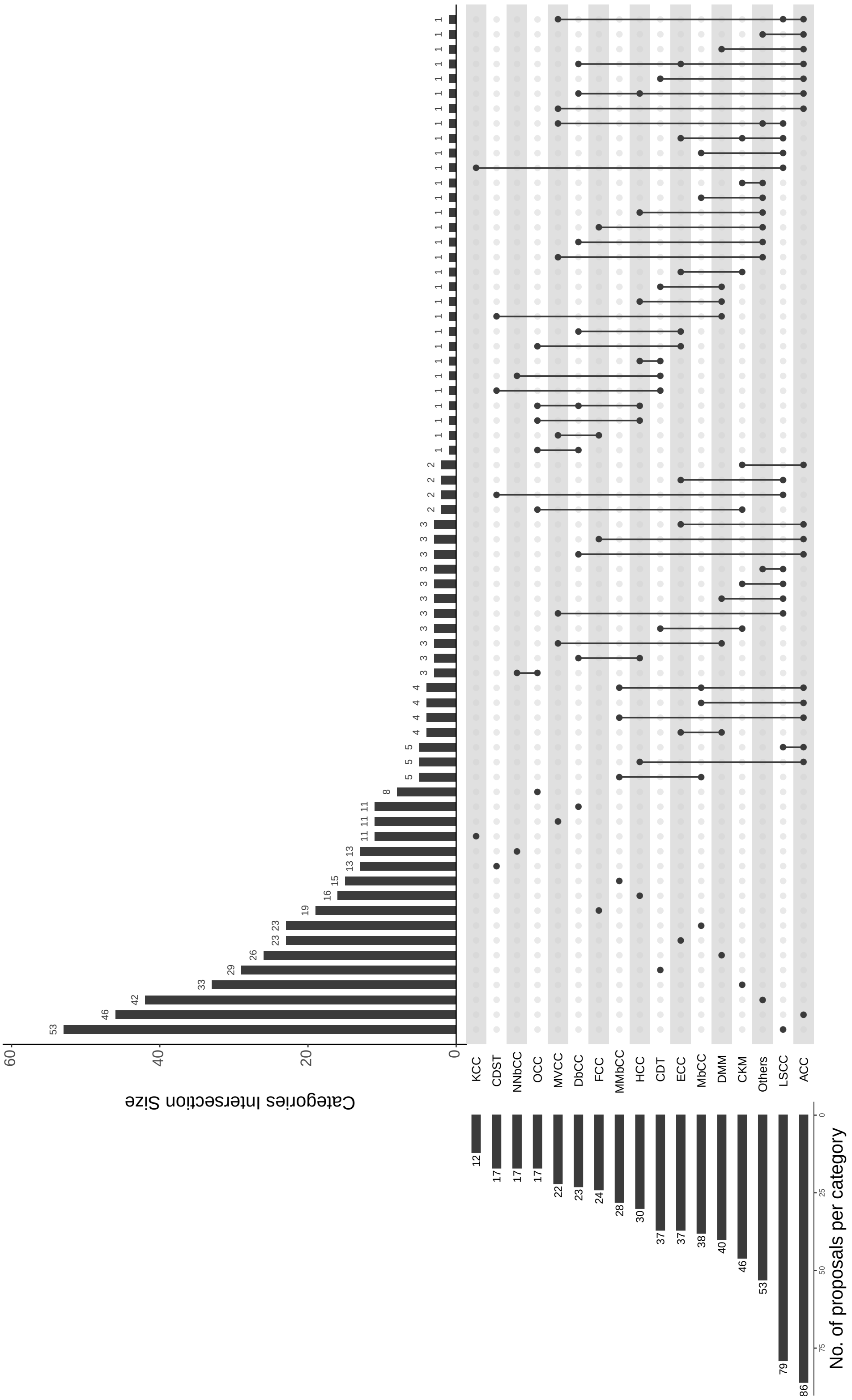


Figure 14: Summary of taxonomy categories and hybridizations.

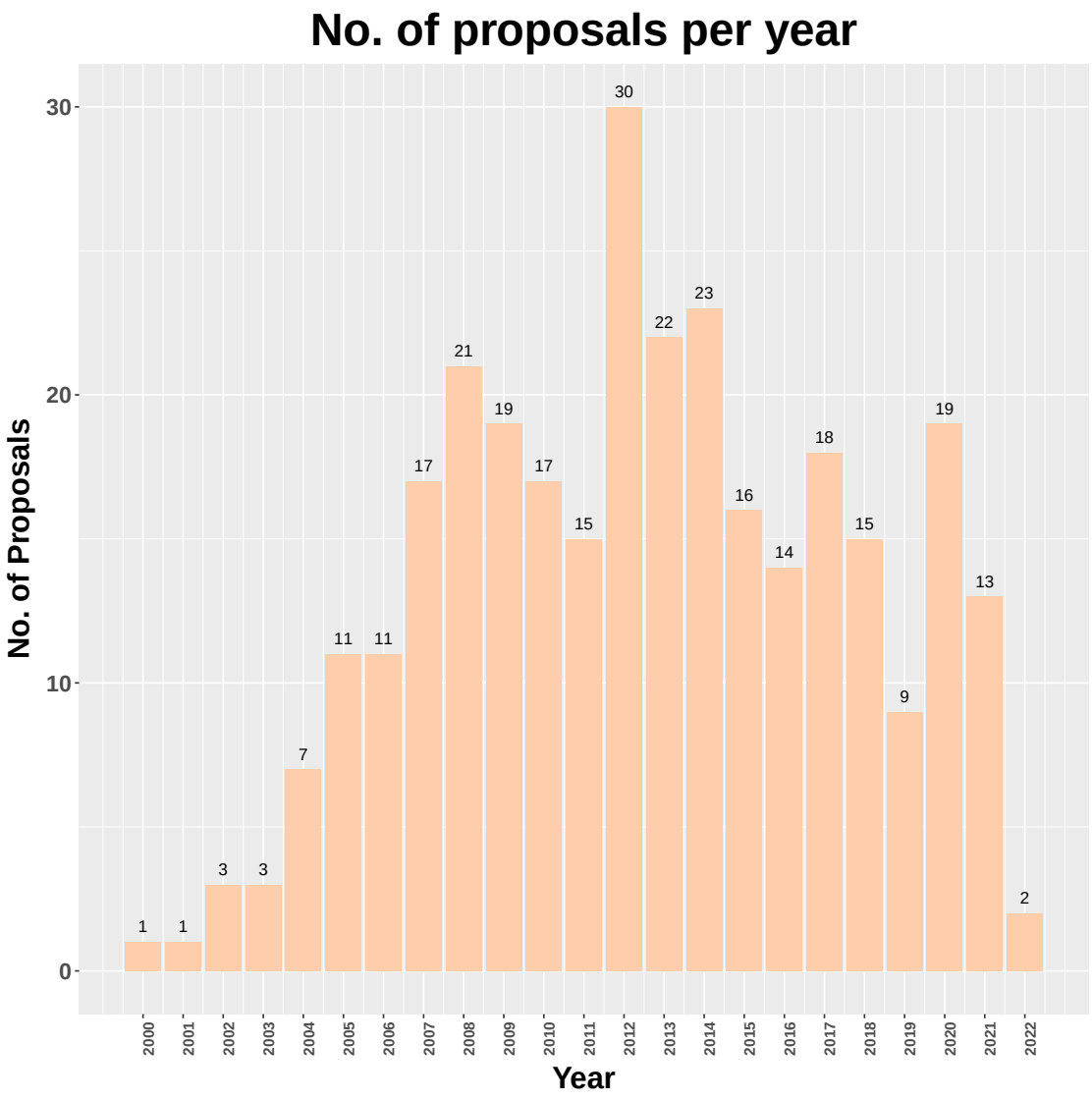


Figure 15: Number of proposals per year.

A relevant concern within any clustering task is the determination of the number of clusters k. The vast majority of both classic clustering and CC methods simply leaves this step out of the clustering process and takes k as an input hyperparameter. However there are other alternatives to approach this problem. Table 35 gathers CC methods that do not need the number of clusters k to be specified by the user, but they include procedures to determine it in some circumstances. For example, SSFCA can handle both a specification for the number of clusters and the lack of it, as it will try to automatically determine it in such case. The BoostCluster method is basically a wrapper which can be applied to any clustering method in order to include constraints in it, thus handling both cases. Other methods ask the user about an interval in which the k lies in, such as CMSSCCP, 3CP and TKC(17). Besides, there

are methods that can only work with a fixed number of clusters (usually $k = 2$), such as CSP and ISL. The rest of the methods do not accept the number of cluster of the output partitions in their inputs, and include procedures to determine it through the clustering process.

Acronym	Type	Ref.	Acronym	Type	Ref.
SSFCA	Hybrid	[125]	URASC	Not Needed	[267]
BoostCluster	Hybrid	[356]	PCCA	Not Needed	[377, 143]
CMSSCCP	Bounded	[401]	SemiDen	Not Needed	[388]
3CP	Bounded	[400]	FIECE-EM+LUC	Not Needed	[305, 306]
TKC(17)	Bounded	[399]	CECM	Not Needed	[366]
CSP	Fixed	[255]	En-Ant	Not Needed	[327]
ISL	Fixed	[71]	CAC	Not Needed	[206, 332]
SSDC	Not Needed	[390]	CELA	Not Needed	[331]
COBRAS	Not Needed	[304]	MELA	Not Needed	[331]
FIECE-EM	Not Needed	[339]	MCLA	Not Needed	[331]
JDG	Not Needed	[339]	SSAP	Not Needed	[395]
Semi-MultiCons	Not Needed	[325]	ACC	Not Needed	[144]
ssFS	Not Needed	[359]	AFCC	Not Needed	[297]
SFFD	Not Needed	[370]	COP-b-coloring	Not Needed	[242]
AAVV	Not Needed	[295]	MOCK	Not Needed	[333]
FIECE-EM+BFCU	Not Needed	[305, 306]	MCGMM	Not Needed	[450]
FIECE-EM+FCU	Not Needed	[305, 306]	COBRA	Not Needed	[108]
FIECE-EM+DVO	Not Needed	[305, 306]	ASCENT	Not Needed	[193]
JBMJ	Not Needed	[414]	-	-	-

Table 35: Methods that do not need K to be specified.

8.1 Statistic Analysis of Ranked Scores

This section tackles the distribution of the final scores, obtained by applying the methodology introduced in Section 6. Firstly, Figure 16 gives the values for α_1 and α_2 for every year, so the reader can have a better understanding of how the final scores are obtained. Overall, α_1 and α_2 do not show sufficiently significant differences between them for the year of publication to be decisive, although enough to be used as a discriminating factor on a reasonable scale.

Figure 17 gives a visual representation of the detailed scorings for the top 20 best ranked methods. As expected, the best score is obtained by the COP-K-Means method, as it features a high quality experimental setup (S1, S2 and S3) and it is the single most cited CC method ever proposed (S7). This is enough for it to obtain the highest score, even if the paper which

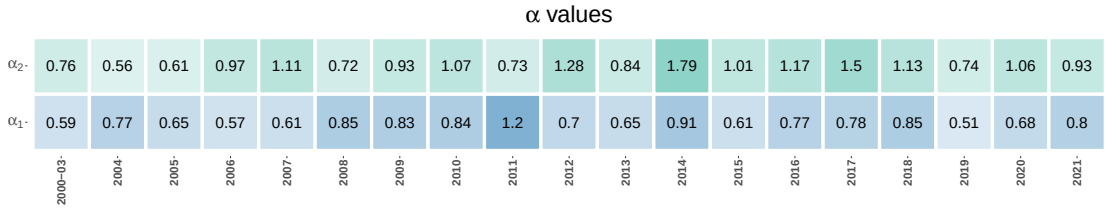


Figure 16: Values for the weighting parameters α_1 and α_2 used in every year.

proposes it lacks of a proper validation procedure (S4 and S5). Please note that the fact that COP-K-Means is ranked as the best method ever proposed is evidence of the scoring system being reluctant to the number of years the method has been available. As a result, one of the oldest CC method reaches top 1 in the ranking, and is followed by a method which was proposed as recently as in 2021, which is SHADE$_{CC}$. A number of baseline method make it to the top 20, such as ERCA, SSKK, MPCK-Means, DCA or CSI. The reason behind this is the high influence of the $I'_{\mathcal{A}}$ term over the final score $S_{\mathcal{A}}$. Other methods reach the top 20 by other means, such as a high experimental quality combined with proper validation procedures.

Finally, Figure 18 shows a histogram with the distribution of all final scores presented in Section 7. It is clear that the majority of methods are scored in $[0, 10]$, fewer methods are in the next higher range of values, which is $(10, 20]$, and very few outlier scores are found in the range $(20, 35]$. Please note that the effective output range of the scoring systems is $[0, 33.33]$, as no method fully complies with its standards. This shows the suitability of the proposed scoring system, as any objective raking procedure should place the majority of methods in the low-medium range and few methods in the upper range of the ranking, with those methods being the more remarkable ones for their quality and the robustness of their conclusions.

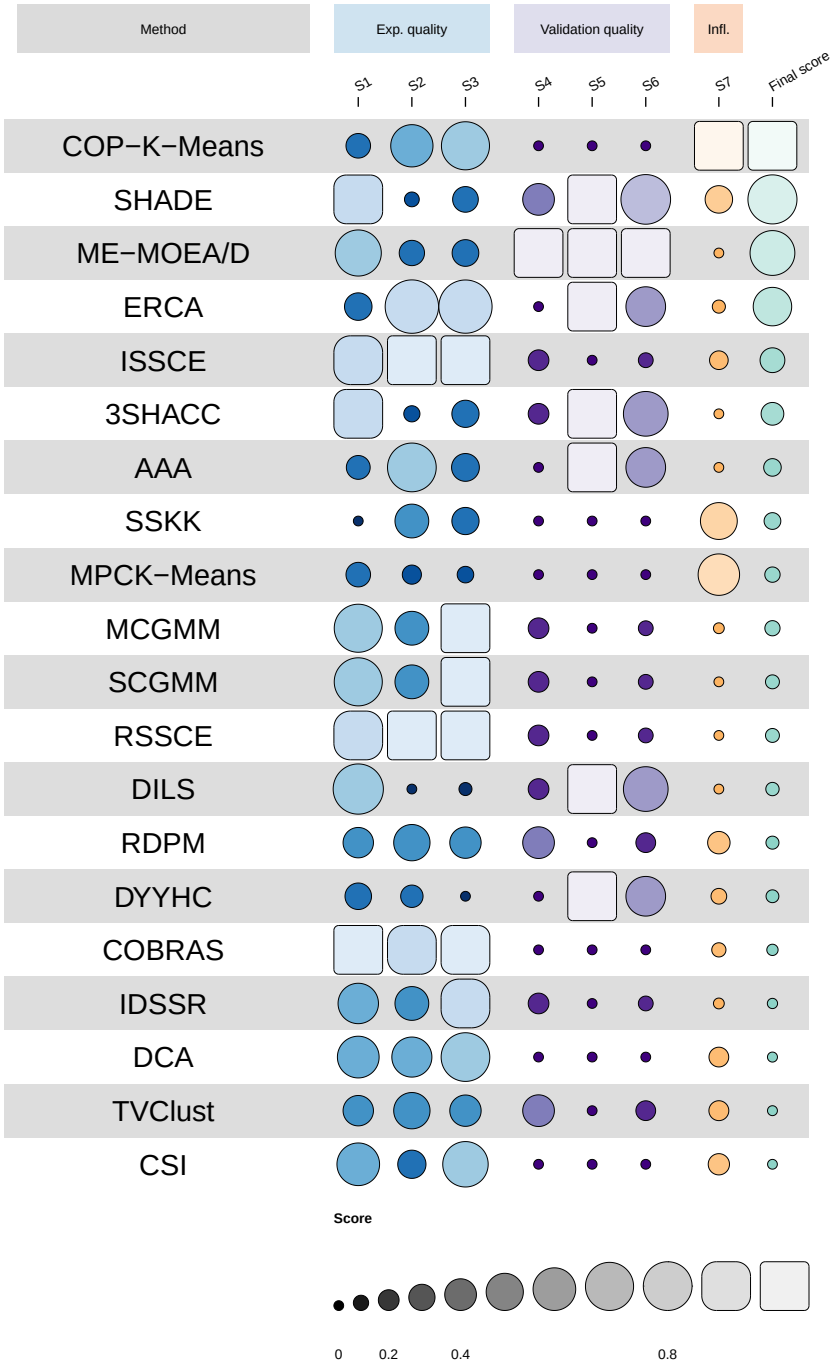


Figure 17: Detailed scorings for the top 20 ranked methods. Every subscore (S1 to S7) has its own meaning and can be understood individually as follows: the weighted year-normalized number of datasets ($\alpha_1^{Y_{\mathcal{A}}} D'_{\mathcal{A}}$), the weighted year-normalized number of methods ($\alpha_2^{Y_{\mathcal{A}}} MS'_{\mathcal{A}}$), the normalized experimental quality ($EQ'_{\mathcal{A}}$), the normalized number of validity indices ($V'_{\mathcal{A}}$), the statistical test usage indicator ($T_{\mathcal{A}}$), the normalized validation procedure quality ($VQ'_{\mathcal{A}}$), and the normalized influence ($I'_{\mathcal{A}}$). Finally, $S_{\mathcal{A}}$ is given in the final score column.

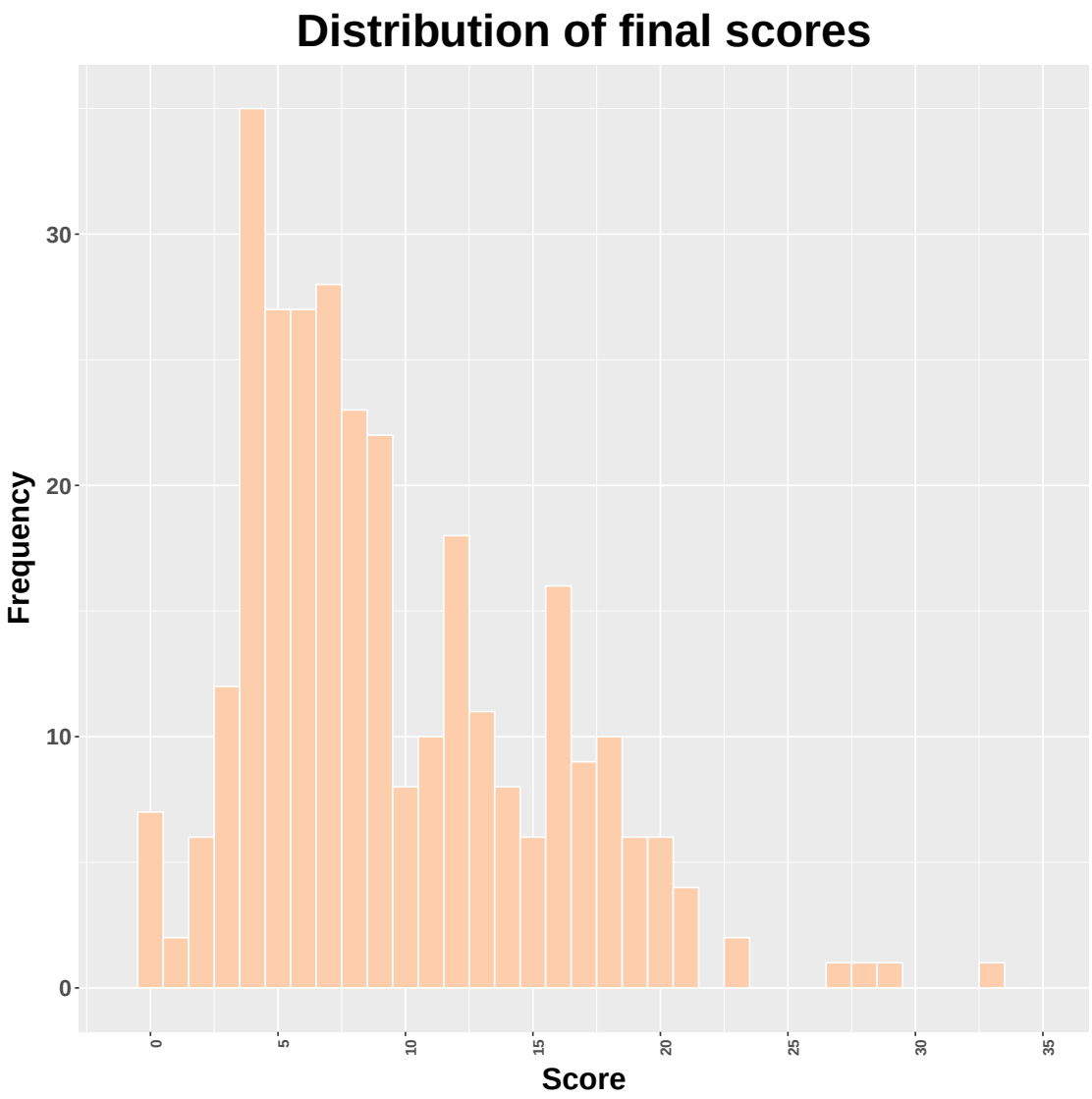


Figure 18: Distribution of all final scores presented in Section 7.

9 Conclusions, Criticisms and Future Research Guidelines

This study presented a systematical review on the Constrained Clustering (CC) research domain. Firstly, a general introduction to the Semi-Supervised Learning (SSL) paradigm is given, after which the specific area of semi-supervised clustering is discussed in detail. The discussed methods within this area are capable of including background knowledge (or incomplete information) about the dataset onto the clustering process. To the best of the authors' knowledge, this study provides the first ever overview and taxonomization of the types of background knowledge (in Section 2) that can be included as constraints in semi-supervised clustering. Afterwards, we motivate why out of all types of background knowledge, the instance-level pairwise Must-Link (ML) and Cannot-Link (CL) constraints, are the most successful and prominent types of constraints. Semi-supervised clustering methods that use ML and CL constraints are known as Constrained Clustering (CC). The CC problem is formalized and presented in later sections, and illustrated by giving examples of several practical fields of applications. Afterwards, advanced CC concepts and structures are described and formalized, allowing to discuss in more detail the advantages and disadvantages of different approaches. Afterwards, a statistical analysis of the experimental elements used in studies which proposes new CC methods has been carried out, revealing the basics research in CC area, as well as the baseline methods, benchmark datasets and suitable validity indices. This overview leads to the proposal of an objective scoring system that captures the potential and relevance of each approach, which is used later to assign every method a score that summarizes its quality, and to produce a ranking of CC methods. Afterwards, a taxonomization of 307 CC methods is conducted, ranking them according to the proposed scoring system and categorizing them in two major families: constrained partitional and constrained DML. These two families are further divided in more specific categories, whose common features and specific methods are described in Sections 7.1 to 7.17.

The proposed taxonomy can be used to:

- Decide which type of approach and model is best suited to a new constrained clustering problem.
- Compare newly proposed techniques to those belonging to the same family in this taxonomy, so that in can be determined whether the new method represents an improvement over the current state-of-the-art.
- Identify the proposals which best support their conclusions and propose more robust methods, thanks to the scoring system.

As any other Computer Science research area, the CC area is not free of flaws and criticism. Having reviewed 270 studies (proposing 307 methods), the authors have identified 5 major problems which affect the vast majority of them. These problems can be summarized as follows:

- **The lack of a unified, general reference.** There is not an updated, general reference unifying the overall CC literature. This affects the foundations of new propos-

als, as it is hard to find the state-of-the-art methods that can be used to compare new techniques with. This is illustrated in Figure 9a, which shows that more than half of the proposed CC methods (52.1%) are never used in subsequent studies. The aim of this study is to address this deficiency.

- **Low number of application studies**. Even if the main purpose of this study is not to review and gather literature concerning CC applications, it has been difficult for the authors to find application studies other than the 95 presented in Section 3.5. This is specially significant, given the high number and diversity found in CC studies which propose new methods.

- **The lack of extensive experimental comparisons.** From Section 5, it is clear that, unfortunately, the CC research area is consistently poor regarding the number of datasets, validity indices and competing methods used to support their conclusions. Very few methods use more than 10 datasets or 2 validity indices. With respect to the number of competing methods, it is shocking that no study has ever considered more than 9 of them, given that there are, at least, 307. The authors firmly believe that (open-source) code unavailability is responsible for this unsettling fact, as implementation details are rarely given in CC studies.

- **Unavailability of specific standardized CC-oriented datasets and constraint sets.** One of the major flaws of the CC area is the lack of specific datasets, which may be justified, provided that the low number of application papers. However, this is not the case when it comes to the constraint sets. In Section 5.4, the constraint generation method used in the vast majority of CC studies was presented. It is clear that this procedure is highly dependent of random effects, as constraints are randomly allocated. For this reason, it is necessary to have access to the specific constraint sets used in a given experimentation if it needs to be reproduced. Unfortunately, constraint sets are only rarely published by authors.

- **Statistically unsupported experimental conclusions**. This is an effect derived from the two previous criticisms, as a low number of standardized experiments is not significant enough to perform statistical testing procedures or to derive generalized conclusions that are broadly applicable. In fact, a very reduced minority of studies support their conclusions using statistical testing. This may greatly affect the confidence future researches may have towards the reviewed studies, reducing their usability in both the development of new methods and their applications in real-world problems.

In respect to future research guidelines, new proposals would greatly benefit from avoiding the mentioned flaws as much as possible. New studies should perform extensive experimental comparisons with state-of-the-art methods. A major goal of this overview is to provide easy access to such methods. Similarly, studies would benefit from the use of multiple datasets focusing on a wide range of application domains to make conclusions more generalisable. To this end, authors should always make their datasets and constraint sets public,

in order for the result to be reproducible, and preferably also the source code. Supporting conclusions with statistical testing is also essential, as this will increase confidence in the results.

Future research based on this study can extend the scoring system to evaluate not only quantitative features regarding the quality of the proposal, but also qualitative features, as their novelty or their applicability. This objective would be arduous to achieve with objective standards in mind, as these features greatly depend on the perception of the researchers. Additionally, the constraint equivalences presented in Section 2.8 would benefit from formalization. Lastly, the creation of a library of CC baseline algorithms would greatly benefit this research area, making it more accessible to new researchers. Thanks to this study, the mentioned CC baselines algorithms can be chosen using objective criteria.

Acknowledgements

This study is has been funded by the research projects PID2020-119478GB-I00, A-TIC-434-UGR20 and PREDOC_01648.

A Full Names for all Methods Reviewed

This appendix gathers Tables 36 to 47, which list all method reviewed in this study, sorted by their identifier, and giving their full names. Methods named after the initials of their authors' names are marked with the word "names". Methods whose name does not refer to neither any acronym nor authors' initials are marked with a hyphen dash ("-").

ID	Acronym	Full Name
1	COP-COBWEB	COnstrained Partitional - COBWEB
2	COP-K-Means	COnstrained Partitional - K-Means
3	SCOP-K-Means	Soft COnstrained Partitional - K-Means
4	CCL	Constrained Complete-Link
5	CSI	Clustring with Side Information
6	KKM	names
7	TBJSBM	names
8	MAA	names
9	PCK-Means	Pairwise Constrained K-Means
10	FFQS	Farthest First Query Selection
11	MPCK-Means	Metric Pairwise Constrained K-Means
12	HMRF-K-Means	Hidden Markov Random Fields - K-Means
13	DistBoost	—
14	Constrained EM	Constrained Expectation-Minimization
15	LLMA	Locally Linear Metric Adaptation
16	ACCESS	Active Constrained Clustering by Examining Spectral eigenvectorS
17	IDSSR	names
18	GPK-Means	Gaussian Propagated K-Means
19	CVQE	Constrained Vector Quantization Error
20	CSC	Constrained Spectral Clustering
21	PPC	Penalized Probabilistic Clustering
22	PCCA	Pairwise Constrained Competitive Agglomeration
23	SSKK	Semi-Supervised Kernel K-Means

Table 36: Full names for all methods reviews, sorted by ID (Part I).

ID	Acronym	Full Name
24	MCGMM	Multiple-Component Gaussian Mixture Model
25	SCGMM	Single-Component Gaussian Mixture Model
26	RCA	Relevant Components Analysis
27	ASSKK	Adaptive Semi Supervised Kernel K-Means
28	COALA	Constrained Orthogonal Average Link Algorithm
29	COALAcat	Constrained Orthogonal Average Link Algorithm (Categorical)
30	MOCK	Multi Objective Clustering with automatic K-determination
31	ERCA	Extended Relevant Components Analysis
32	DCA	Discriminative Component Analysis
33	KDCA	Kernel - Discriminative Component Analysis
34	CME	names
35	Comraf	Combinatorial Markov Random Fields
36	SS-FKCN	Semi-Supervised Fuzzy Kohonen Clustering Network
37	SMCE	Subspace Metric Cluster Ensemble
38	COP-b-coloring	COnstrained Partitional - b-coloring
39	C-DBSCAN	Constraint-driven - DBSCAN
40	BoostCluster	—
41	SCREEN	Semi-supervised Clustering method based on spheRical k-mEans via fEature projectioN
42	PCSK-Means	Pairwise Constrained Spherical K-Means
43	RHWL	names
44	LCVQE	Linear Constrained Vector Quantization Error
45	NMFS	Non-negative Matrix Factorization-based Semi-supervised
46	ClusILC	Clustering with Instance-Level Constraints
47	DYYHC	names
48	SPGP	Semi-supervised Pairwise Gaussian Process classifier
49	SSCARD	Semi-Supervised Clustering and Aggregating Relational Data
50	ITML	Information-Theoretic Metric Learning

Table 37: Full names for all methods reviews, sorted by ID (Part II).

ID	Acronym	Full Name
51	SMR	names
52	S-SCAD	Semi-Supervised Clustering and Attribute Discrimination
53	EICC	Efficient Incremental Constrained Clustering
54	PAST-Toss	Pick A Spanning Tree – Toss
55	CLAC	Constrained Locally Adaptive Clustering
56	PT	Penta-Training
57	MLC-K-Means	Must-Link Constrained - K-Means
58	CDPMM08	Constrained Dirichlet Process Mixture Models 2008
59	AFCC	Active Fuzzy Constrained Clustering
60	ACC	Adaptive Constrained Clustering
61	LCPN	names
62	COP-CGA	COnstrained Partitional - Clustering Genetic Algorithm
63	S3-K-Means	Semi-Supervised Spectral K-Means
64	PCP	Pairwise Constraint Propagation
65	COP-HGA	COnstrained Partitional - Hybrid Genetic Algorithm
66	CLWC	Constrained Locally Weighted Clustering
67	ADFT	Alternative Distance Function Transformation
68	CMMC	Constrained Maximum Margin Clustering
69	MMFFQS	Min-Max Farthest First Query Selection
70	YLYM	names
71	COPGB-K-Means	COnstrained Partitional Graph-Based - K-Means
72	PCCK-Means	Partial Closure-based Constrained K-Means
73	SS-NMF(08)	Semi-Supervised - Non-negative Matrix Factorization
74	Xiang's	—
75	Cop-EAC-SL	Constrained partitional Evicende ACcumulation Single Link
76	SCK-Means	Soft Constrained K-Means
77	PrTM	Probabilistic Topographic Mapping
78	PCsFCM	Pairwise Constrained standard Fuzzy c-means
79	PCeFCM	Pairwise Constrained entropy Fuzzy c-means

Table 38: Full names for all methods reviews, sorted by ID (Part III).

ID	Acronym	Full Name
80	C-DenStream	Constrained - Density Stream
81	RLC-NC	Constrained Normalized Cut
82	Lo-NC	Local Normalized Cut
83	MSSB	names
84	cut	—
85	SSAP	Semi-Supervised Affinity Propagation
86	RSCPC	Regularized Spectral Clustering with Pairwise Constraints
87	CCSR	Constrained Clustering with Spectral Regularization
88	SCLC	Spectral Clustering with Linear Constraints
89	CCSKL	Constrained Clustering by Spectral Kernel Learning
90	CMSC	Constrained Mean Shift Clustering
91	MCLA	Must-link Cannot-link Leader Ant
92	MELA	Must-link ϵ-link Leader Ant
93	CELA	Cannot-link ϵ-link Leader Ant
94	AHC-CTP	Agglomerative Hierarchical Clustering – Clusterwise Tolerance Pairwise
95	E^2CP	Exhaustive and Efficient Constraint Propagation
96	OSS-NMF	Orthogonal Semi-Supervised - Non-negative Matrix tri-Factorization
97	RJFM	names
98	GBSSC	Graph-Based Semi-Supervised Clustering
99	Samarah	—
100	MSBSBS	names
101	SCKMM	Semi-supervised Clustering Kernel Method based on Metric learning
102	PCBK-Means	Pairwise Constrained Based K-Means
103	ISL	names
104	ICOP-K-Means	Improved COP-K-Means
105	CSP	Constrained SPectral clustering
106	ASC(10)	Ability to Separate between Clusters
107	MSBSBS(10)	names

Table 39: Full names for all methods reviews, sorted by ID (Part IV).

ID	Acronym	Full Name
108	LRML	Laplacian Regularized Metric Learning
109	SS-NMF	Semi-Supervised Nonnegative Matrix Factorization
110	ASC	Active Spectral Clustering
111	S3OM	Semi-Supervised Self Organizing Map
112	CCHAMELEON	Constrained CHAMELEON
113	CAC1	Constrained Active Clustering 1
114	PCKMMR	Parallel COP-K-Means based on MapReduce
115	CLC-K-Means	Cannot-Link Constrained – K-Means
116	SDHCC	Semi-supervised Divisive Hierarchical Clustering of Categorical data
117	AHCP	Agglomerative Hierarchical Clustering with Penalties
118	LRKL	Low-Rank Kernel Learning
119	CDJPBY	names
120	SRCP	Symmetric graph Regularized Constraint Propagation
121	SGID	names
122	MMCP	Multi-Modal Constraint Propagation
123	PNMF	Penalty - Nonnegative Matrix Factorization
124	SCAP	Semi-supervised fuzzy Clustering Algorithm with Pairwise constraints
125	SFFA	names
126	PSC	Pareto based multi objective algorithm for Semi-supervised Clustering
127	SemiStream	—
128	LCP	Local Constraint Propagation
129	SSCsNMF	Semi-supervised symmetriC Non-negative Matrix Factorization
130	JPBMS	names
131	SSL-EC	Semi-Supervised Learning based on Exemplar Constraints
132	CAC	Constrained Ant Clustering
133	En-Ant	Ensemble Ant
134	SSFCA	Semi-Supervised Fuzzy Clustering Algorithm

Table 40: Full names for all methods reviews, sorted by ID (Part V).

ID	Acronym	Full Name
135	NSDR-NCuts	Near Stranger or Distant Relatives – NCuts
136	SNN	Similarity Neural Networks
137	Cons-DBSCAN	Constrained - DBSCAN
138	COSC	Constrained One Spectral Clustering
139	CECM	Constrained Evidential C-Means
140	PMMC	Pairwise-constrained Maximum Margin Clustering
141	TRAGEK	TRAnsductive Graph Embedding Kernel
142	ENPAKL	Efficient Non-PArametric Kernel Learning
143	COP-SOM-E	COnstrained Partitional - Self Organazing Map - Ensemble
144	IU-Red	Iterative Uncertainty Reduction
145	RFD	Random Forest Distance
146	SCEV	Semi-supervised Clustering Ensemble by Voting
147	sRLe-GDM-FFS	semi-supervised Robust Learning of finite Generalized Dirichlet Mixture models and Feature Subset Selection
148	LSE	Learned Spectral Embedding
149	CPSC	Constrained Polygonal Spatial Clustering
150	CPSC*	Constrained Polygonal Spatial Clustering*
151	CPSC*-PS	Constrained Polygonal Spatial Clustering-Polygon Split
152	SSC-ESE	Semi-Supervised Clustering with Enhanced Spectral Embedding
153	CPSSAP	Constraint Projections Semi-Supervised Affinity Propagation
154	TKC	names
155	SRBR	names
156	A-ITML-K-Means	Active - Information Theoric Metric Learning - K-Means
157	SSKSRM	Semi-Supervised Kernel Switching Regression Models
158	SSSeKRM	Semi-Supervised Sequential Kernel Regression Models

Table 41: Full names for all methods reviews, sorted by ID (Part VI).

ID	Acronym	Full Name
159	O-LCVQE	Online - Linear Constrained Vector Quantization Error
160	C-RPCL	Constrained - Rival Penalyzed Competitive Learning
161	SSsFCMCT	Semi-Supervised standard Fuzzy C-Means for data with Clusterwise Tolerance by opposite criteria
162	SSeFCMCT	Semi-Supervised entropy Fuzzy C-Means for data with Clusterwise Tolerance by opposite criteria
163	SS-FCC	Semi-Supervised - Fuzzy Co-Clustering
164	MVSCE	Majority Voting Semi-supervised Clustering Ensemble
165	MCCC	Matrix Completion based Constraint Clustering
166	CAF	Constraints As Features
167	UCP	Unified Constraint Propagation
168	SS-CLAMP	Semi-Supervised fuzzy C-medoids CLustering Algorithm of relational data with Multiple Prototype representation
169	MSCP	Multi-Source Constraint Propagation
170	E^2CPE	Exhaustive and Efficient Constraint Propagation Ensemble
171	TwoClaCMMC	Two Classes Constrained Maximum Margin Clustering
172	MPHS-Linear	Mid-Perpendicular Hyperplane Similarity - Linear
173	MPHS-Gauss	Mid-Perpendicular Hyperplane Similarity - Gaussian
174	MPHS-PCP	Mid-Perpendicular Hyperplane Similarity - Pairwise Constraint Propagation
175	CCG	Constrained Column Generation
176	PC-HCM-NM	Pairwise Constrained - Hard C-Means - Non Metric
177	PC-eFCM-NM	Pairwise Constrained - entropy Fuzzy C-Means - Non Metric
178	PC-sFCM-NM	Pairwise Constrained - standard Fuzzy C-Means - Non Metric
179	ACC(14)	Adaptive Constrained Clustering
180	ALCSSC	Active Learning of Constraints for Semi-Supervised Clustering

Table 42: Full names for all methods reviews, sorted by ID (Part VII).

ID	Acronym	Full Name
181	SSCA	Semi-supervised Spectral Clustering Algorithm
182	MVCC	Multi-View Constrained Clustering
183	FHCSC	Flexible Highly Constrained Spectral Clustering
184	SCRAWL	Semi-supervised Clustering via RAndom WaLk
185	AC-CF-tree	Active Constrained - Clustering Feature - tree
186	SACS	Sequential Approach for Constraint Selection
187	CNP-K-Means	Constraint Neighborhood Projections - K-Means
188	STSC	Self-Taught Spectral Clustering
189	Active-HACC	Active - Hierarchical Agglomerative Constrained Clustering
190	SKMS	Semi-supervised Kernel Mean Shift
191	PCS	PCK-Means with Size constraints
192	CEVCLUS	Constrained EVidential CLUStering
193	TDCK-Means	Temporal-Driven Constrained K-Means
194	SKML	Spectral Kernel Metric Learning
195	SMVC	Semi-supervised Multi-View Clustering
196	3SMIC	Semi-Supervised Squared-loss Mutual Information Clustering
197	3CP	Constrained Clustering by Constraint Programming
198	MCPCP	Matrix Completion - Pairwise Constraint Propagation
199	CMSSCCP	Constrained Minimum Sum of Squares Clustering by Constraint Programming
200	LXDXD	names
201	CS2GS	Constrained Semi-Supervised Growing SOM
202	CMKIPCM	Constrained Multiple Kernels Improved Possibilistic C-Means
203	CCCPYL	names
204	LSCP	Learning Similarity of Constraint Propagation
205	BBMSC	Branch-and-Bound Method for Subspace Clustering
206	SCSC	Semi-supervised Clustering with Sequential Constraints
207	HSCE	Hybrid Semi-supervised Clustering Ensemble

Table 43: Full names for all methods reviews, sorted by ID (Part VIII).

ID	Acronym	Full Name
208	LMRPCP	Low-rank Matrix Recovery based Pairwise Constraint Propagation
209	TVClust	Two-Views Clustering
210	RDPM	Relational Diritchlet Process Means
211	CPSNMF	Constrained Propagation for Semi-supervised Nonnegative Matrix Factorization
212	CEAC	Constrained Evolutionary Algorithm for Clustering
213	SDenPeak	Semi-Supervised Density Peak
214	ISSCE	Incremental Semi-Supervised Clustering Ensemble
215	RSSCE	Random Subspace based Semi-supervised Clustering Ensemble
216	SSMMHF	Semi-Supervised Max-Margin Hierarchy Forest
217	C^3	Constrained Community Clustering
218	FAST-GE	Fast-Generalized Spectral Clustering
219	NMFCC	Non-negative Matrix Factorization based Constrained Clustering
220	SymNMFCC	Symmetric Non-negative Matrix Factorization based Constrained Clustering
221	AAA	names
222	DCPR	—
223	CCLS	Constrained Clustering by Local Search
224	SemiDen	Semi-supervised Density-based data clustering
225	COBRA	COnstraint-Based Repeated Aggregation
226	MVMC	Multi-View Matrix Completion
227	URASC	Uncertainty Reducing Active Spectral Clustering
228	JBMJ	names
229	TKC(17)	names
230	SCHAIN	Semi-supervised Clustering in Heterogeneous Attributed Information Networks
231	FAST-GE2.0	Fast-Generalized Spectral Clustering 2.0
232	FQH	names
233	TI-APJCF	Type-I Affinity and Penalty Jointly Constrained Spectral Clustering

Table 44: Full names for all methods reviews, sorted by ID (Part IX).

ID	Acronym	Full Name
234	TII-APJCF	Type-II Affinity and Penalty Jointly Constrained Spectral Clustering
235	BRKGA+LS	Biased Random Key Genetic Algorithm + Local Search
236	CG+PR+LS	Column Generation + Path Relinking + Local Search
237	RSEMICE	Random subspace based SEMI-supervised Clustering Ensemble framework
238	C_4S	Constrained Clustering with a Complex Cluster Structure
239	YZWD	names
240	COBS	COnstraint-Based Selection
241	A-COBS	Active - COnstraint-Based Selection
242	CVQE+	Constrained Vector Quantization Error +
243	SSDC	Semi-Supervised DenPeak Clustering
244	COBRAS	COnstraint-Based Repeated Aggregation and Splitting
245	k-CEVCLUS	k - Constrained EVidentialCLUStering
246	AHMRF	A - Hidden Markov Random Field
247	FIECE-EM	Feasible-Infeasible Evolutionary Create & Eliminate - Expectation Maximization
248	CCC-GLPCA	Convex Constrained Clustering - Graph-Laplacian PCA
249	JDG	names
250	d-graph	—
251	PCPSNMF	Pairwise Constraint Propagation-induced Symmetric NMF
252	CESCP	Clustering Ensemble based on Selected Constraint Projection
253	DCECP	Double-weighting Clustering Ensemble with Constraint Projection
254	CMVNMF	Constrained Multi-View NMF
255	BCK-Means	Binary Constrained K-Means
256	SSKMP	Semi-Supervised K-Medioids Problem

Table 45: Full names for all methods reviews, sorted by ID (Part X).

ID	Acronym	Full Name
257	SFS^{3}EC	Stratified Feature Sampling for Semi-Supervised Ensemble Clustering
258	SemiSync	—
259	SCAN	Semi-supervised clustering with Coupled attributes in Attributed heterogeneous information Networks
260	SDEC	Semi-supervised Deep Embedded Clustering
261	MVCSC	Multi-View Constrained Spectral Clustering
262	2SHACC	2 -Stages Hybrid Agglomerative Constrained Clustering
263	ARSCE	Adaptive Regularized Semi-supervised Clustering Ensemble
264	CCPP	Constrained Clustering via Post-Processing
265	DCC	Deep Constrained Clustering
266	AMH-L	names
267	AMH-NL	names
268	S^{3}C^2	Semi-Supervised Siamese Classifiers for Clustering
269	PCOG	Pairwise Constrained Optimal Graph
270	DILS$_{CC}$	Dual Iterative Local Search - Constrained Clustering
271	NLPPC	New Label Propagation with Pairwise Constraints
272	SSDPC	Semi-Supervised Density Peak Clustering
273	PCPDAMR	Pairwise Constraint Propagation with Dual Adversarial Manifold Regularization
274	CDC	Constrained Deep Clustering
275	CDEC	Constrained Deep Embedded Clustering
276	PCSK-Means(21)	Pairwise Constrained Sparse K-Means
277	ILMCP	Instance Level Multi-modal Constraint Propagation
278	WECR K-Means	WEighted Consensus of Random K-Means ensemble
279	fssK-Means	fast semi-supervised K-Means
280	fssDBSCAN	fast semi-supervised DBSCAN
281	ADPE	Active Density Peak Ensemble
282	ADP	Active Density Peak

Table 46: Full names for all methods reviews, sorted by ID (Part XI).

ID	Acronym	Full Name
283	DP-GLPCA	Dissimilarity Propagation-guided - Graph-Laplacian Principal Component Analysis
284	NN-EVCLUS	Neural Network-based EVidential CLUSstering
285	SHADE$_{CC}$	Succes History-based Adaptive Differential Evolution - Constrained Clustering
286	ME-MOEA/D$_{CC}$	Memetic Elitist - Multiobjective Optimization Evolutionary Algorithm based on Decomposition - Constrained Clustering
287	3SHACC	3 -Stages Hybrid Agglomerative Constrained Clustering
288	Semi-MultiCons	Semi-supervised Multiple Consensus
289	DGPC	names
290	KAKB	names
291	ssFS	semi-supervised subgraph Feature Selection
292	JDFD	names
293	WAKL	names
294	MICS	Most InformativeConStraints
295	SFFD	Semi-supervised Fuzzy clustering with Feature Discrimination
296	AAA(18)	names
297	RWACS	Random Walk Approach to Constraints Selection
298	AAA(19)	names
299	ALPCS	Active Learning Pairwise Constraint based on Skeletons
300	LCML	names
301	AIPC	Active Informative Pairwise Constraints algorithm
302	AAVV	names
303	FIECE-EM+BFCU	FIECE-EM + Best Feasible Classification Uncertainty
304	FIECE-EM+FCU	FIECE-EM + Feasible Classification Uncertainty
305	FIECE-EM+DVO	FIECE-EM + Distance to Violated Objects
306	FIECE-EM+LUC	FIECE-EM + Largest Unlabeled Clusters
307	ASCENT	—

Table 47: Full names for all methods reviews, sorted by ID (Part XII).

References

[1] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[2] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[3] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[4] Sadaaki Miyamoto and A Terami. Inductive vs. transductive clustering using kernel functions and pairwise constraints. In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 1258–1264. IEEE, 2011.

[5] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

[6] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[7] Zhi-Hua Zhou. Semi-supervised learning. In *Machine Learning*, pages 315–341. Springer, 2021.

[8] Amarnag Subramanya and Partha Pratim Talukdar. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(4):1–125, 2014.

[9] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2):245–284, 2015.

[10] Eric Bair. Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(5):349–361, 2013.

[11] Ian Davidson and Sugato Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from data*, 1:1–41, 2007.

[12] Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, 2008.

[13] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *A review of machine learning techniques for processing multimedia content*, 1:9–16, 2004.

[14] Hongfu Liu and Yun Fu. Clustering with partition level side information. In *2015 IEEE international conference on data mining*, pages 877–882. IEEE, 2015.

[15] Marek Śmieja and Bernhard C Geiger. Semi-supervised cross-entropy clustering with information bottleneck constraint. *Information Sciences*, 421:254–271, 2017.

[16] Sugato Basu et al. *Semi-supervised clustering: Learning with limited user feedback*. Computer Science Department, University of Texas at Austin, 2003.

[17] Hongfu Liu, Zhiqiang Tao, and Yun Fu. Partition level constrained clustering. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2469–2483, 2017.

[18] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[19] Christian Böhm and Claudia Plant. Hissclu: a hierarchical density-based method for semi-supervised clustering. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pages 440–451, 2008.

[20] Levi Lelis and Jörg Sander. Semi-supervised density-based clustering. In *2009 Ninth IEEE International Conference on Data Mining*, pages 842–847. IEEE, 2009.

[21] Haifeng Liu and Zhaohui Wu. Non-negative matrix factorization with constraints. In *Twenty-fourth AAAI conference on artificial intelligence*, 2010.

[22] Haifeng Liu, Zhaohui Wu, Xuelong Li, Deng Cai, and Thomas S Huang. Constrained nonnegative matrix factorization for image representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1299–1311, 2011.

[23] Kristóf Marussy and Krisztian Buza. Success: a new approach for semi-supervised classification of time-series. In *International conference on artificial intelligence and soft computing*, pages 437–447. Springer, 2013.

[24] Long Lan, Naiyang Guan, Xiang Zhang, Dacheng Tao, and Zhigang Luo. Soft-constrained nonnegative matrix factorization via normalization. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 3025–3030. IEEE, 2014.

[25] Kritsana Treechalong, Thanawin Rakthanmanon, and Kitsana Waiyamai. Semi-supervised stream clustering using labeled data points. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 281–295. Springer, 2015.

[26] Jaegul Choo, Changhyun Lee, Chandan K Reddy, and Haesun Park. Weakly supervised nonnegative matrix factorization for user-driven clustering. *Data mining and knowledge discovery*, 29(6):1598–1621, 2015.

[27] Viet-Vu Vu and Hong-Quan Do. Graph-based clustering with background knowledge. In *Proceedings of the Eighth International Symposium on Information and Communication Technology*, pages 167–172, 2017.

[28] Dino Ienco and Ruggero G Pensa. Semi-supervised clustering with multiresolution autoencoders. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[29] Chien-Liang Liu, Wen-Hoar Hsaio, Tao-Hsing Chang, and Hsuan-Hsun Li. Clustering data with partial background information. *International Journal of Machine Learning and Cybernetics*, 10(5):1123–1138, 2019.

[30] Jiarui Xie and Violaine Antoine. On a new evidential c-means algorithm with instance-level constraints. In *International Conference on Scalable Uncertainty Management*, pages 66–78. Springer, 2019.

[31] Lutz Herrmann and Alfred Ultsch. Label propagation for semi-supervised learning in self-organizing maps. In *International Workshop on Self-Organizing Maps: Proceedings (2007)*, 2007.

[32] Huaxiang Zhang and Jing Lu. Semi-supervised fuzzy clustering: a kernel-based approach. *Knowledge-Based Systems*, 22(6):477–481, 2009.

[33] Yun Yang and Xingchen Liu. A robust semi-supervised learning approach via mixture of label information. *Pattern Recognition Letters*, 68:15–21, 2015.

[34] Violaine Antoine, Kévin Gravouil, and Nicolas Labroche. On evidential clustering with partial supervision. In *International Conference on Belief Functions*, pages 14–21. Springer, 2018.

[35] Haitao Gan. Safe semi-supervised fuzzy c-means clustering. *IEEE Access*, 7:95659–95664, 2019.

[36] Jian-Ping Mei. Semisupervised fuzzy clustering with partition information of subsets. *IEEE Transactions on Fuzzy Systems*, 27(9):1726–1737, 2018.

[37] Violaine Antoine, Jose A Guerrero, and Jiarui Xie. Fast semi-supervised evidential clustering. *International Journal of Approximate Reasoning*, 133:116–132, 2021.

[38] David Gondek and Thomas Hofmann. Non-redundant data clustering. *Knowledge and Information Systems*, 12(1):1–24, 2007.

[39] Endo Yasunori, Hamasuna Yukihiro, Yamashiro Makito, and Miyamoto Sadaaki. On semi-supervised fuzzy c-means clustering. In *2009 IEEE International Conference on Fuzzy Systems*, pages 1119–1124. IEEE, 2009.

[40] Xuesong Yin, Ting Shu, and Qi Huang. Semi-supervised fuzzy clustering with metric learning and entropy regularization. *Knowledge-Based Systems*, 35:304–311, 2012.

[41] Shan Zeng, Xiaojun Tong, Nong Sang, and Rui Huang. A study on semi-supervised fcm algorithm. *Knowledge and information systems*, 35(3):585–612, 2013.

[42] Nguyen-Viet-Dung Nghiem, Christel Vrain, Thi-Bich-Hanh Dao, and Ian Davidson. Constrained clustering via post-processing. In *International Conference on Discovery Science*, pages 53–67. Springer, 2020.

[43] Hongjing Zhang, Sugato Basu, and Ian Davidson. A framework for deep constrained clustering-algorithms and advances. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 57–72. Springer, 2019.

[44] Marianne Mueller and Stefan Kramer. Integer linear programming models for constrained clustering. In *International Conference on Discovery Science*, pages 159–173. Springer, 2010.

[45] M Eduardo Ares, Javier Parapar, and Álvaro Barreiro. Avoiding bias in text clustering using constrained k-means and may-not-links. In *Conference on the Theory of Information Retrieval*, pages 322–329. Springer, 2009.

[46] Irene Diaz-Valenzuela, M Amparo Vila, and Maria J Martin-Bautista. On the use of fuzzy constraints in semisupervised clustering. *IEEE Transactions on Fuzzy Systems*, 24(4):992–999, 2015.

[47] He Jiang, Zhilei Ren, Jifeng Xuan, and Xindong Wu. Extracting elite pairwise constraints for clustering. *Neurocomputing*, 99:124–133, 2013.

[48] Yanshan Xiao, Bo Liu, and Zhifeng Hao. A maximum margin approach for semisupervised ordinal regression clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 27(5):1003–1019, 2015.

[49] Pengjiang Qian, Yizhang Jiang, Shitong Wang, Kuan-Hao Su, Jun Wang, Lingzhi Hu, and Raymond F Muzic. Affinity and penalty jointly constrained spectral clustering with all-compatibility, flexibility, and robustness. *IEEE transactions on neural networks and learning systems*, 28(5):1123–1138, 2016.

[50] Stella X Yu and Jianbo Shi. Segmentation given partial grouping constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):173–183, 2004.

[51] Martin HC Law, Alexander Topchy, and Anil K Jain. Clustering with soft and group constraints. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 662–670. Springer, 2004.

[52] Yuanli Pei, Xiaoli Z Fern, Teresa Vania Tjahja, and Rómer Rosales. Comparing clustering with pairwise and relative constraints: A unified framework. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(2):1–26, 2016.

[53] Eric Yi Liu, Zhaojun Zhang, and Wei Wang. Clustering with relative constraints. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–955, 2011.

[54] Ehsan Amid, Aristides Gionis, and Antti Ukkonen. A kernel-learning approach to semi-supervised clustering with relative distance comparisons. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 219–234. Springer, 2015.

[55] Dino Ienco and Ruggero G Pensa. Deep triplet-driven semi-supervised embedding clustering. In *International Conference on Discovery Science*, pages 220–234. Springer, 2019.

[56] Korinna Bade and Andreas Nürnberger. Hierarchical constraints. *Machine learning*, 94(3):371–399, 2014.

[57] Yasunori Endo and Yukihiro Hamasuna. Fuzzy c-means clustering with mutual relation constraints. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 131–140. Springer, 2011.

[58] María Teresa Gallegos and Gunter Ritter. Using combinatorial optimization in model-based trimmed clustering with cardinality constraints. *Computational Statistics & Data Analysis*, 54(3):637–654, 2010.

[59] Shunzhi Zhu, Dingding Wang, and Tao Li. Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889, 2010.

[60] Wei Tang, Yang Yang, Lanling Zeng, and Yongzhao Zhan. Size constrained clustering with milp formulation. *IEEE Access*, 8:1587–1599, 2019.

[61] Behrouz Babaki, Tias Guns, and Siegfried Nijssen. Constrained clustering using column generation. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 438–454. Springer, 2014.

[62] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. A declarative framework for constrained clustering. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 419–434. Springer, 2013.

[63] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, MSR-TR-2000-65, Microsoft Research, 2000.

[64] Kai Lei, Sibo Wang, Weiwei Song, and Qilin Li. Size-constrained clustering using an initial points selection method. In *International Conference on Knowledge Science, Engineering and Management*, pages 195–205. Springer, 2013.

[65] David Rebollo-Monedero, Marc Solé, Jordi Nin, and Jordi Forné. A modification of the k-means method for quasi-unsupervised learning. *Knowledge-Based Systems*, 37:176–185, 2013.

[66] Rong Ge, Martin Ester, Wen Jin, and Ian Davidson. Constraint-driven clustering. In *Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 320–329, 2007.

[67] Shi Zhong and Joydeep Ghosh. A unified framework for model-based clustering. *The Journal of Machine Learning Research*, 4:1001–1037, 2003.

[68] Arindam Banerjee and Joydeep Ghosh. Scalable clustering algorithms with balancing constraints. *Data Mining and Knowledge Discovery*, 13(3):365–395, 2006.

[69] Wei Tang, Yang Yang, Lanling Zeng, and Yongzhao Zhan. Optimizing mse for clustering with balanced size constraints. *Symmetry*, 11(3):338, 2019.

[70] Jun Sun, Wenbo Zhao, Jiangwei Xue, Zhiyong Shen, and Yidong Shen. Clustering with feature order preferences. *Intelligent Data Analysis*, 14(4):479–495, 2010.

[71] Ian Davidson, SS Ravi, and Leonid Shamis. A sat-based framework for efficient constrained clustering. In *Proceedings of the 2010 SIAM international conference on data mining*, pages 94–105. SIAM, 2010.

[72] Ian Davidson and SS Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 59–70. Springer, 2005.

[73] Yuanli Pei and Xiaoli Z Fern. Constrained instance clustering in multi-instance multi-label learning. *Pattern Recognition Letters*, 37:107–114, 2014.

[74] Pan Hu, Celine Vens, Bart Verstrynge, and Hendrik Blockeel. Generalizing from example clusters. In *International Conference on Discovery Science*, pages 64–78. Springer, 2013.

[75] Celine Vens, Bart Verstrynge, and Hendrik Blockeel. Semi-supervised clustering with example clusters. In *Proceedings of the 5th International Conference on Knowledge Discovery and Information Retrieval and the 5th International Conference on Knowledge Management and Information Sharing*, pages 45–51, 2013.

[76] Eya Ben Ahmed, Ahlem Nabli, and Faïez Gargouri. Shacun: Semi-supervised hierarchical active clustering based on ranking constraints. In *Industrial Conference on Data Mining*, pages 194–208. Springer, 2012.

[77] Eya Ben Ahmed, Ahlem Nabli, and Faïez Gargouri. Towards quantitative constraints ranking in data clustering. In *International Conference on Database and Expert Systems Applications*, pages 121–128. Springer, 2012.

[78] Bruno Magalhaes Nogueira, Yuri Karan Benevides Tomas, and Ricardo Marcondes Marcacini. Integrating distance metric learning and cluster-level constraints in semi-supervised clustering. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4118–4125. IEEE, 2017.

[79] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[80] Ian Davidson, Kiri L Wagstaff, and Sugato Basu. Measuring constraint-set utility for partitional clustering algorithms. In *European conference on principles of data mining and knowledge discovery*, pages 115–126. Springer, 2006.

[81] Ian Davidson and SS Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 International Conference on Data Mining*, pages 138–149. SIAM, 2005.

[82] Ian Davidson and Sekharipuram S Ravi. Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. *Data mining and knowledge discovery*, 18(2):257–282, 2009.

[83] Frank Harary. On the notion of balance of a signed graph. *Michigan Mathematical Journal*, 2(2):143–146, 1953.

[84] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. *AAAI/IAAI*, 1097:577–584, 2000.

[85] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1103–1110. Citeseer, 2000.

[86] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 577–584. Morgan Kaufmann Publishers Inc., 2001.

[87] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *Proceedings of the ICML-2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, pages 42–49. Citeseer, 2003.

[88] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, volume 15, page 12. Citeseer, 2002.

[89] Feng Li, Shoumei Li, and Thierry Denœux. k-cevclus: Constrained evidential clustering of large dissimilarity data. *Knowledge-Based Systems*, 142:29–44, 2018.

[90] Feiping Nie, Han Zhang, Rong Wang, and Xuelong Li. Semi-supervised clustering via pairwise constrained optimal graph. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3160–3166, 2021.

[91] Linlin Zong, Xianchao Zhang, and Xinyue Liu. Multi-view clustering on unmapped data via constrained non-negative matrix factorization. *Neural Networks*, 108:155–171, 2018.

[92] Thierry Denoeux. Nn-evclus: Neural network-based evidential clustering. *Information Sciences*, 572:297–330, 2021.

[93] Elham Amirizadeh and Reza Boostani. Cdec: a constrained deep embedded clustering. *International Journal of Intelligent Computing and Cybernetics*, 2021.

[94] Germán González-Almagro, Alejandro Rosales-Pérez, Julián Luengo, José-Ramón Cano, and Salvador García. Me-meoa/dcc: Multiobjective constrained clustering through decomposition-based memetic elitism. *Swarm and Evolutionary Computation*, 66:100939, 2021.

[95] Germán González-Almagro, Julián Luengo, José-Ramón Cano, and Salvador García. Dils: constrained clustering through dual iterative local search. *Computers & Operations Research*, page 104979, 2020.

[96] Ruizhang Huang and Wai Lam. Semi-supervised document clustering via active learning with pairwise constraints. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 517–522. IEEE, 2007.

[97] Ruizhang Huang and Wai Lam. An active learning framework for semi-supervised document clustering with language modeling. *Data & Knowledge Engineering*, 68(1):49–67, 2009.

[98] Huifang Ma, Weizhong Zhao, Qing Tan, and Zhongzhi Shi. Orthogonal nonnegative matrix tri-factorization for semi-supervised document co-clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 189–200. Springer, 2010.

[99] Yang Yan, Lihui Chen, and William-Chandra Tjhi. Fuzzy semi-supervised co-clustering for text documents. *Fuzzy Sets and Systems*, 215:74–89, 2013.

[100] Chien-Liang Liu, Wen-Hoar Hsaio, Chia-Hoang Lee, and Chun-Hsien Chen. Clustering tagged documents with labeled and unlabeled documents. *Information processing & management*, 49(3):596–606, 2013.

[101] Huifang Ma, Weizhong Zhao, and Zhongzhi Shi. A nonnegative matrix factorization framework for semi-supervised document clustering with dual constraints. *Knowledge and information systems*, 36(3):629–651, 2013.

[102] Taufik Sutanto and Richi Nayak. The ranking based constrained document clustering method and its application to social event detection. In *International Conference on Database Systems for Advanced Applications*, pages 47–60. Springer, 2014.

[103] Yeming Hu, Evangelos E Milios, and James Blustein. Document clustering with dual supervision through feature reweighting. *Computational Intelligence*, 32(3):480–513, 2016.

[104] Irene Diaz-Valenzuela, Vincenzo Loia, Maria J Martin-Bautista, Sabrina Senatore, and M Amparo Vila. Automatic constraints generation for semisupervised clustering: experiences with documents classification. *Soft Computing*, 20(6):2329–2339, 2016.

[105] Amine Trabelsi and Osmar R Zaïane. Mining contentious documents. *Knowledge and Information Systems*, 48(3):537–560, 2016.

[106] MA Balafar, R Hazratgholizadeh, and MRF Derakhshi. Active learning for constrained document clustering with uncertainty region. *Complexity*, 2020, 2020.

[107] Uraiwan Buatoom, Waree Kongprawechnon, and Thanaruk Theeramunkong. Document clustering using k-means with term weighting as similarity-based constraints. *Symmetry*, 12(6):967, 2020.

[108] Toon Van Craenendonck, Sebastijan Dumančic, and Hendrik Blockeel. Cobra: a fast and simple method for active clustering with pairwise constraints. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2871–2877, 2017.

[109] Guobiao Hu, Shuigeng Zhou, Jihong Guan, and Xiaohua Hu. Towards effective document clustering: A constrained k-means based approach. *Information Processing & Management*, 44(4):1397–1409, 2008.

[110] Yanhua Chen, Lijun Wang, and Ming Dong. Semi-supervised document clustering with simultaneous text representation and categorization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 211–226. Springer, 2009.

[111] Weizhong Zhao, Qing He, Huifang Ma, and Zhongzhi Shi. Effective semi-supervised document clustering via active learning with instance-level constraints. *Knowledge and information systems*, 30(3):569–587, 2012.

[112] Yang Yan, Lihui Chen, and William-Chandra Tjhi. Semi-supervised fuzzy coclustering algorithm for document categorization. *Knowledge and information systems*, 34(1):55–74, 2013.

[113] Jun Gu, Wei Feng, Jia Zeng, Hiroshi Mamitsuka, and Shanfeng Zhu. Efficient semisupervised medline document clustering with mesh-semantic and global-content constraints. *IEEE transactions on cybernetics*, 43(4):1265–1276, 2012.

[114] Javier Parapar and Álvaro Barreiro. Language modelling of constraints for text clustering. In *European Conference on Information Retrieval*, pages 352–363. Springer, 2012.

[115] Yangqiu Song, Shimei Pan, Shixia Liu, Furu Wei, Michelle X Zhou, and Weihong Qian. Constrained text coclustering with supervised and unsupervised constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1227–1239, 2012.

[116] Yan Zhu, Liping Jing, and Jian Yu. Text clustering via constrained nonnegative matrix factorization. In *2011 IEEE 11th International Conference on Data Mining*, pages 1278–1283. IEEE, 2011.

[117] Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. Unsupervised and constrained dirichlet process mixture models for verb clustering. In *Proceedings of the workshop on geometrical models of natural language semantics*, pages 74–82, 2009.

[118] Kazunari Sugiyama and Manabu Okumura. Semi-supervised clustering for word instances and its effect on word sense disambiguation. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 266–279. Springer, 2009.

[119] Huifang Ma, Meihuizi Jia, YaKai Shi, and Zhanjun Hao. Semi-supervised nonnegative matrix factorization for microblog clustering based on term correlation. In *Asia-Pacific Web Conference*, pages 511–516. Springer, 2014.

[120] Huifang Ma, Meihuizi Jia, Weizhong Zhao, and Xianghong Lin. Semi-supervised microblog clustering method via dual constraints. In *International Conference on Knowledge Science, Engineering and Management*, pages 360–369. Springer, 2015.

[121] Huifang Ma, Di Zhang, Meihuizi Jia, and Xianghong Lin. A term correlation based semi-supervised microblog clustering with dual constraints. *International Journal of Machine Learning and Cybernetics*, 10(4):679–692, 2019.

[122] Na Tang and V Rao Vemuri. User-interest-based document filtering via semi-supervised clustering. In *International Symposium on Methodologies for Intelligent Systems*, pages 573–582. Springer, 2005.

[123] Chuan Duan, Horatiu Dumitru, Jane Cleland-Huang, and Bamshad Mobasher. User-constrained clustering in online requirements forums. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 284–299. Springer, 2015.

[124] Vincent S Tseng, Lien-Chin Chen, and Ching-Pin Kao. Constrained clustering for gene expression data mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 759–766. Springer, 2008.

[125] Ioannis A Maraziotis. A semi-supervised fuzzy clustering algorithm applied to gene expression data. *Pattern Recognition*, 45(1):637–648, 2012.

[126] Sriparna Saha, Abhay Kumar Alok, and Asif Ekbal. Use of semisupervised clustering and feature-selection techniques for identification of co-expressed genes. *IEEE journal of biomedical and health informatics*, 20(4):1171–1177, 2015.

[127] Ioannis A Maraziotis, Andrei Dragomir, and Anastasios Bezerianos. Semi supervised fuzzy clustering networks for constrained analysis of time-series gene expression data. In *International Conference on Artificial Neural Networks*, pages 818–826. Springer, 2006.

[128] Zhiwen Yu, Hongsheng Chen, Jane You, Hau-San Wong, Jiming Liu, Le Li, and Guoqiang Han. Double selection based semi-supervised clustering ensemble for tumor clustering from gene expression profiles. *IEEE/ACM transactions on computational biology and bioinformatics*, 11(4):727–740, 2014.

[129] Michele Ceccarelli and Antonio Maratea. Semi-supervised fuzzy c-means clustering of biological data. In *International Workshop on Fuzzy Logic and Applications*, pages 259–266. Springer, 2005.

[130] Erliang Zeng, Chengyong Yang, Tao Li, and Giri Narasimhan. On the effectiveness of constraints sets in clustering genes. In *2007 IEEE 7th International Symposium on BioInformatics and BioEngineering*, pages 79–86. IEEE, 2007.

[131] Tian Tian, Jie Zhang, Xiang Lin, Zhi Wei, and Hakon Hakonarson. Model-based deep embedding for constrained clustering analysis of single cell rna-seq data. *Nature communications*, 12(1):1–12, 2021.

[132] Alok Mishra and Duncan Gillies. Semi supervised spectral clustering for regulatory module discovery. In *International Workshop on Data Integration in the Life Sciences*, pages 192–203. Springer, 2008.

[133] Daniel Duarte Abdala and Xiaoyi Jiang. Fiber segmentation using constrained clustering. In *International Conference on Medical Biometrics*, pages 1–10. Springer, 2010.

[134] Zhiwen Yu, Hau-San Wongb, Jane You, Qinmin Yang, and Hongying Liao. Knowledge based cluster ensemble for cancer discovery from biomolecular data. *IEEE transactions on nanobioscience*, 10(2):76–85, 2011.

[135] Hanuman Verma, RK Agrawal, and Aditi Sharan. An improved intuitionistic fuzzy c-means clustering algorithm incorporating local information for brain image segmentation. *Applied Soft Computing*, 46:543–557, 2016.

[136] Rodrigo Veras, Kelson Aires, Laurindo Britto, et al. Medical image segmentation using seeded fuzzy c-means: A semi-supervised clustering algorithm. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2018.

[137] Zhenzhou Wang and Yongming Yang. A non-iterative clustering based soft segmentation approach for a class of fuzzy images. *Applied Soft Computing*, 70:988–999, 2018.

[138] Jiaming Guo, Loong-Fah Cheong, Robby T Tan, and Steven Zhiying Zhou. Consistent foreground co-segmentation. In *Asian Conference on Computer Vision*, pages 241–257. Springer, 2014.

[139] Yajuan Li, Yue Liu, Bingde Cui, Chao Sun, Xiaoxuan Ji, Jing Zhang, Bufang Li, Huanhuan Chen, Jianwu Zhang, Yalei Wang, et al. Spatial constrained k-means for image segmentation. In *Proceedings of SAI Intelligent Systems Conference*, pages 662–672. Springer, 2020.

[140] Arijit Biswas and David Jacobs. Large scale image clustering with active pairwise constraints. In *International Conference in Machine Learning 2011 Workshop on Combining Learning Strategies to Reduce Label Cost*, volume 2. Citeseer, 2011.

[141] Steven CH Hoi, Wei Liu, and Shih-Fu Chang. Semi-supervised distance metric learning for collaborative image retrieval and clustering. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 6(3):1–26, 2010.

[142] Arijit Biswas and David Jacobs. Active image clustering with pairwise constraints from humans. *International Journal of Computer Vision*, 108(1):133–147, 2014.

[143] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Fuzzy clustering with pairwise constraints for knowledge-driven image categorisation. *IEE Proceedings-Vision, Image and Signal Processing*, 153(3):299–304, 2006.

[144] Hichem Frigui and Jason Meredith. Image database categorization under spatial constraints using adaptive constrained clustering. In *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, pages 2268–2276. IEEE, 2008.

[145] MOHAMED MAHER BEN ISMAIL and Ouiem Bchir. Automatic image annotation based on semi-supervised clustering and membership-based cross media relevance model. *International Journal of Pattern Recognition and Artificial Intelligence*, 26(06):1255009, 2012.

[146] Rui Xiaoguang, Yuan Pingbo, and Yu Nenghai. Image annotations based on semi-supervised clustering with semantic soft constraints. In *Pacific-Rim Conference on Multimedia*, pages 624–632. Springer, 2006.

[147] Hien Phuong Lai, Muriel Visani, Alain Boucher, and Jean-Marc Ogier. A new interactive semi-supervised clustering model for large image database indexing. *Pattern Recognition Letters*, 37:94–106, 2014.

[148] Thanh-Hieu Bui and Seong-Bae Park. Point of interest mining with proper semantic annotation. *Multimedia Tools and Applications*, 76(22):23435–23457, 2017.

[149] Tiancheng Li, Fernando De la Prieta Pintado, Juan M Corchado, and Javier Bajo. Multi-source homogeneous data clustering for multi-target detection from cluttered background with misdetection. *Applied Soft Computing*, 60:436–446, 2017.

[150] Viet-Vu Vu, Hong-Quan Do, Vu-Tuan Dang, and Nang-Toan Do. An efficient density-based clustering with side information and active learning: a case study for facial expression recognition task. *Intelligent Data Analysis*, 23(1):227–240, 2019.

[151] Baptiste Lafabregue, Jonathan Weber, Pierre Gançarski, and Germain Forestier. Deep constrained clustering applied to satellite image time series. In *ECML/PKDD Workshop on Machine Learning for Earth Observation Data (MACLEAN)*, Würzburg, Germany, 2019.

[152] Juan Carlos Niebles, Bohyung Han, Andras Ferencz, and Li Fei-Fei. Extracting moving people from internet videos. In *European conference on computer vision*, pages 527–540. Springer, 2008.

[153] Amjad Mahmood, Tianrui Li, Yan Yang, and Hongjun Wang. Semi-supervised clustering ensemble evolved by genetic algorithm for web video categorization. In *International Conference on Advanced Data Mining and Applications*, pages 1–12. Springer, 2013.

[154] Baoyuan Wu, Yifan Zhang, Bao-Gang Hu, and Qiang Ji. Constrained clustering and its application to face clustering in videos. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3507–3514, 2013.

[155] Baoyuan Wu, Bao-Gang Hu, and Qiang Ji. A coupled hidden markov random field model for simultaneous face clustering and tracking in videos. *Pattern Recognition*, 64:361–373, 2017.

[156] Baoyuan Wu, Siwei Lyu, Bao-Gang Hu, and Qiang Ji. Simultaneous clustering and tracklet linking for multi-face tracking in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 2856–2863, 2013.

[157] Yanlu Xie, Minghui Liu, Zhiqiang Yao, and Beiqian Dai. Improved two-stage wiener filter for robust speaker identification. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 4, pages 310–313. IEEE, 2006.

[158] Chuan Duan, Jane Cleland-Huang, and Bamshad Mobasher. A consensus based approach to constrained clustering of software requirements. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1073–1082, 2008.

[159] Nan Li, Yanming Li, Haining Liu, and Chengliang Liu. An evolving machinery fault diagnosis approach based on affinity propagation algorithm. In *International Conference on Intelligent Robotics and Applications*, pages 654–664. Springer, 2010.

[160] Longfei Han, Senlin Luo, Huaiqing Wang, Limin Pan, Xincheng Ma, and Tiemei Zhang. An intelligible risk stratification model based on pairwise and size constrained kmeans. *IEEE journal of biomedical and health informatics*, 21(5):1288–1296, 2016.

[161] Jiao Zhang and Dan Chang. Semi-supervised patient similarity clustering algorithm based on electronic medical records. *IEEE Access*, 7:90705–90714, 2019.

[162] Hui-Chu Chang and Hsiao-Ping Tsai. Group rfm analysis as a novel framework to discover better customer consumption behavior. *Expert Systems with Applications*, 38(12):14499–14513, 2011.

[163] Su-min Yu, Zhi-jiao Du, Xueyang Zhang, Hanyang Luo, and Xudong Lin. Trust copkmeans clustering analysis and minimum-cost consensus model considering voluntary trust loss in social network large-scale decision-making. *IEEE Transactions on Fuzzy Systems*, 2021.

[164] Alex Seret, Thomas Verbraken, and Bart Baesens. A new knowledge-based constrained clustering approach: Theory and application in direct marketing. *Applied Soft Computing*, 24:316–327, 2014.

[165] Ines Akaichi and Patrice Wislez. Pairwise constrained clustering and robust regression: A case study on french enterprise activities and expenses data. In *2021 International Conference of Women in Data Science at Taif University (WiDSTaif)*, pages 1–7. IEEE, 2021.

[166] Eya Ben Ahmed, Ahlem Nabli, and Faïez Gargouri. Group extraction from professional social network using a new semi-supervised hierarchical clustering. *Knowledge and information systems*, 40(1):29–47, 2014.

[167] Ghalib A Shah, Fatih Alagoz, Etimad A Fadel, and Ozgur B Akan. A spectrum-aware clustering for efficient multimedia routing in cognitive radio sensor networks. *IEEE Transactions on Vehicular Technology*, 63(7):3369–3380, 2014.

[168] Deepti Joshi, Leen-Kiat Soh, and Ashok Samal. Redistricting using constrained polygonal clustering. *IEEE Transactions on Knowledge and Data Engineering*, 24(11):2065–2079, 2011.

[169] Jie Song, Hanfa Xing, Huanxue Zhang, Yuetong Xu, and Yuan Meng. An adaptive network-constrained clustering (ancc) model for fine-scale urban functional zones. *IEEE Access*, 9:53013–53029, 2021.

[170] Rodrigo Araujo and Mohamed S Kamel. Audio-visual emotion analysis using semi-supervised temporal clustering with constraint propagation. In *International Conference Image Analysis and Recognition*, pages 3–11. Springer, 2014.

[171] Shufeng Xiong and Donghong Ji. Exploiting capacity-constrained k-means clustering for aspect-phrase grouping. In *International Conference on Knowledge Science, Engineering and Management*, pages 370–381. Springer, 2015.

[172] Caglar Tirkaz, Berrin Yanikoglu, and T Metin Sezgin. Sketched symbol recognition with auto-completion. *Pattern Recognition*, 45(11):3926–3937, 2012.

[173] Samaneh Hosseini Semnani, Otman A Basir, and Peter Van Beek. Constrained clustering for flocking-based tracking in maneuvering target environment. *Robotics and Autonomous Systems*, 83:243–250, 2016.

[174] Firas Saidi, Zouheir Trabelsi, and Henda Ben Ghazela. A novel approach for terrorist sub-communities detection based on constrained evidential clustering. In *2018 12th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–8. IEEE, 2018.

[175] Yan Zhao and Zhiyi Gai. Transformation and optimization of rural ecological endowment industry chain based on constrained clustering algorithm. *Scientific Programming*, 2022, 2022.

[176] Mohammed El-Kholany, Konstantin Schekotihin, and Martin Gebser. Decomposition-based job-shop scheduling with constrained clustering. In *International Symposium on Practical Aspects of Declarative Languages*, pages 165–180. Springer, 2022.

[177] Pieter De Koninck, Klaas Nelissen, Bart Baesens, Monique Snoeck, Jochen De Weerdt, et al. Expert-driven trace clustering with instance-level constraints. *Knowledge and Information Systems*, 63(5):1197–1220, 2021.

[178] Yingying Zhang, Volodymyr Melnykov, and Igor Melnykov. Semi-supervised clustering of time-dependent categorical sequences with application to discovering education-based life patterns. *Statistical Modelling*, page 1471082X21989170, 2021.

[179] Tue Boesen, Eldad Haber, and G Michael Hoversten. Data-driven semi-supervised clustering for oil prediction. *Computers & Geosciences*, 148:104684, 2021.

[180] Yongzheng Zhang, Shuyuan Zhao, and Yafei Sang. Towards unknown traffic identification using deep auto-encoder and constrained clustering. In *International Conference on Computational Science*, pages 309–322. Springer, 2019.

[181] Claudia Malzer and Marcus Baum. Constraint-based hierarchical cluster selection in automotive radar data. *Sensors*, 21(10):3410, 2021.

[182] Safa Ben Ayed, Zied Elouedi, and Eric Lefevre. Cevm: constrained evidential vocabulary maintenance policy for cbr systems. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 579–592. Springer, 2019.

[183] Son T Mai, Sihem Amer-Yahia, Sébastien Bailly, Jean-Louis Pépin, Ahlame Douzal Chouakria, Ky T Nguyen, and Anh-Duong Nguyen. Evolutionary active constrained clustering for obstructive sleep apnea analysis. *Data Science and Engineering*, 3(4):359–378, 2018.

[184] Yu Wang, Yang Xiang, Jun Zhang, Wanlei Zhou, Guiyi Wei, and Laurence T Yang. Internet traffic classification using constrained clustering. *IEEE transactions on parallel and distributed systems*, 25(11):2932–2943, 2013.

[185] Tetsuya Yoshida. Performance evaluation of constraints in graph-based semi-supervised clustering. In *International Conference on Active Media Technology*, pages 138–149. Springer, 2010.

[186] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning distance functions using equivalence relations. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 11–18, 2003.

[187] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, Daphna Weinshall, and Greg Ridgeway. Learning a mahalanobis metric from equivalence constraints. *Journal of machine learning research*, 6(6), 2005.

[188] Hao Cheng, Kien A Hua, and Khanh Vu. Constrained locally weighted clustering. *Proceedings of the VLDB Endowment*, 1(1):90–101, 2008.

[189] Sugato Basu, Arindam Banerjee, and Raymond J Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 333–344. SIAM, 2004.

[190] Germán González-Almagro, Juan Luis Suárez, Julián Luengo, José-Ramón Cano, and Salvador García. 3shacc: Three stages hybrid agglomerative constrained clustering. *Neurocomputing*, 2021.

[191] Viet-Vu Vu, Nicolas Labroche, and Bernadette Bouchon-Meunier. Improving constrained clustering with active query selection. *Pattern Recognition*, 45(4):1749–1758, 2012.

[192] Ping He, Xiaohua Xu, Kongfa Hu, and Ling Chen. Semi-supervised clustering via multi-level random walk. *Pattern recognition*, 47(2):820–832, 2014.

[193] Yanchao Li, Yongli Wang, Dong-Jun Yu, Ning Ye, Peng Hu, and Ruxin Zhao. Ascent: Active supervision for semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(5):868–882, 2019.

[194] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68, 2004.

[195] Byron E Dom. An information-theoretic external cluster-validity measure. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 137–145, 2002.

[196] Han Hu, Jianjiang Feng, and Jie Zhou. Exploiting unsupervised and supervised constraints for subspace clustering. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1542–1557, 2014.

[197] Zhenguo Li, Jianzhuang Liu, and Xiaoou Tang. Constrained clustering via spectral regularization. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 421–428. IEEE, 2009.

[198] Zhenguo Li and Jianzhuang Liu. Constrained clustering by spectral kernel learning. In *2009 IEEE 12th International Conference on Computer Vision*, pages 421–427. IEEE, 2009.

[199] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[200] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[201] Guoxiang Zhong, Xiuqin Deng, and Shengbing Xu. Active informative pairwise constraint formulation algorithm for constraint-based clustering. *IEEE Access*, 7:81983–81993, 2019.

[202] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[203] George Hripcsak and Adam S Rothschild. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American medical informatics association*, 12(3):296–298, 2005.

[204] Qi Zhao and David J Miller. Mixture modeling with pairwise, instance-level class constraints. *Neural computation*, 17(11):2482–2507, 2005.

[205] Maria Halkidi, Myra Spiliopoulou, and Aikaterini Pavlou. A semi-supervised incremental clustering algorithm for streaming data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 578–590. Springer, 2012.

[206] Xiaohua Xu, Zhoujin Pan, Ping He, and Ling Chen. Constrained clustering via swarm intelligence. In *International Conference on Intelligent Computing*, pages 404–409. Springer, 2011.

[207] Eric Bae and James Bailey. Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 53–62. IEEE, 2006.

[208] Marina Meilă. Comparing clusterings by the variation of information. In *Learning theory and kernel machines*, pages 173–187. Springer, 2003.

[209] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of intelligent information systems*, 17(2):107–145, 2001.

[210] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[211] Andreas Vlachos, Zoubin Ghahramani, and Anna Korhonen. Dirichlet process mixture models for verb clustering. In *Proceedings of the ICML workshop on Prior Knowledge for Text and Language*, 2008.

[212] Shifei Ding, Hongjie Jia, Mingjing Du, and Yu Xue. A semi-supervised approximate spectral clustering algorithm based on hmrf model. *Information Sciences*, 429:215–228, 2018.

[213] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[214] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[215] Wei Tang, Hui Xiong, Shi Zhong, and Jie Wu. Enhancing semi-supervised clustering: a feature projection perspective. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 707–716, 2007.

[216] Carlotta Domeniconi and Muna Al-Razgan. Penta-training: Clustering ensembles with bootstrapping of constraints. In *Workshop on Supervised and Unsupervised Ensemble Methods and their Applications*, page 47, 2008.

[217] Haichao Huang, Yong Cheng, and Ruilian Zhao. A semi-supervised clustering algorithm based on must-link set. In *International Conference on Advanced Data Mining and Applications*, pages 492–499. Springer, 2008.

[218] Kai Rothaus and Xiaoyi Jiang. Constrained clustering by a novel graph-based distance transformation. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[219] Shaohong Zhang and Hau-San Wong. Partial closure-based constrained clustering with order ranking. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[220] Oncel Tuzel, Fatih Porikli, and Peter Meer. Kernel methods for weakly supervised mean shift clustering. In *2009 IEEE 12th International Conference on Computer Vision*, pages 48–55. IEEE, 2009.

[221] Xuesong Yin, Songcan Chen, Enliang Hu, and Daoqiang Zhang. Semi-supervised clustering with metric learning: An adaptive kernel method. *Pattern Recognition*, 43(4):1320–1333, 2010.

[222] WEI TAN, YAN YANG, and TIANRUI LI. An improved cop-kmeans algorithm for solving constraint violation. In *Computational Intelligence: Foundations and Applications*, pages 690–696. World Scientific, 2010.

[223] Tonny Rutayisire, Yan Yang, Chao Lin, and Jinyuan Zhang. A modified cop-kmeans algorithm based on sequenced cannot-link set. In *International Conference on Rough Sets and Knowledge Technology*, pages 217–225. Springer, 2011.

[224] Saket Anand, Sushil Mittal, Oncel Tuzel, and Peter Meer. Semi-supervised kernel mean shift clustering. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1201–1215, 2013.

[225] Huu M Le, Anders Eriksson, Thanh-Toan Do, and Michael Milford. A binary optimization approach for constrained k-means clustering. In *Asian Conference on Computer Vision*, pages 383–398. Springer, 2018.

[226] Rodrigo Randel, Daniel Aloise, Nenad Mladenović, and Pierre Hansen. On the k-medoids model for semi-supervised clustering. In *International Conference on Variable Neighborhood Search*, pages 13–27. Springer, 2018.

[227] Kiri Lou Wagstaff. *Intelligent clustering with instance-level constraints*. Cornell University USA, 2002.

[228] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 11, 2004.

[229] Eric Robert Eaton. *Clustering with propagated constraints*. PhD thesis, University of Maryland, Baltimore County, 2005.

[230] Dan Pelleg and Dorit Baras. K-means with large and noisy constraint sets. In *European Conference on Machine Learning*, pages 674–682. Springer, 2007.

[231] Hichem Frigui and Rami Mahdi. Semi-supervised clustering and feature discrimination with instance-level constraints. In *2007 IEEE International Fuzzy Systems Conference*, pages 1–6. IEEE, 2007.

[232] Yasunori Endo, Naohiko Kinoshita, Kuniaki Iwakura, and Yukihiro Hamasuna. Hard and fuzzy c-means algorithms with pairwise constraints by non-metric terms. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 145–157. Springer, 2014.

[233] Shaohong Zhang, Hau-San Wong, and Dongqing Xie. Semi-supervised clustering with pairwise and size constraints. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2450–2457. IEEE, 2014.

[234] Marian-Andrei Rizoiu, Julien Velcin, and Stéphane Lallich. How to use temporal-driven constrained clustering to detect typical evolutions. *International Journal on Artificial Intelligence Tools*, 23(04):1460013, 2014.

[235] Siting Wei, Zhixin Li, and Canlong Zhang. A semi-supervised clustering ensemble approach integrated constraint-based and metric-based. In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*, pages 1–6, 2015.

[236] Siting Wei, Zhixin Li, and Canlong Zhang. Combined constraint-based with metric-based in semi-supervised clustering ensemble. *International Journal of Machine Learning and Cybernetics*, 9(7):1085–1100, 2018.

[237] Son T Mai, Sihem Amer-Yahia, Ahlame Douzal Chouakria, Ky T Nguyen, and Anh-Duong Nguyen. Scalable active constrained clustering for temporal data. In *International Conference on Database Systems for Advanced Applications*, pages 566–582. Springer, 2018.

[238] Avgoustinos Vouros and Eleni Vasilaki. A semi-supervised sparse k-means algorithm. *Pattern Recognition Letters*, 142:65–71, 2021.

[239] Guoliang He, Yanzhou Pan, Xuewen Xia, Jinrong He, Rong Peng, and Neal N Xiong. A fast semi-supervised clustering framework for large-scale time series data. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(7):4201–4216, 2019.

[240] Liang Yang, Xiaochun Cao, Di Jin, Xiao Wang, and Dan Meng. A unified semi-supervised community detection framework using latent space graph regularization. *IEEE transactions on cybernetics*, 45(11):2585–2598, 2014.

[241] Carlotta Domeniconi, Jing Peng, and Bojun Yan. Composite kernels for semi-supervised clustering. *Knowledge and information systems*, 28(1):99–116, 2011.

[242] Haytham Elghazel, Khalid Benabdeslem, and Alain Dussauchoy. Constrained graph b-coloring based clustering approach. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 262–271. Springer, 2007.

[243] Tom Coleman, James Saunderson, and Anthony Wirth. A local-search 2-approximation for 2-correlation-clustering. In *European Symposium on Algorithms*, pages 308–319. Springer, 2008.

[244] Tetsuya Yoshida and Kazuhiro Okatani. A graph-based projection approach for semi-supervised clustering. In *Pacific Rim Knowledge Acquisition Workshop*, pages 1–13. Springer, 2010.

[245] Tetsuya Yoshida. Pairwise constraint propagation for graph-based semi-supervised clustering. In *International Symposium on Methodologies for Intelligent Systems*, pages 358–364. Springer, 2011.

[246] Tetsuya Yoshida. A graph-based approach for semisupervised clustering. *Computational Intelligence*, 30(2):263–284, 2014.

[247] Rajul Anand and Chandan K Reddy. Graph-based clustering with constraints. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 51–62. Springer, 2011.

[248] Kamvar Kamvar, Sepandar Sepandar, Klein Klein, Dan Dan, Manning Manning, and Christopher Christopher. Spectral learning. In *International Joint Conference of Artificial Intelligence*. Stanford InfoLab, 2003.

[249] Tijl De Bie, Johan Suykens, and Bart De Moor. Learning from general label constraints. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 671–679. Springer, 2004.

[250] Qianjun Xu, Kiri L Wagstaff, et al. Active constrained clustering by examining spectral eigenvectors. In *International Conference on Discovery Science*, pages 294–307. Springer, 2005.

[251] Qianjun Xu, Marie Desjardins, and Kiri Wagstaff. Constrained spectral clustering under a local proximity structure assumption. In *In FLAIRS Conference*. Citeseer, 2005.

[252] Zhengdong Lu and Miguel A Carreira-Perpinan. Constrained spectral clustering through affinity propagation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[253] Carlos Alzate and Johan AK Suykens. A regularized formulation for spectral clustering with pairwise constraints. In *2009 International Joint Conference on Neural Networks*, pages 141–148. IEEE, 2009.

[254] Linli Xu, Wenye Li, and Dale Schuurmans. Fast normalized cut with linear constraints. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2866–2873. IEEE, 2009.

[255] Xiang Wang and Ian Davidson. Flexible constrained spectral clustering. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 563–572, 2010.

[256] Xiang Wang and Ian Davidson. Active spectral clustering. In *2010 IEEE International Conference on Data Mining*, pages 561–568. IEEE, 2010.

[257] LC Jiao, Fanhua Shang, Fei Wang, and Yuanyuan Liu. Fast semi-supervised clustering with enhanced spectral embedding. *Pattern Recognition*, 45(12):4358–4369, 2012.

[258] Fabian L Wauthier, Nebojsa Jojic, and Michael I Jordan. Active spectral clustering via iterative uncertainty reduction. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1339–1347, 2012.

[259] Fanhua Shang, LC Jiao, Yuanyuan Liu, and Fei Wang. Learning spectral embedding via iterative eigenvalue thresholding. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1507–1511, 2012.

[260] Weifu Chen and Guocan Feng. Spectral clustering: a semi-supervised approach. *Neurocomputing*, 77(1):229–242, 2012.

[261] Syama Sundar Rangapuram and Matthias Hein. Constrained 1-spectral clustering. In *Artificial Intelligence and Statistics*, pages 1143–1151. PMLR, 2012.

[262] Shifei Ding, Hongjie Jia, Liwen Zhang, and Fengxiang Jin. Research of semi-supervised spectral clustering algorithm based on pairwise constraints. *Neural Computing and Applications*, 24(1):211–219, 2014.

[263] Xiang Wang, Buyue Qian, and Ian Davidson. On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery*, 28(1):1–30, 2014.

[264] Hongjun Wang, Tao Li, Tianrui Li, and Yan Yang. Constraint neighborhood projections for semi-supervised clustering. *IEEE transactions on cybernetics*, 44(5):636–643, 2014.

[265] Xiang Wang, Jun Wang, Buyue Qian, Fei Wang, and Ian Davidson. Self-taught spectral clustering via constraint augmentation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 416–424. SIAM, 2014.

[266] Mihai Cucuringu, Ioannis Koutis, Sanjay Chawla, Gary Miller, and Richard Peng. Simple and scalable constrained clustering: a generalized spectral method. In *Artificial Intelligence and Statistics*, pages 445–454. PMLR, 2016.

[267] Caiming Xiong, David M Johnson, and Jason J Corso. Active clustering with model-based uncertainty reduction. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):5–17, 2016.

[268] Chengming Jiang, Huiqing Xie, and Zhaojun Bai. Robust and efficient computation of eigenvectors in a generalized spectral method for constrained clustering. In *Artificial Intelligence and Statistics*, pages 757–766. PMLR, 2017.

[269] Chuan Chen, Hui Qian, Wuhui Chen, Zibin Zheng, and Hong Zhu. Auto-weighted multi-view constrained spectral clustering. *Neurocomputing*, 366:1–11, 2019.

[270] Jialin Tian, Yazhou Ren, and Xiang Cheng. Stratified feature sampling for semi-supervised ensemble clustering. *IEEE Access*, 7:128669–128675, 2019.

[271] Xianchao Zhang, Linlin Zong, Xinyue Liu, and Hong Yu. Constrained nmf-based multi-view clustering on unmapped data. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[272] Tao Li, Chris Ding, and Michael I Jordan. Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 577–582. IEEE, 2007.

[273] Yanhua Chen, Manjeet Rege, Ming Dong, and Jing Hua. Non-negative matrix factorization for semi-supervised data clustering. *Knowledge and Information Systems*, 17(3):355–379, 2008.

[274] Yanhua Chen, Lijun Wang, and Ming Dong. Non-negative matrix factorization for semisupervised heterogeneous data coclustering. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1459–1474, 2009.

[275] Liping Jing, Jian Yu, Tieyong Zeng, and Yan Zhu. Semi-supervised clustering via constrained symmetric non-negative matrix factorization. In *International Conference on Brain Informatics*, pages 309–319. Springer, 2012.

[276] Di Wang, Xinbo Gao, and Xiumei Wang. Semi-supervised nonnegative matrix factorization via constraint propagation. *IEEE transactions on cybernetics*, 46(1):233–244, 2015.

[277] Xianchao Zhang, Linlin Zong, Xinyue Liu, and Jiebo Luo. Constrained clustering with nonnegative matrix factorization. *IEEE transactions on neural networks and learning systems*, 27(7):1514–1526, 2015.

[278] Wenhui Wu, Yuheng Jia, Sam Kwong, and Junhui Hou. Pairwise constraint propagation-induced symmetric nonnegative matrix factorization. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):6348–6361, 2018.

[279] Burr Settles. Active learning literature survey. 2009.

[280] Derek Greene and Pádraig Cunningham. Constraint selection by committee: An ensemble approach to identifying informative constraints for semi-supervised clustering. In *European Conference on Machine Learning*, pages 140–151. Springer, 2007.

[281] Pavan Kumar Mallapragada, Rong Jin, and Anil K Jain. Active query selection for semi-supervised clustering. In *2008 19Th international conference on pattern recognition*, pages 1–4. IEEE, 2008.

[282] Viet-Vu Vu, Nicolas Labroche, and Bernadette Bouchon-Meunier. An efficient active constraint selection algorithm for clustering. In *2010 20th International Conference on Pattern Recognition*, pages 2969–2972. IEEE, 2010.

[283] Viet-Vu Vu, Nicolas Labroche, and Bernadette Bouchon-Meunier. Boosting clustering by active constraint selection. In *ECAI*, volume 10, pages 297–302, 2010.

[284] Kais Allab and Khalid Benabdeslem. Constraint selection for semi-supervised topological clustering. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 28–43. Springer, 2011.

[285] Hongjun Wang, Tao Li, Tianrui Li, and Yan Yang. Exemplars-constraints for semi-supervised clustering. In *International Conference on Advanced Data Mining and Applications*, pages 115–126. Springer, 2012.

[286] João MM Duarte, Ana LN Fred, and Fernando Jorge F Duarte. A constraint acquisition method for data clustering. In *Iberoamerican Congress on Pattern Recognition*, pages 108–116. Springer, 2013.

[287] Ahmad Ali Abin and Hamid Beigy. Active selection of clustering constraints: a sequential approach. *Pattern Recognition*, 47(3):1443–1458, 2014.

[288] Walid Atwa and Kan Li. Active query selection for constraint-based clustering algorithms. In *International Conference on Database and Expert Systems Applications*, pages 438–445. Springer, 2014.

[289] Yinghui Yang, Zijie Qi, and Hongyan Liu. Selective domain information acquisition to improve segmentation quality. In *Proceedings of the 17th International Conference on Electronic Commerce 2015*, pages 1–8, 2015.

[290] Chin-Chun Chang and Po-Yi Lin. Active learning for semi-supervised clustering based on locally linear propagation reconstruction. *Neural Networks*, 63:170–184, 2015.

[291] Lijun Cai, Tinghao Yu, Tingqin He, Lei Chen, and Meiqi Lin. Active learning method for constraint-based clustering algorithms. In *International Conference on Web-Age Information Management*, pages 319–329. Springer, 2016.

[292] Ahmad Ali Abin. Querying beneficial constraints before clustering using facility location analysis. *IEEE transactions on cybernetics*, 48(1):312–323, 2016.

[293] Ahmad Ali Abin. A random walk approach to query informative constraints for clustering. *IEEE Transactions on Cybernetics*, 48(8):2272–2283, 2017.

[294] Ahmad Ali Abin. Querying informative constraints for data clustering: An embedding approach. *Applied Soft Computing*, 80:31–41, 2019.

[295] Ahmad Ali Abin and Viet-Vu Vu. A density-based approach for querying informative constraints for clustering. *Expert Systems with Applications*, 161:113690, 2020.

[296] Duo Wen Chen and Ying Hua Jin. An active learning algorithm based on shannon entropy for constraint-based clustering. *IEEE Access*, 8:171447–171456, 2020.

[297] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Active semi-supervised fuzzy clustering. *Pattern Recognition*, 41(5):1834–1844, 2008.

[298] Swapna Raj Prabakara Raj and Balaraman Ravindran. Incremental constrained clustering: A decision theoretic approach. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 475–486. Springer, 2013.

[299] Vidyadhar Rao and CV Jawahar. Semi-supervised clustering by selecting informative constraints. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 213–221. Springer, 2013.

[300] Sicheng Xiong, Javad Azimi, and Xiaoli Z Fern. Active learning of constraints for semi-supervised clustering. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):43–54, 2013.

[301] Ahmad Ali Abin and Hamid Beigy. Active constrained fuzzy clustering: A multiple kernels learning approach. *Pattern Recognition*, 48(3):953–967, 2015.

[302] Ahmad Ali Abin. Clustering with side information: Further efforts to improve efficiency. *Pattern Recognition Letters*, 84:252–258, 2016.

[303] Toon Van Craenendonck and Hendrik Blockeel. Constraint-based clustering selection. *Machine Learning*, 106(9):1497–1521, 2017.

[304] Toon Van Craenendonck, Sebastijan Dumančić, Elia Van Wolputte, and Hendrik Blockeel. Cobras: Interactive clustering with pairwise queries. In *International Symposium on Intelligent Data Analysis*, pages 353–366. Springer, 2018.

[305] Matheus Campos Fernandes, Thiago Ferreira Covões, and André Luiz Vizine Pereira. Active learning for evolutionary constrained clustering. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 162–167. IEEE, 2019.

[306] Matheus Campos Fernandes, Thiago Ferreira Covões, and André Luiz Vizine Pereira. Improving evolutionary constrained clustering using active learning. *Knowledge-Based Systems*, 209:106452, 2020.

[307] Yifan Shi, Zhiwen Yu, Wenming Cao, CL Philip Chen, Hau-San Wong, and Guoqiang Han. Fast and effective active clustering ensemble based on density peak. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3593–3607, 2020.

[308] Khalid Benabdeslem and Jihene Snoussi. A probabilistic approach for constrained clustering with topological map. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 413–426. Springer, 2009.

[309] Amin Allahyar, Hadi Sadoghi Yazdi, and Ahad Harati. Constrained semi-supervised growing self-organizing map. *Neurocomputing*, 147:456–471, 2015.

[310] Yazhou Ren, Kangrong Hu, Xinyi Dai, Lili Pan, Steven CH Hoi, and Zenglin Xu. Semi-supervised deep embedded clustering. *Neurocomputing*, 325:121–130, 2019.

[311] Hongjing Zhang, Tianyang Zhan, Sugato Basu, and Ian Davidson. A framework for deep constrained clustering. *Data Mining and Knowledge Discovery*, 35(2):593–620, 2021.

[312] Marco Maggini, Stefano Melacci, and Lorenzo Sarti. Learning from pairwise constraints by similarity neural networks. *Neural Networks*, 26:141–158, 2012.

[313] Marek Śmieja, Łukasz Struski, and Mário AT Figueiredo. A classification-based approach to semi-supervised clustering with pairwise constraints. *Neural Networks*, 127:193–203, 2020.

[314] Yi Cui, Xianchao Zhang, Linlin Zong, and Jie Mu. Maintaining consistency with constraints: A constrained deep clustering method. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 219–230. Springer, 2021.

[315] Bojun Yan and Carlotta Domeniconi. Subspace metric ensembles for semi-supervised clustering of high dimensional data. In *European Conference on Machine Learning*, pages 509–520. Springer, 2006.

[316] Germain Forestier, Pierre Gançarski, and Cédric Wemmert. Collaborative clustering with background knowledge. *Data & Knowledge Engineering*, 69(2):211–228, 2010.

[317] Ashraf Mohammed Iqbal, Abidalrahman Moh'd, and Zahoor Khan. Semi-supervised clustering ensemble by voting. *arXiv preprint arXiv:1208.4138*, 2012.

[318] Dahai Chen, Yan Yang, Hongjun Wang, and Amjad Mahmood. Convergence analysis of semi-supervised clustering ensemble. In *2013 IEEE Third International Conference on Information Science and Technology (ICIST)*, pages 783–788. IEEE, 2013.

[319] Zhiwu Lu and Yuxin Peng. Exhaustive and efficient constraint propagation: A graph-based learning approach and its applications. *International journal of computer vision*, 103(3):306–325, 2013.

[320] Jinfeng Yi, Lijun Zhang, Tianbao Yang, Wei Liu, and Jun Wang. An efficient semi-supervised clustering algorithm with sequential constraints. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1405–1414, 2015.

[321] Zhiwen Yu, Peinan Luo, Jane You, Hau-San Wong, Hareton Leung, Si Wu, Jun Zhang, and Guoqiang Han. Incremental semi-supervised clustering ensemble for high dimensional data clustering. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):701–714, 2015.

[322] Fan Yang, Tao Li, Qifeng Zhou, and Han Xiao. Cluster ensemble selection with constraints. *Neurocomputing*, 235:59–70, 2017.

[323] Zhiwen Yu, Peinan Luo, Jiming Liu, Hau-San Wong, Jane You, Guoqiang Han, and Jun Zhang. Semi-supervised ensemble clustering based on selected constraint projection. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2394–2407, 2018.

[324] Yongxuan Lai, Songyao He, Zhijie Lin, Fan Yang, Qifeng Zhou, and Xiaofang Zhou. An adaptive robust semi-supervised clustering framework using weighted consensus of random *k* k-means ensemble. *IEEE Transactions on Knowledge and Data Engineering*, 33(5):1877–1890, 2019.

[325] Tianshu Yang, Nicolas Pasquier, and Frédéric Precioso. Semi-supervised consensus clustering based on closed patterns. *Knowledge-Based Systems*, 235:107599, 2022.

[326] Daniel Duarte Abdala and Xiaoyi Jiang. An evidence accumulation approach to constrained clustering combination. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 361–371. Springer, 2009.

[327] Yan Yang, Hongjun Wang, Chao Lin, and Jinyuan Zhang. Semi-supervised clustering ensemble based on multi-ant colonies algorithm. In *International Conference on Rough Sets and Knowledge Technology*, pages 302–309. Springer, 2012.

[328] Yan Yang, Wei Tan, Tianrui Li, and Da Ruan. Consensus clustering based on constrained self-organizing map and improved cop-kmeans ensemble in intelligent decision support systems. *Knowledge-Based Systems*, 32:101–115, 2012.

[329] Zhiwen Yu, Zongqiang Kuang, Jiming Liu, Hongsheng Chen, Jun Zhang, Jane You, Hau-San Wong, and Guoqiang Han. Adaptive ensembling of semi-supervised clustering solutions. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1577–1590, 2017.

[330] Rui Luo, Zhiwen Yu, Wenming Cao, Cheng Liu, Hau-San Wong, and CL Philip Chen. Adaptive regularized semi-supervised clustering ensemble. *IEEE Access*, 8:17926–17934, 2019.

[331] Viet-Vu Vu, Nicolas Labroche, and Bernadette Bouchon-Meunier. Leader ant clustering with constraints. In *2009 IEEE-RIVF International Conference on Computing and Communication Technologies*, pages 1–8. IEEE, 2009.

[332] Xiaohua Xu, Lin Lu, Ping He, Zhoujin Pan, and Ling Chen. Improving constrained clustering via swarm intelligence. *Neurocomputing*, 116:317–325, 2013.

[333] Julia Handl and Joshua Knowles. On semi-supervised clustering via multiobjective optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1465–1472, 2006.

[334] Yi Hong, Sam Kwong, Hanli Wang, Qingsheng Ren, and Yuchou Chang. Probabilistic and graphical model based genetic algorithm driven clustering with instance-level constraints. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 322–329. IEEE, 2008.

[335] Yi Hong, Sam Kwong, Hui Xiong, and Qingsheng Ren. Genetic-guided semi-supervised clustering algorithm with instance-level constraints. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1381–1388, 2008.

[336] Javid Ebrahimi and Mohammad Saniee Abadeh. Semi supervised clustering: a pareto approach. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 237–251. Springer, 2012.

[337] Zhenfeng He. Evolutionary k-means with pair-wise constraints. *Soft Computing*, 20(1):287–301, 2016.

[338] Rudinei Martins de Oliveira, Antonio Augusto Chaves, and Luiz Antonio Nogueira Lorena. A comparison of two hybrid methods for constrained clustering problems. *Applied Soft Computing*, 54:256–266, 2017.

[339] Thiago F Covões and Eduardo R Hruschka. Classification with multi-modal classes using evolutionary algorithms and constrained clustering. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2018.

[340] Germán González-Almagro, Julián Luengo, José-Ramón Cano, and Salvador García. Enhancing instance-level constrained clustering through differential evolution. *Applied Soft Computing*, page 107435, 2021.

[341] Tran Khanh Hiep, Nguyen Minh Duc, and Bui Quoc Trung. Local search approach for the pairwise constrained clustering problem. In *Proceedings of the Seventh Symposium on Information and Communication Technology*, pages 115–122, 2016.

[342] Zhong Zhang, Didi Kang, Chongming Gao, and Junming Shao. Semisync: Semi-supervised clustering by synchronization. In *International Conference on Database Systems for Advanced Applications*, pages 358–362. Springer, 2019.

[343] Yan Yang and Hao Wang. Multi-view clustering: A survey. *Big Data Mining and Analytics*, 1(2):83–107, 2018.

[344] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93, 2000.

[345] Hichem Frigui and Cheul Hwang. Adaptive concept learning through clustering and aggregation of relational data. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 90–101. SIAM, 2007.

[346] Hichem Frigui and Cheul Hwang. Fuzzy clustering and aggregation of relational data with instance-level constraints. *IEEE Transactions on Fuzzy Systems*, 16(6):1565–1581, 2008.

[347] Eric Eaton, Marie Desjardins, and Sara Jacob. Multi-view constrained clustering with an incomplete mapping between views. *Knowledge and information systems*, 38(1):231–257, 2014.

[348] Stephan Günnemann, Ines Färber, Matthias Rüdiger, and Thomas Seidl. Smvc: semi-supervised multi-view clustering in subspace projections. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 253–262, 2014.

[349] Daniel Khashabi, John Wieting, Jeffrey Yufei Liu, and Feng Liang. Clustering with side information: From a probabilistic model to a deterministic algorithm. *arXiv preprint arXiv:1508.06235*, 2015.

[350] Peng Zhao, Yuan Jiang, and Zhi-Hua Zhou. Multi-view matrix completion for clustering with side information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 403–415. Springer, 2017.

[351] Zhiwu Lu and Yuxin Peng. Unified constraint propagation on multi-view data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pages 640–646, 2013.

[352] Zheng Yang, Yao Hu, Haifeng Liu, Huajun Chen, and Zhaohui Wu. Matrix completion for cross-view pairwise constraint propagation. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 897–900, 2014.

[353] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. Semi-supervised graph clustering: a kernel approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 457–464, 2005.

[354] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 74(1):1–22, 2009.

[355] Bojun Yan and Carlotta Domeniconi. An adaptive kernel method for semi-supervised clustering. In *European conference on machine learning*, pages 521–532. Springer, 2006.

[356] Yi Liu, Rong Jin, and Anil K Jain. Boostcluster: boosting clustering by pairwise constraints. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 450–459, 2007.

[357] Sílvia Grasiella Moreira Almeida, Frederico Gualberto F Coelho, Frederico Gadelha Guimarães, and Antonio Pádua Braga. A general approach for adaptive kernels in semi-supervised clustering. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 508–515. Springer, 2012.

[358] Changyou Chen, Junping Zhang, Xuefang He, and Zhi-Hua Zhou. Non-parametric kernel learning with robust pairwise constraints. *International Journal of Machine Learning and Cybernetics*, 3(2):83–96, 2012.

[359] Xin Huang, Hong Cheng, Jiong Yang, Jeffery Xu Yu, Hongliang Fei, and Jun Huan. Semi-supervised clustering of graph objects: A subgraph mining approach. In *International Conference on Database Systems for Advanced Applications*, pages 197–212. Springer, 2012.

[360] Hengjin Tang and Sadaaki Miyamoto. Semi-supervised sequential kernel regression models with pairwise constraints. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 166–178. Springer, 2013.

[361] M Soleymani Baghshah, Fatemeh Afsari, S Bagheri Shouraki, and Esfandiar Eslami. Scalable semi-supervised clustering by spectral kernel learning. *Pattern Recognition Letters*, 45:161–171, 2014.

[362] Andrea Baraldi and Palma Blonda. A survey of fuzzy clustering algorithms for pattern recognition. i. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(6):778–785, 1999.

[363] Enrique H Ruspini, James C Bezdek, and James M Keller. Fuzzy clustering: A historical perspective. *IEEE Computational Intelligence Magazine*, 14(1):45–55, 2019.

[364] Yuchi Kanzawa, Yasunori Endo, and Sadaaki Miyamoto. Some pairwise constrained semi-supervised fuzzy c-means clustering algorithms. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 268–281. Springer, 2009.

[365] Cui-Fang Gao and Xiao-Jun Wu. A new semi-supervised clustering algorithm with pairwise constraints by competitive agglomeration. *Applied Soft Computing*, 11(8):5281–5291, 2011.

[366] Violaine Antoine, Benjamin Quost, M-H Masson, and Thierry Denoeux. Cecm: Constrained evidential c-means algorithm. *Computational Statistics & Data Analysis*, 56(4):894–914, 2012.

[367] Filipe M de Melo and Francisco de AT de Carvalho. Semi-supervised fuzzy c-medoids clustering algorithm with multiple prototype representation. In *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7. IEEE, 2013.

[368] Yukihiro Hamasuna and Yasunori Endo. On semi-supervised fuzzy c-means clustering for data with clusterwise tolerance by opposite criteria. *Soft Computing*, 17(1):71–81, 2013.

[369] Violaine Antoine, Benjamin Quost, M-H Masson, and Thierry Denoeux. Cevclus: evidential clustering with instance-level constraints for relational data. *Soft Computing*, 18(7):1321–1335, 2014.

[370] Longlong Li, Jonathan M Garibaldi, Dongjian He, and Meili Wang. Semi-supervised fuzzy clustering with feature discrimination. *PloS one*, 10(9):e0131160, 2015.

[371] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[372] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing gaussian mixture models with em using equivalence constraints. *Advances in neural information processing systems*, 16(8):465–472, 2004.

[373] Zhengdong Lu and Todd Leen. Semi-supervised learning with penalized probabilistic clustering. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005.

[374] Zhengdong Lu and Todd K Leen. Penalized probabilistic clustering. *Neural Computation*, 19(6):1528–1567, 2007.

[375] Zhengdong Lu. Semi-supervised clustering with pairwise constraints: A discriminative approach. In *Artificial Intelligence and Statistics*, pages 299–306. PMLR, 2007.

[376] Marek Śmieja and Magdalena Wiercioch. Constrained clustering with a complex cluster structure. *Advances in Data Analysis and Classification*, 11(3):493–518, 2017.

[377] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Semi-supervised fuzzy clustering with pairwise-constrained competitive agglomeration. In *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05.*, pages 867–872. IEEE, 2005.

[378] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. C-dbscan: Density-based clustering with constraints. In *International workshop on rough sets, fuzzy sets, data mining, and granular-soft computing*, pages 216–223. Springer, 2007.

[379] Carlos Ruiz, Ernestina Menasalvas, and Myra Spiliopoulou. C-denstream: Using domain knowledge on a data stream. In *International Conference on Discovery Science*, pages 287–301. Springer, 2009.

[380] Yukihiro Hamasuna, Yasunori Endo, and Sadaaki Miyamoto. Semi-supervised agglomerative hierarchical clustering using clusterwise tolerance based pairwise constraints. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 152–162. Springer, 2010.

[381] Yukihiro Hamasuna, Yasunori Endo, and Sadaaki Miyamoto. Semi-supervised agglomerative hierarchical clustering with ward method using clusterwise tolerance. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 103–113. Springer, 2011.

[382] Tengke Xiong, Shengrui Wang, André Mayers, and Ernest Monga. Semi-supervised parameter-free divisive hierarchical clustering of categorical data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 265–276. Springer, 2011.

[383] Sadaaki Miyamoto and Akihisa Terami. Constrained agglomerative hierarchical clustering algorithms with penalties. In *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, pages 422–427. IEEE, 2011.

[384] Sean Gilpin and Ian Davidson. Incorporating sat solvers into hierarchical clustering algorithms: an efficient and flexible approach. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1136–1144, 2011.

[385] Germán González-Almagro, Juan Luis Suarez, Julián Luengo, José-Ramón Cano, and Salvador García. Agglomerative constrained clustering through similarity and distance recalculation. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 424–436. Springer, 2020.

[386] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(3):231–240, 2011.

[387] Wen-Qi Fan, Chang-Dong Wang, and Jian-Huang Lai. Sdenpeak: semi-supervised nonlinear clustering based on density and distance. In *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 269–275. IEEE, 2016.

[388] Walid Atwa and Kan Li. Constraint-based clustering algorithm for multi-density data and arbitrary shapes. In *Industrial Conference on Data Mining*, pages 78–92. Springer, 2017.

[389] Yun Yang, Zongze Li, Wei Wang, and Dapeng Tao. An adaptive semi-supervised clustering approach via multiple density-based information. *Neurocomputing*, 257:193–205, 2017.

[390] Yazhou Ren, Xiaohui Hu, Ke Shi, Guoxian Yu, Dezhong Yao, and Zenglin Xu. Semi-supervised denpeak clustering with pairwise constraints. In *Pacific Rim International Conference on Artificial Intelligence*, pages 837–850. Springer, 2018.

[391] Shan Yan, Hongjun Wang, Tianrui Li, Jielei Chu, and Jin Guo. Semi-supervised density peaks clustering based on constraint projection. *Int. J. Comput. Intell. Syst.*, 14(1):140–147, 2021.

[392] Lei Xu, Adam Krzyzak, and Erkki Oja. Rival penalized competitive learning for clustering analysis, rbf net, and curve detection. *IEEE Transactions on Neural networks*, 4(4):636–649, 1993.

[393] Carlos Ruiz Moreno, Myra Spiliopoulou, and Ernestina Menasalvas. User constraints over data streams. *Knowl. Discov. from Data Streams*, page 117, 2006.

[394] Thiago F Covões, Eduardo R Hruschka, and Joydeep Ghosh. Competitive learning with pairwise constraints. *IEEE transactions on neural networks and learning systems*, 24(1):164–169, 2012.

[395] Inmar Givoni and Brendan Frey. Semi-supervised affinity propagation with instance-level constraints. In *Artificial intelligence and statistics*, pages 161–168. PMLR, 2009.

[396] Ian Davidson and Zijie Qi. Finding alternative clusterings using constraints. In *2008 Eighth IEEE International Conference on Data Mining*, pages 773–778. IEEE, 2008.

[397] Jan Struyf and Sašo Džeroski. Clustering trees with instance level constraints. In *European Conference on Machine Learning*, pages 359–370. Springer, 2007.

[398] Ruggero G Pensa, Jean-Francois Boulicaut, Francesca Cordero, and Maurizio Atzori. Co-clustering numerical data under user-defined constraints. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(1):38–55, 2010.

[399] Khanh-Chuong Duong, Christel Vrain, et al. Constrained clustering by constraint programming. *Artificial Intelligence*, 244:70–94, 2017.

[400] Daniele Calandriello, Gang Niu, and Masashi Sugiyama. Semi-supervised information-maximization clustering. *Neural networks*, 57:103–111, 2014.

[401] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. Constrained minimum sum of squares clustering by constraint programming. In *International Conference on Principles and Practice of Constraint Programming*, pages 557–573. Springer, 2015.

[402] Yu Xia. A global optimization method for semi-supervised clustering. *Data mining and knowledge discovery*, 18(2):214–256, 2009.

[403] Yuanhong Li, Ming Dong, and Yunqian Ma. Feature selection for clustering with constraints using jensen-shannon divergence. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[404] Jianan Zhao, Ding Xiao, Linmei Hu, and Chuan Shi. Coupled semi-supervised clustering: Exploring attribute correlations in heterogeneous information networks. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 95–109. Springer, 2019.

[405] Xiang Li, Yao Wu, Martin Ester, Ben Kao, Xin Wang, and Yudian Zheng. Semi-supervised clustering in attributed heterogeneous information networks. In *Proceedings of the 26th international conference on world wide web*, pages 1621–1629, 2017.

[406] Ian Davidson, SS Ravi, and Martin Ester. Efficient incremental constrained clustering. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 240–249, 2007.

[407] Liang Bai, Junbin Wang, Jiye Liang, and Hangyuan Du. New label propagation algorithm with pairwise constraints. *Pattern Recognition*, 106:107411, 2020.

[408] Chao Lin, Yan Yang, and Tonny Rutayisire. A parallel cop-kmeans clustering algorithm based on mapreduce framework. In *Knowledge Engineering and Management*, pages 93–102. Springer, 2011.

[409] Hong Zeng and Yiu-ming Cheung. Semi-supervised maximum margin clustering with pairwise constraints. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):926–939, 2011.

[410] Yang Hu, Jingdong Wang, Nenghai Yu, and Xian-Sheng Hua. Maximum margin clustering with pairwise constraints. In *2008 Eighth IEEE International Conference on Data Mining*, pages 253–262. IEEE, 2008.

[411] Hong Zeng, Aiguo Song, and Yiu Ming Cheung. Improving clustering with pairwise constraints: a discriminative approach. *Knowledge and information systems*, 36(2):489–515, 2013.

[412] Marek Śmieja, Oleksandr Myronov, and Jacek Tabor. Semi-supervised discriminative clustering with graph regularization. *Knowledge-Based Systems*, 151:24–36, 2018.

[413] Jean-Philippe Métivier, Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, and Samir Loudni. Constrained clustering using sat. In *International Symposium on Intelligent Data Analysis*, pages 207–218. Springer, 2012.

[414] Jeremias Berg and Matti Järvisalo. Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability. *Artificial Intelligence*, 244:110–142, 2017.

[415] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall. Boosting margin based distance functions for clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 50, 2004.

[416] Hong Chang and Dit-Yan Yeung. Locally linear metric adaptation for semi-supervised clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 20, New York, NY, USA, 2004. Association for Computing Machinery.

[417] Dit-Yan Yeung and Hong Chang. Extending the relevant component analysis algorithm for metric learning using both positive and negative equivalence constraints. *Pattern Recognition*, 39(5):1007–1010, 2006.

[418] Steven CH Hoi, Wei Liu, Michael R Lyu, and Wei-Ying Ma. Learning distance metrics with contextual constraints for image retrieval. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2072–2078. IEEE, 2006.

[419] Ron Bekkerman, Mehran Sahami, and Erik Learned-Miller. Combinatorial markov random fields. In *European Conference on Machine Learning*, pages 30–41. Springer, 2006.

[420] Dit-Yan Yeung and Hong Chang. A kernel approach for semisupervised metric learning. *IEEE Transactions on Neural Networks*, 18(1):141–149, 2007.

[421] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, 2007.

[422] Steven CH Hoi, Rong Jin, and Michael R Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *Proceedings of the 24th international conference on Machine learning*, pages 361–368, 2007.

[423] Shiming Xiang, Feiping Nie, and Changshui Zhang. Learning a mahalanobis distance metric for data clustering and classification. *Pattern recognition*, 41(12):3600–3612, 2008.

[424] Mahdieh Soleymani Baghshah and Saeed Bagheri Shouraki. Metric learning for semi-supervised clustering using pairwise constraints and the geometrical structure of data. *Intelligent Data Analysis*, 13(6):887–899, 2009.

[425] Mahdieh Soleymani Baghshah and Saeed Bagheri Shouraki. Kernel-based metric learning for semi-supervised clustering. *Neurocomputing*, 73(7-9):1352–1361, 2010.

[426] Mahdieh Soleymani Baghshah and Saeed Bagheri Shouraki. Non-linear metric learning using pairwise similarity and dissimilarity constraints and the geometrical structure of data. *Pattern Recognition*, 43(8):2982–2992, 2010.

[427] Mahdieh Soleymani Baghshah and Saeed Bagheri Shouraki. Learning low-rank kernel matrices for constrained clustering. *Neurocomputing*, 74(12-13):2201–2211, 2011.

[428] Caiming Xiong, David Johnson, Ran Xu, and Jason J Corso. Random forests for metric learning with implicit pairwise position dependence. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 958–966, 2012.

[429] Zhenyong Fu, Zhiwu Lu, Horace HS Ip, Hongtao Lu, and Yunyun Wang. Local similarity learning for pairwise constraint propagation. *Multimedia Tools and Applications*, 74(11):3739–3758, 2015.

[430] David M Johnson, Caiming Xiong, and Jason J Corso. Semi-supervised nonlinear distance metric learning via forests of max-margin cluster hierarchies. *IEEE Transactions on Knowledge and Data Engineering*, 28(4):1035–1046, 2015.

[431] Ahmad Ali Abin, Mohammad Ali Bashiri, and Hamid Beigy. Learning a metric when clustering data points in the presence of constraints. *Advances in Data Analysis and Classification*, 14(1):29–56, 2020.

[432] Dan Klein, Sepandar D Kamvar, and Christopher D Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. Technical report, Stanford, 2002.

[433] Su Yan, Hai Wang, Dongwon Lee, and C Lee Giles. Pairwise constrained clustering for sparse and high dimensional feature spaces. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 620–627. Springer, 2009.

[434] Zhiwu Lu and Horace HS Ip. Constrained spectral clustering via exhaustive and efficient constraint propagation. In *European Conference on Computer Vision*, pages 1–14. Springer, 2010.

[435] Zhenyong Fu, Zhiwu Lu, Horace Ip, Yuxin Peng, and Hongtao Lu. Symmetric graph regularized constraint propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 350–355, 2011.

[436] Zhenyong Fu, Horace HS Ip, Hongtao Lu, and Zhiwu Lu. Multi-modal constraint propagation for heterogeneous image clustering. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 143–152, 2011.

[437] Ping He, Xiaohua Xu, and Ling Chen. Constrained clustering with local constraint propagation. In *European Conference on Computer Vision*, pages 223–232. Springer, 2012.

[438] Shmuel Asafi and Daniel Cohen-Or. Constraints as features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1634–1641, 2013.

[439] Ping He, Xiaohua Xu, Lei Zhang, Wei Zhang, Kanwen Li, and Heng Qian. Constrained community clustering. In *International Conference on Intelligent Computing*, pages 797–802. Springer, 2014.

[440] Xiaohua Xu and Ping He. Improving clustering with constrained communities. *Neurocomputing*, 188:239–252, 2016.

[441] Yuheng Jia, Hui Liu, Junhui Hou, and Sam Kwong. Pairwise constraint propagation with dual adversarial manifold regularization. *IEEE Transactions on Neural Networks and Learning Systems*, 31(12):5575–5587, 2020.

[442] Yaoyi Li and Hongtao Lu. Multi-modal constraint propagation via compatible conditional distribution reconstruction. *Neurocomputing*, 426:185–194, 2021.

[443] Jinfeng Yi, Lijun Zhang, Rong Jin, Qi Qian, and Anil Jain. Semi-supervised clustering by input pattern assisted pairwise similarity matrix completion. In *International conference on machine learning*, pages 1400–1408. PMLR, 2013.

[444] Zhenyong Fu. Pairwise constraint propagation via low-rank matrix recovery. *Computational Visual Media*, 1(3):211–220, 2015.

[445] Zhenguo Li, Jianzhuang Liu, and Xiaoou Tang. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *Proceedings of the 25th international conference on Machine learning*, pages 576–583, 2008.

[446] Hongjun Wang, Ruihua Nie, Xingnian Liu, and Tianrui Li. Constraint projections for semi-supervised affinity propagation. *Knowledge-Based Systems*, 36:315–321, 2012.

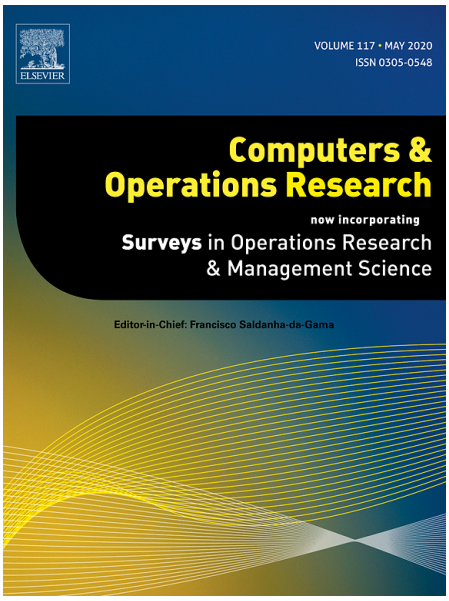
[447] Shan Gao, Chen Zu, and Daoqiang Zhang. Learning mid-perpendicular hyperplane similarity from cannot-link constraints. *Neurocomputing*, 113:195–203, 2013.

[448] Yuheng Jia, Sam Kwong, Junhui Hou, and Wenhui Wu. Convex constrained clustering with graph-laplacian pca. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2018.

[449] Yuheng Jia, Junhui Hou, and Sam Kwong. Constrained clustering with dissimilarity propagation-guided graph-laplacian pca. *IEEE Transactions on Neural Networks and Learning Systems*, 32(9):3985–3997, 2020.

[450] Jayaram Raghuram, David J Miller, and George Kesidis. Instance-level constraint-based semisupervised learning with imposed space-partitioning. *IEEE transactions on neural networks and learning systems*, 25(8):1520–1537, 2014.

2 DILS: Constrained clustering through dual iterative local search



- **Journal:** Computers & Operations Research (COR)
- **JCR Impact Factor:** 4.008
- **Rank:** 39/111
- **Quartile:** Q2
- **Category:** Computer Science, Interdisciplinary Applications
- **Status:** Published

Ref.: González-Almagro, G., Luengo, J., Cano, J. R., & García, S. (2020). DILS: constrained clustering through dual iterative local search. Computers & Operations Research, 121, 104979. DOI: https://doi.org/10.1016/j.cor.2020.104979.

DILS: CONSTRAINED CLUSTERING THROUGH DUAL ITERATIVE LOCAL SEARCH

Germán González Almagro [*,a,b] **Julián Luengo** [a,b] **José-Ramón Cano** [c,b]

Salvador García [a,b]

[a] *Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada, Granada, Spain*
[b] *Andalusian Institute of Data Science and Computational Intelligence (DaSCI)*
[c] *Department of Computer Science, University of Jaén, Jaén, Spain*

ABSTRACT

Clustering has always been a powerful tool in knowledge discovery. Traditionally unsupervised, it has recently received renewed attention, as it has shown to produce better results when provided with new types of information; thus leading to a new kind of semi-supervised learning: constrained clustering. This technique is a generalization of traditional clustering that considers additional information encoded by constraints. Constraints can be given in the form of instance-level must-link and cannot-link constraints, which is the focus of this paper. We propose a new metaheuristic algorithm, the Dual Iterative Local Search, and prove its ability to produce quality results for the constrained clustering problem. We compare the results obtained by this proposal to those obtained by the state-of-the-art algorithms on 25 datasets with incremental levels of constraint-based information, supporting our conclusions with the aid of Bayesian statistical tests.

Keywords Constrained Clustering · Instance-level · Must-link · Cannot-link · Dual Iterative Local Search.

* Corresponding Author (germangalmagro@ugr.es)
Email addresses: julianlm@decsai.ugr.es (Julián Luengo), jrcano@ujaen.es (José Ramón Cano), salvagl@decsai.ugr.es (Salvador García)

1 Introduction

Clustering is one of the most well-known and extensively studied data analysis problems. It constitutes a key research area in the field of unsupervised learning, where there is no supervision on how the information should be handled. Partitional clustering can be defined as the task of grouping the instances of a dataset into k clusters, so that new information can be extracted from them. A dataset X is composed of n instances, and each instance is described by u features. More formally, $X = \{x_1, \cdots, x_n\}$, with the ith instance noted as $x_i = (x_{[i,1]}, \cdots, x_{[i,u]})$. A typical clustering algorithm assigns a class label l_i to each instance $x_i \in X$. As a result, the set of labels $L = \{l_1, \cdots, l_n\}$ is obtained, with $l_i \in \{1, \cdots, k\}$, that effectively splits X into k non-overlapping clusters c_i to form a partition called C. The criterion used to assign an instance to a given cluster is the similarity to the rest of elements in that cluster, and the dissimilarity to the rest of instances of the dataset, and can be obtained with some kind of distance measurement [1].

Semi-supervised learning (SSL) is a machine learning paradigm that arises from adding incomplete information to unsupervised learning [2]. Following this paradigm, background information to the clustering process can be incorporated, resulting in constrained clustering, which is the main subject of the study presented in this paper [3]. The objective of constrained clustering is to find a partition of the dataset that meets the proper characteristics of a clustering method result, in addition to satisfying a certain constraint set. It has been successfully applied in many knowledge fields, among which the following are worth mentioning: advanced robotics applications [4], hyperspectral image classification [5], applied marketing [6], obstructive sleep apnea analysis [7], terrorist sub-communities detection [8], vocabulary maintenance policy for case-based reasoning systems [9], electoral district designing, [10], and lane finding in GPS data [11] among others.

Constraints can be understood in different ways, resulting in three main types of constrained clustering: cluster-level [12], instance-level [13] and feature-level constrained clustering [14]. Moreover, hybrid approaches which try to integrate different types of constraints have also been proposed [15]. Particularly, there are two main types of instance-level constraints in the literature: pairwise constraints and distance-based constraints. Specifically, pairwise constraints tell us if two specific instances of a dataset must be placed in the same or in different clusters, resulting in Must-link (ML) and Cannot-link (CL) constraints respectively. This paper focuses on these types of constraints (ML and CL), which will be discussed later in Section 2.1.

With regard to the degree to which the constraints have to be satisfied, a distinction between the concepts of hard [11] and soft [16] constraints can be made. Hard constraints must necessarily be satisfied in the output partition of any algorithm that makes use of them, whereas soft constraints are taken as a strong guide for the algorithm that uses them but can be partially satisfied in the output partition [6]. For the purpose of this paper, soft constraints will be employed, given their practical applicability to real-world problems.

Finding the optimal partition in a dataset, with respect to any kind of reasonable criteria, is known as an **NP**-hard problem. Therefore, the incorporation of constraints may modify

the complexity of the clustering problem, depending on the type of constraints used. As it will be studied in more depth in Section 2.2, the use of ML and CL constraints makes the constrained clustering problem **NP**-complete [17].

The constrained clustering problem can be formulated in terms of optimization so that various optimization techniques to solve it can be applied. As mentioned earlier, its resolution poses a tough challenge, and metaheuristics are presented as a promising option to find quality approximate solutions. Metaheuristic algorithms are designed to explore the solution space of a problem with the guidance of a fitness (heuristic) function [18]. In this type of approach, the key is to find a good exploration-exploitation trade-off. For this reason, the Iterated Local Search (ILS) metaheuristic algorithm, derived from Local Search (LS), introduces periodic perturbations in the LS exploration of the solution space process to escape local optima [19]. ILS—or variants of it—has been applied to a wide variety of problems, including the traveling salesman [20], the quadratic multiple knapsack [21] or the vehicle routing problem [22, 23]. ILS is also commonly used in the design of hybrid methods with good exploration-exploitation trade-off, such as its combination with expediting mechanisms [24] or with the success-history based differential evolution algorithm [25].

Metaheuristics methods have been successfully applied to the clustering problem [26, 27, 28]; although little work has been done on investigating their suitability for the constrained clustering problem, particularly in highly constrained environments. Encouraged by this success, we propose a new ILS variant, which we call Dual Iterative Local Search (DILS) and which constitutes the main contribution of this paper. DILS seeks to explore the solution space by combining the exploitation capability of ILS and classic diversity-introducing techniques used in the metaheuristics field such as recombination and mutation operators. DILS is also able to adapt to the problem at hand by implementing a restarting mechanism to avoid local optima. To carry out these tasks, DILS optimizes two individuals at the same time, which allows it to recombine and compare them in order to to manage the exploration of the solution space. We have built the application of DILS to constrained clustering—which is referred to as DILS$_{CC}$—with the aid of an integer-based problem representation and a penalty-style fitness function. Additionally, it has been proven that it constitutes a competitive approach to obtain quality results for the constrained clustering problem. Particularly, it has been shown how the DILS approach is able to scale the quality of the results as the number of constraints increases, making it suitable for highly constrained problems. This is due to the exploitation capability DILS exhibits when exploring the solution space.

In terms of the organization of this paper, Section 2 reviews the existing knowledge concerning constrained clustering and the state-of-the-art. In Sections 3 and 4, the DILS algorithm and its formulation for constrained clustering DILS$_{CC}$ will be reviewed. Sections 5 to 7 present the experimental setup, the results and their analysis, respectively. Finally, in Section 9 the conclusions are discussed.

2 Background

In this section, the background knowledge concerning constrained clustering (Section 2.1) is presented, its computational complexity (Section 2.2) and a brief description of some of the state-of-the-art methods for constrained clustering (Section 2.3).

2.1 Constrained Clustering

In most clustering applications, it is common to have some kind of information about the dataset which will be analyzed. In pairwise instance-level constrained clustering, this information is given in the form of pairs of instances. A constraint states whether the instances which it refers to must, or must not, be assigned to the same cluster. It is possible to obtain a better result by using this type of information than by using completely unsupervised clustering algorithms. The two types of constraints mentioned can now be formalized:

- Must-link constraints $C_{=}(x_i, x_j)$: instances x_i and x_j from X must be placed in the same cluster.
- Cannot-link constraints $C_{\neq}(x_i, x_j)$: instances x_i and x_j from X cannot be assigned to the same cluster.

The goal of constrained clustering is to find a partition (or clustering) of k clusters $C = \{c_1, \cdots, c_k\}$ of the dataset X that ideally satisfies all constraints in the constraint set. As in the original clustering problem, the sum of instances in each cluster c_i is equal to the number of instances in X, which has been defined as $n = |X| = \sum_{i=1}^{k} |c_i|$.

Knowing how a constraint is defined, ML constraints are an example of an equivalence relation; therefore, ML constraints are reflexive, transitive and symmetrical. In this manner, given constraints $C_{=}(x_a, x_b)$ and $C_{=}(x_b, x_c)$, then $C_{=}(x_a, x_c)$ is verified. In addition, if $x_a \in c_i$ and $x_b \in c_j$ are related by $C_{=}(x_a, x_b)$, then $C_{=}(x_c, x_d)$ is verified for any $x_c \in c_i$ and $x_d \in c_j$ [13].

It can also be proven that CL constraints do not constitute an equivalence relation. However, analogously, given $x_a \in c_i$ and $x_b \in c_j$, and the constraint $C_{\neq}(x_a, x_b)$, then it is also true that $C_{\neq}(x_c, x_d)$ for any $x_c \in c_i$ and $x_d \in c_j$ [13].

2.2 The Feasibility Problem

Given that constrained clustering adds a new element to the clustering problem, it should be considered how this element affects the complexity of the problem. The feasibility problem for instance-level constrained clustering was defined in [17] as can be seen in Definition 1.

Definition 1 ***Feasibility Problem****: given a dataset X, a constraint set CS, and the bounds on the number of clusters $k_l \leq k \leq k_u$, is there a partition C of X with k clusters such that all constraints in CS are satisfied? [17]*

In [17] it is proven that, when $k_l = 1$ and $k_u \geq 3$, the feasibility problem for constrained clustering is **NP**-complete, by reducing it from the Graph K-Colorability problem (it is also proven that it is not harder, so both have the same complexity). Table 1 shows the complexity of the feasibility for different types of constraints.

Constraints	Complexity
Must-Link	**P**
Cannot-Link	**NP**-complete
ML and CL	**NP**-complete

Table 1: Feasibility problem complexity [17].

These complexity results show that the feasibility problem with CL constraints is intractable, and hence constrained clustering is intractable too. For more details on the complexity of constrained clustering see [17].

Intractable problems are hard to solve with deterministic and exact methods. That is the reason why metaheuristic algorithms constitute useful approaches to finding quality solutions to the constrained clustering problem.

2.3 Constrained Clustering State-of-the-art Methods

The first adaptation of a classic clustering method for constrained clustering was proposed in [11]. It involved modifying the widely studied K-means algorithm to take into account instance-level constraints: the already-known ML and CL. This method, named COP-kmeans, introduces a modification to the assignment rule of instances to clusters of the K-means algorithm so that an instance can be assigned to a cluster only if the assignment does not violate any constraints.

Within the fuzzy clustering family of methods, Constrained Evidential c-means (CECM), a variant of the Evidential c-means (ECM [29]) algorithm, is proposed in [30]. The particularity of this algorithm is that the membership of instances to a cluster is defined by a probabilistic belief function. This method redefines constraints from the point of view of belief functions and includes them in the cost function.

A modification of the Constrained Vector Quantization Error algorithm (CVQE [17]) is proposed in [31]. The CVQE algorithm proved to produce high quality results, at the cost of a very high computational complexity. Linear CVQE (LCVQE) introduces a modification of the cost function of CVQE to make it more intuitive and less computationally complex. The experimentation resulted in a dramatic improvement of clustering quality over both noisy and clean constraint sets.

Two Views Clustering (TVClust) and Relation Dirichlet Process - Means (RDPM) were proposed in [32]. TVClust is able to incorporate the constraints into the clustering problem by making a soft interpretation of them. The authors model the dataset and constraints in different ways, perform clustering methods on them and try to find a consensus between both

interpretations. Using this model as a basis, the authors derive the deterministic algorithm RDP-means. This method can be viewed as an extension of K-means that includes side information (constraints) and is characterized by the fact that the number of clusters (k) does not need to be specified.

An adaptation of the Biased Random-Key Genetic Algorithm (BRKGA) [33]—a variant of the Random-Key genetic algorithm [34]—for the constrained clustering problem, the BRKGA+LS method, is proposed in [35]. It uses a genetic algorithm, combined with a LS procedure and guided by a penalty-style fitness function, to obtain a partition of the input dataset. It constitutes one of the few approaches to constrained clustering from the point of view of evolutionary computing. An adaptive version of the BRKGA+LS algorithm (A-BRKGA) can also be found in the literature [36]. Even if it has not yet been applied to constrained clustering, the reader may consider this method for future comparisons.

3 The Dual Iterative Local Search Method

The Dual Iterative Local Search (DILS) is a new variant of the classic ILS method [19]. Its goal is to perform a search in the solution space to find the solution with the best fitness value (given by fitness function f) by introducing diversity in an adaptive and guided way. While ILS works with a single individual, DILS keeps two of them $\{m_b, m_w\}$ in memory at all times, which allows it to guide diversity-inducing methods and to avoid local optima. The individual m_b provides the best fitness value, whereas m_w provides the worst fitness value at the end of each stage of the optimization process. These stages involve generating, evaluating and optimizing new individuals, which will be referred to as "generations". Successive generations will eventually lead to finding better individuals (in terms of f).

Recombination and mutation. DILS builds a new individual in each generation G of the optimization process by applying a recombination operator to m_b and m_w. After that, it applies a strong mutation operator to the newly generated individual. This is the individual DILS applies the LS procedure in order to improve its fitness value, resulting in the trial individual m_t. The trial individual m_t replaces the worst of its predecessors, m_w, if it achieves a better fitness value; this operation represents the acceptance criterion for DILS. Equation 1 displays the expression for this criterion in the case of a minimization problem.

$$m_{w,G} = \begin{cases} m_t & \text{if } f(m_t) < f(m_w) \\ m_{w,G} & \text{otherwise} \end{cases} . \tag{1}$$

Reinitialization method. The process described so far will most likely fall into a local optimum, effectively stopping the exploration of the solution space. To avoid it, DILS implements a reinitialization method for m_w based on the differences between the two individuals it keeps in memory ($\{m_b, m_w\}$). Equation 2 shows the reinitialization criterion for a minimization problem. RandInit() is a function that returns a randomly initialized individual and $\xi \in [-1, 1]$ is the parameter that controls the tolerance of the reinitialization method.

$$m_{w,G+1} = \begin{cases} \text{RandInit()} & \text{if } f(m_b) - f(m_w) > f(m_b) * \xi \\ m_{w,G} & \text{if } f(m_b) - f(m_w) \leq f(m_b) * \xi \ \wedge \ f(m_b) < f(m_w) \\ m_{b,G} & \text{if } f(m_b) - f(m_w) \leq f(m_b) * \xi \ \wedge \ f(m_b) > f(m_w) \end{cases} . \quad (2)$$

Maintaining consistency. In order to maintain consistency between generations, we also need to reassign m_b to the true best individual once the reinitialization criterion has been tested, as shown in Equation 3. As a result, the best individual is always preserved and therefore always takes part in the process of generating new individuals (via the recombination operator) as the best predecessor.

$$m_{b,G+1} = \begin{cases} m_{b,G} & \text{if } f(m_b) < f(m_w) \\ m_{w,G} & \text{if } f(m_b) > f(m_w) \end{cases} . \quad (3)$$

Figure 1 summarizes the DILS optimization process. Dashed arrows indicate stages of the algorithm where m_b and m_w are reassigned to the true best and worst individual, respectively. Consequently, at the point where the reinitialization criterion is tested, it is possible for $f(m_w)$ to yield a smaller value than $f(m_b)$, so that $f(m_b) - f(m_w)$ can be either positive or negative.

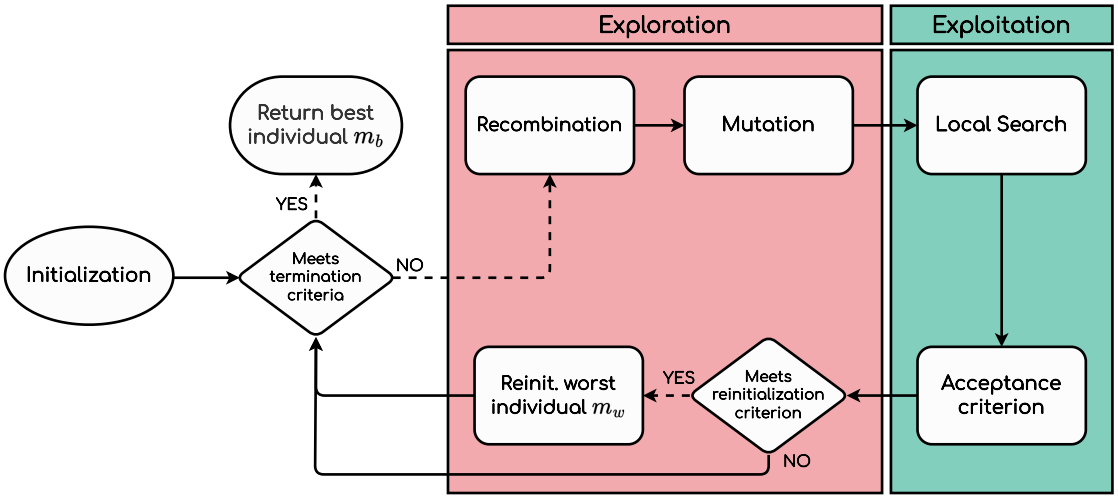


Figure 1: Diagram summarizing the DILS optimization process.

On the influence of ξ. Let us consider a minimization problem again. On the one hand, if $\xi > 0$, then the true worst individual is only reinitialized when m_t replaces m_w and is also better than m_b by a margin. On the other hand, if $\xi < 0$, we allow DILS to reinitialize the true worst individual even when m_t is not better than m_b. It becomes clear that when $\xi = 0$ the worst individual is restarted if $f(m_b) = f(m_w)$. Note that ξ could take any real value, but it is not realistic to set it outside of the range $[-1, 1]$, and it is recommended $[-0.5, 0.5]$ preserving a good exploration-exploitation trade-off. Otherwise, ξ would bias the search

towards a random-restart LS in the case of $\xi < -0.5$, or towards a simple-path optimization process applied to two individuals in the case of $\xi > 0.5$. Briefly stated, the closer ξ is to 1, the more restrictive the reinitialization criterion is—in the minimization case. It is also worth noting that in early stages of the optimization process, the reinitialization criterion tends to be met more often than in later stages. The reason for this is that the fitness value of the best individual $f(m_b)$ never worsens ($f(m_{b,G}) \geq f(m_{b,G+1})$), so it becomes increasingly harder for m_w to achieve similar or better scores. Algorithm 1 summarizes the overall DILS optimization process.

Algorithm 1: Dual Iterative Local Search (DILS)

Input: Probability for recombination operator p_r, segment size for mutation operator p_s, reinitialization method tolerance ξ.

```
// Initialization phase
[1]  G ← 0
[2]  m_b ← RandInit(); m_w ← RandInit()
     // Main loop
[3]  while Termination criteria are not met do
         // Find best and worst individual
[4]      m_b ← Best({m_b, m_w}); m_w ← Worst({m_b, m_w})
         // Generate new individual
[5]      m_t ← Recombination(m_b, m_w)
[6]      m_t ← Mutation(m_t)
         // Improve new individual
[7]      m_t ← LocalSearch(m_t)
         // Apply replacement operator
[8]      if f(m_t) < f(m_w) then
[9]          m_w ← m_t
[10]     end
         // Check restart criterion
[11]     if f(m_b) − f(m_w) > f(m_b) × ξ then
[12]         m_b ← Best({m_b, m_w})
[13]         m_w ← RandInit()
[14]     end
[15]     G ← G + 1
[16] end
[17] m_b ← Best({m_b, m_w})
[18] return m_b
```

For the recombination and mutation operators that appear in Algorithm 1, we use the uniform recombination and the segment mutation operator, respectively. The uniform recombination consists in selecting features from the best individual m_b based on a given probability p_r to introduce them in the resulting individual, so the rest of the features are taken from m_w [37]. The segment mutation operator consists in replacing a fixed-size segment from the features of the individual with randomly generated features. The size of the segment p_s must be fixed at the beginning of the optimization process. The starting point of the segment to replace is chosen randomly. Although other (problem-specific) recombination and mutation operators could be used in DILS, we highly recommend the operators described above. The uniform recombination operator provides a non-biased way to include the features of the best solutions found in the search process so far, while the segment mutation operator is able to produce diversity to widen the search space.

It should be noted that the optimization process described so far is applicable to any minimization problem, although it is clear that there is a direct adaptation of DILS for maximization problems. In addition, the LS procedure used to obtain the trial individual m_t must be specified for each problem, as well as numerical representation details and termination criteria.

On the complexity of DILS. Regarding the algorithmic complexity of the proposed DILS method, the mutation and recombination operators are the only complex algorithmic operations that have been explicitly defined. Both of them can be performed in linear time $\mathcal{O}(n)$, with n being the length of the vector representing the solution to the problem. However, the fitness function f and the LS procedure need to be defined for each problem and will likely represent the algorithmic bottleneck for the optimization process, as most hard problems (suitable to be addressed with DILS) will make use of complex evaluation and local optimization methods.

4 DILS application scheme for constrained clustering

In this section, the application scheme of DILS for constrained clustering is generated. This new approach will be referred to as DILS$_{CC}$. An integer-based representation will be used for all the individuals with which DILS$_{CC}$ works. In doing so, each individual m is defined as a vector of n integers, with n being the number of instances of the dataset. Each integer $m_i \in [0, k)$ in m represents the label of the cluster that instance x_i is assigned to. Therefore, each individual m is a solution to the clustering problem. a label-based representation has been selected because of its straightforward application to DILS operators such as recombination and mutation operators. Please notice that the neighborhood-generation operator from the LS procedure benefits from a label-based representation by simplifying the tracking of the already generated neighbors. Additionally, in order not to bias the exploration of the solution space, a random initialization will be employed to set m_b and m_w; this procedure will be applied in the reinitialization of m_w as well.

When it comes to the diversity-introducing operators, the recommended uniform recombination and segment mutation operators are used. Please note that no repairing procedure

is applied to the newly generated solutions (using the mentioned operators). As a consequence, the resulting solutions will most likely violate an unspecified number of constraint. Experimental results show that there is no need for that kind of repairing procedure, as it would highly bias the exploration of the solution space.

Regarding the fitness function f, for each individual m its fitness value f_m can be calculated as seen in Equation 4.

$$f_m = z_m * \overbrace{(\text{infeasibility}_m + 1)}^{penalty}, \tag{4}$$

where infeasibility_m is the number of unsatisfied constraints and z_m is the within-cluster-sum-of-squares that can be computed as shown in Equation (5). As infeasibility is used as a scale factor, this fitness function allows DILS to clearly identify solutions which satisfy different numbers of constraints. Let us remember that, even though a soft interpretation of the constraints is made, they must be used as a strong guide for the optimization process. The fitness function is designed to favor the exploration of the solution space from the point of view of the constraint set, so that in Figure 1 the exploration module optimizes the penalty term in Equation (5) while the exploitation module optimizes z_m. With this in mind, both the exploration and the exploitation module share the overall fitness optimization workload without being completely independent from each other.

$$z_m = \sum_{c_i \in C_m} \left[\frac{\sum_{x_a, x_b \in c_i} d^2(x_a, x_b)}{|c_i|} \right]. \tag{5}$$

Furthermore, it is necessary to specify an LS procedure for DILS to improve the mutant individual to obtain the trial vector m_t. Algorithm 2 shows the LS used in DILS$_{CC}$. In each LS iteration, a random index $i \in [1, n]$ (instance x_i) is chosen to explore its neighborhood, which is generated by changing its label l_i (moving it from one cluster to another) in a random fashion, in order not to bias the search. When a more suitable neighbor is found,($f(m') < f(m)$) it replaces the current solution m and neighborhood generation stops. If no improvement is found, a new random index that has not already been explored is selected. If all indices have been explored, the explored set is rearranged to empty. Note that, in order not to bias the search, indices from vector m and labels from the set of labels are randomly chosen to generate the neighborhood of m. The maximum number of neighbors p_m that can be generated in each call is stablished as well; which is done to ensure a better exploration-exploitation trade-off.

On the complexity of DILS$_{CC}$. In terms of the algorithmic complexity of the DILS optimization scheme for constrained clustering, it is necessary to start focusing on the defined specific LS procedure . The maximum number of iterations is given by the p_m parameter, and in each iteration the entire set of labels could be analyzed ($p_m * k$). Additionally, we need to study the complexity of the fitness function f, which is also specific to the constrained clustering problem. Constraints can be given in the form of lists or matrices. Computing the

Algorithm 2: Local Search

Input: Dataset X, constraint sets $C_=$ and $C_{\neq}$, individual (solution) m, number of clusters k, maximum number of neighbors that can be generated p_m.

[1] $explored \leftarrow \emptyset$
[2] **while** $improvement$ **and** $generated < p_m$ **do**
[3] $improvement \leftarrow$ `false`
`// Select random index (object) from` m
[4] $i \leftarrow$ `Rand`$(\{1, \cdots, n\} - explored)$
[5] $explored \leftarrow explored \cup \{i\}$
`// Random shuffle labels set`
[6] $RSL \leftarrow$ `RandomShuffle`$(\{1, \cdots, k\})$
[7] **for** $l \in RSL$ ***and while not*** $improvement$ **do**
[8] $m' \leftarrow m$
`/* Move object` i `from` m `to the cluster associated with label` l `*/`
[9] $m'_i \leftarrow l$
[10] **if** $f(m') < f(m)$ **then**
[11] $m \leftarrow m'$
[12] $improvement \leftarrow$ `true`
[13] **end**
[14] $generated \leftarrow generated + 1$
[15] **end**
[16] **if** $|explored| == n$ **then**
[17] $explored \leftarrow \emptyset$
[18] **end**
[19] **end**
[20] **return** m

infeasibility of a newly generated partition of a given dataset requires $n^2/2 - 1$ comparisons when the constraints are given in matrix form (as the matrix must be symmetrical), and $|C_=| + |C_{\neq}|$ when they are given in the form of a list. Please note that, unless the complete graph of constraints is available, $|C_=| + |C_{\neq}|$ will always be smaller than $n^2/2 - 1$. Consequently, we selected the list representation of constraint for the implementation of DILS$_{CC}$. The real bottleneck of DILS$_{CC}$ is found in the computation of the within-cluster-sum-of-squares (z_m). This can be achieved by firstly computing the centroid of each cluster, which involves iterating over the dataset. Afterwars, computing the distance of each instance the centroid of the cluster it has been assigned to, which again involves iterating over the entire dataset. In doing so, the computation of the within-cluster-sum-of-squares can be carried out in $2 * n * u$ operations. Bearing this in mind, the algorithmic complexity of DILS$_{CC}$ can be extracted, as in Expression 6.

$$\overbrace{\mathcal{O}(p_m \times k \times (\overbrace{\underbrace{|C_=| + |C_{\neq}|}_{\text{Infeasibility}} \times \underbrace{n \times u}_{z_m}}^{\text{Fitness Function } f}))}^{\text{Local Search}} \tag{6}$$

5 Experimental Setup

As for our experiments, the results obtained by DILS$_{CC}$ and state-of-the-art methods over 25 datasets and three constraint sets for each one of them will be compared. Most of these datasets can be found at the Keel-dataset repository[1] [38], although some of them have been obtained via `scikit-learn` python package[2] [39]. We also include three artificial datasets in our analysis, namely: *Circles*, *Moons* and *Spiral*, which can be found at GitHub[3]. Table 2 displays a summary of every dataset.

Classification datasets are commonly used in the literature to test constrained clustering algorithms; the reason behind this being that they enable us to generate constraints with respect to the true labels (see Section 5.1). They also facilitate an easy evaluation of the quality of the algorithm by means of measures like the Adjusted Rand Index (see Section 5.2).

5.1 Constraint Generation

Since we have the true labels associated with each dataset, the method proposed in [11] to generate artificial constraint sets will be applied. This method consists of randomly selecting two instances of a dataset, then comparing its labels, and finally setting an ML or CL constraint depending on whether the labels are the same or different.

For each dataset, three different sets of constraints—CS_{10}, CS_{15} and CS_{20}— will be generated. These will be associated with three small percentages of the size of the dataset: 10%, 15% and 20%. With n_f being the fraction of the size of the dataset associated with each of

[1] https://sci2s.ugr.es/keel/category.php?cat=clas
[2] https://scikit-learn.org/stable/datasets/index.html
[3] https://github.com/GermangUgr/DILS_CC

Name	No. Instances	No. Classes	No. Features
Appendicitis	106	2	7
Breast Cancer	569	2	30
Bupa	345	2	6
Circles	300	2	2
Ecoli	336	8	7
Glass	214	6	9
Haberman	306	2	3
Hayesroth	160	3	4
Heart	270	2	13
Ionosphere	351	2	33
Iris	150	3	4
Led7Digit	500	10	7
Monk2	432	2	6
Moons	300	2	2
Movement Libras	360	15	90
Newthyroid	215	3	5
Saheart	462	2	9
Sonar	208	2	60
Soybean	47	4	35
Spectfheart	267	2	44
Spiral	300	2	2
Tae	151	3	5
Vehicle	846	4	18
Wine	178	3	13
Zoo	101	7	16

Table 2: Summary of datasets used for the experiments.

these percentages, the formula $\frac{n_f(n_f-1)}{2}$ tells us how many artificial constraints will be created for each constraint set; this number is equivalent to how many edges a complete graph with n_f vertices would have.

The random allocation of constraints has a potential advantage over simply using the constraints contained in an n_f-vertex complete graph: there is a lower probability of biasing the constraint set towards having classes with poor representation. Table 3 shows the number of constraints of each type obtained for each dataset. All constraint sets used in our experiments are available here [4].

Note that the greater the number of classes present in the dataset, the fewer ML constraints obtained with the method proposed in [11]. The reason for this being that the probability of

[4] https://drive.google.com/drive/u/1/folders/1sjnPYitey8q9zrPKa_YpFS7iNKTT15QH

Dataset	CS_{10}		CS_{15}		CS_{20}	
	ML	CL	ML	CL	ML	CL
Appendicitis	39	16	71	49	164	67
Breast Cancer	876	720	1965	1690	3487	2954
Bupa	323	272	699	627	1201	1145
Circles	208	227	502	488	853	917
Ecoli	163	398	357	918	609	1669
Glass	52	179	139	389	259	644
Haberman	304	161	634	401	1135	756
Hayesroth	39	81	102	174	177	319
Heart	178	173	396	424	744	687
Ionosphere	330	300	732	646	1299	1186
Iris	26	79	82	171	136	299
Led7Digit	126	1099	267	2508	460	4490
Monk2	473	473	979	1101	1917	1824
Moons	200	235	494	496	900	870
Movement Libras	27	603	112	1319	158	2398
Newthyroid	108	123	270	258	449	454
Saheart	595	486	1292	1123	2330	1948
Sonar	100	110	245	251	436	425
Soybean	4	6	6	22	12	33
Spectfheart	233	118	543	277	965	466
Spiral	224	211	487	503	918	852
Tae	40	80	82	171	151	314
Vehicle	874	2696	1955	6046	3589	10776
Wine	49	104	121	230	217	413
Zoo	21	34	29	91	41	169

Table 3: Number of constraints used in experiments.

randomly choosing two individuals from the same class decreases as the number of classes included in the dataset increases.

5.2 Evaluation Method

Taking in to account that the true labels associated with each of the datasets are within our scope, they can be used to evaluate the results provided by each method. The Adjusted Rand Index (ARI) will be used to measure the accuracy of the predictions resulting from each tested method [40]. The basic Rand Index computes the degree of agreement between two partitions C_1 and C_2 of a given dataset X. C_1 and C_2 are viewed as collections of $n(n-1)/2$ pairwise decisions [41].

For each pair of instances x_i and x_j in X, a partition assigns them to the same cluster or to different clusters. We take a as the number of pairings where x_i is in the same cluster as x_j in both C_1 and C_2, and b as the opposite event (x_i and x_j are in different clusters in C_1 and C_2). Following this, the degree of similarity between C_1 and C_2 is calculated as in Equation (7).

$$\text{Rand}(C_1, C_2) = \frac{a+b}{n(n-1)/2} \tag{7}$$

The ARI is a corrected-for-chance version of the Rand Index. This correction uses the expected similarity of all comparisons between clusterings specified by a random model to set up a baseline. The ARI is computed as seen in Equation (8),

$$\text{ARI}(C_1, C_2) = \frac{\text{Rand}(C_1, C_2) - \text{Expected Index}}{\text{Maximum Index} - \text{Expected Index}}, \tag{8}$$

where Maximum Index is expected to be 1 and Expected Index is the already mentioned expected degree of similarity with a random model. It is particularly noticeable that $\text{ARI}(C_1, C_2) \in [-1, 1]$, such that an ARI value close to 1 means a high degree of agreement between C_1 and C_2, a positive value close to 0 means no agreement and a value smaller that 0 means that the $\text{Rand}(C_1, C_2)$ is less than expected when comparing with random partitions. To summarize, the higher the ARI, the greater the degree of similarity between C_1 and C_2. For more details on ARI see [40].

Our objective is to quantify the quality of the solutions obtained as a result of the methods presented in this paper. To accomplish this task, we set one of the two partitions given to compute ARI as the ground truth labels.

5.3 Validation of results

In order to validate the results which will be presented in Section 6, Bayesian statistical tests will be used instead of the classic Null Hypothesis Statistical Tests (NHST). In [42] we can find an in-depth analysis of the disadvantages of NHST, and a new model is proposed for carrying out the comparisons in which researchers are interested. *"In a nutshell: NHST do not answer the question we ask"*. To shed some light on this matter, the disadvantages of the NHST that the authors highlight in [42] are based on the trap of black-and-white thinking, that is: to reject, or not to reject?

At the outset, NHST do not provide us with the probabilities associated with the analyzed hypotheses, and therefore it is not possible to answer the question "what is the probability that two methods are different?" Another pitfall of NHST is that, with a sufficiently large number of observations, it is possible to reject almost any hypothesis. This is explained as the p-value does not allow us to separate between the effective size and the sample size, which is established by the researcher.

Likewise, NHST do not provide information about the magnitude of the effects and the uncertainty of its estimate. As a consequence, NHST may reject hypotheses despite tiny effects, or even if there is significant uncertainty in the magnitude of the effects.

Furthermore, and this is a situation that all researchers have faced, NHST do not provide any information about the null hypothesis! In other words: What can we conclude when NHST do not reject the null hypothesis? We can not infer anything, as NHST cannot provide evidence in its favor.

All things considered, there are two other questions that researchers face when performing NHST. The first one is the choice of the significance level α, for which there are no objective guidelines despite being critical to the test results. The second one is the need to previously formalize the intentions of the sampling of the results, which are usually fixed *a posteriori*; this could lead to a misreading of said results.

As shown in [42], most of these problems can be avoided by using Bayesian tests instead of NHST. In particular, the Bayesian sign test will be employed, which is the Bayesian version of the frequentist non-parametric sign test. As for its use, we will run the R package `rNPBST`, whose documentation and guide can be found in [43].

The Bayesian sign test is based on obtaining the statistical distribution of a certain parameter ρ according to the difference between the results, under the assumption that said distribution is a Dirichlet distribution. To get the distribution of ρ we count the number of times that $A-B < 0$, the number of times where there are no significant differences, and the number of times that $A-B > 0$. In order to identify cases where there are no significant differences, the region of practical equivalence (rope) $[r_{\min}, r_{\max}]$ is defined, so that $P(A \approx B) = P(\rho \in \text{rope})$. Using these results, we calculate the weights of the Dirichlet distribution and sample it to get a set of triplets with the following form:

$$[P(\rho < r_{\min}) = P(A - B < 0), \;\; P(\rho \in \text{rope}), \;\; P(\rho > r_{\max}) = P(A - B > 0)]$$

5.4 Calibration

Table 4 shows a summary of the parameter setup used for the DILS$_{CC}$.

The stop criterion is given by the *Evals* parameter, which refers to the number of evaluations of the fitness function f and which will be set at 300000. The implementation for DILS$_{CC}$ can be found at GitHub[5].

To compare our proposal with the state-of-the-art methods mentioned in Section 2.3, we will use the parameters setup shown in Table 5 for the python implementation that can be found at GitHub[6]. The implementation for the BRKGA+LS algorithm can be found in the same link as DILS$_{CC}$. In all cases, the value of k is equal to the number of classes displayed in Table 2.

[5]https://github.com/GermangUgr/DILS_CC
[6]https://github.com/GermangUgr/TFG/tree/master/Software

Parameter	Meaning	Value
Evals	Fitness function evaluations	300000
p_s	Segment size for the mutation operator	$0.3 \times n$
p_m	Maximum number of neighbors generated in each call to the LS procedure	Evals $\times 0.01$
p_r	Probability that a feature is selected from m_b for recombination	0.3
ξ	Reinitialization method tolerance	0.0
k	Output partition number of clusters	No. Classes (Table 2)

Table 4: Parameters setup used for DILS$_{CC}$.

Method name	Parameters name and values
BRKGA+LS	Evals = 300000; $\|P\| = 100$; $P_e = P_m = 0.2 * \|P\|$; $p_{\text{inherit}} = 50\%$
COPKM	`max_iter` = 300; `tolerance` $= 1 * 10^{-4}$; `init_mode` = ``rand''
CECM	`max_iter` = 300; $\alpha = 1$, $\rho = 100$, $\xi = 0.5$, `stop_threshold` $= 1 * 10^{-3}$, `init_mode` = ``rand''
LCVQE	`max_iter` = 300; `initial_centroids` = $\varnothing$
RDPM	`max_iter` = 300; $\xi_0 = 0.1$, $\xi_{\text{rate}} = 1$, λ is calculated on the basis of the mean distances in the dataset.
TVClust	`max_iter` = 300; $\alpha_0 = 1.2$, `stop_threshold` $= 5 * 10^{-4}$

Table 5: Parameters setup used for the state-of-the-art algorithms.

Parameter values have been assigned following the guidelines of the original creators of the different proposals. Since the evaluation in the experimental stage operates with a high number of datasets, tuning each parameter specifically for each dataset is not feasible. Indeed, our goal is not optimization in a case-by-case basis but instead a comparison in the most general and plausible scenario. Therefore, given that the purpose of this work is to draw a fair comparison between the algorithms and assess their robustness in a common environment with multiple datasets, we have not included a tuning step to maximize any particular performance metric.

As previously stated, the programming language chosen to implement all methods described in this paper is Python. All our experiments have been carried out in the University of Granada Hercules Computing Server, which features 51 computing nodes with a 2.8 GHz

Intel i7 processor, 24 GB of RAM and 1TB SATA2 hard drive disk in each node. Two Gigabit Ethernet internal nets are used to interconnect nodes. Ubuntu 18.04.1 LTS is installed in each node.

6 Experimental Results

In this section, Tables from 6 to 8 are presented. These tables display the results obtained by DILS$_{CC}$ and six state-of-the-art methods to be compared for each dataset and constraint set. Figures 2 to 4 offer a visual summary of the information contained in the tables.

Since some of the methods compared involve non-deterministic procedures, the results may vary from one run to another. To lessen the effect this may have on the results, each method will be executed 30 times to each dataset and constraint set, so that the measures shown in Tables 6 to 8 correspond to the average of the 30 runs. The "Avg." column shows the average results for every compared method in every dataset, while the "SD" column shows the Standard Deviation. With 25 datasets, 3 constraint sets for each one of them, 7 different methods, and 30 runs, a total of 15750 experiments for our study have been conducted.

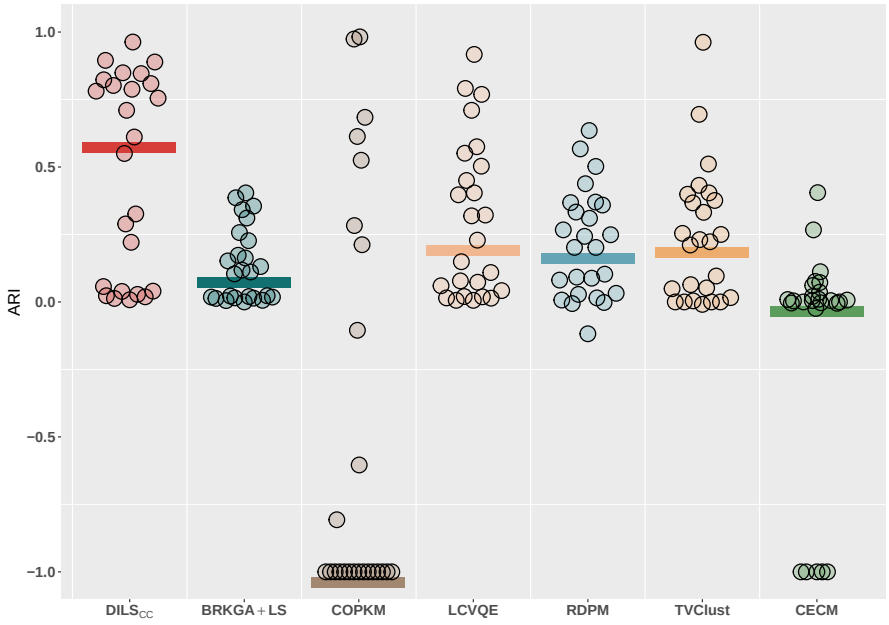


Figure 2: CS$_{10}$ comparative average plot.

It should be noted that there are some missing results in these tables. In the case of the COPKM algorithm, this is due to the fact that it is highly dependent on the order in which constraints are analyzed. It is possible that COPKM cannot find a solution, even though it is always feasible, since the constraints have been generated based on the true labels. In the case of CECM, some of the results are not available because the memory structures that hold the algorithm grow non-linearly with the number of classes and the number of features of

ARI Results for CS_{10}														
Dataset	DILS$_{CC}$		BRKGA+LS		COPKM		LCVQE		RDPM		TVClust		CECM	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**.611**	**.000**	.118	.078	-	-	.450	.000	.267	.081	.211	.184	.005	.001
Breast Cancer	.755	.040	.173	.198	-.604	.792	**.917**	**.000**	.502	.000	.016	.048	.005	.013
Bupa	**.889**	**.022**	.355	.233	-	-	.149	.009	-.006	.003	-.009	.002	-.005	.002
Circles	**.781**	**.014**	.104	.055	-	-	.006	.008	.243	.280	.404	.377	.072	.049
Ecoli	.039	.008	.018	.008	-	-	.404	.066	.333	.057	**.432**	**.223**	-	-
Glass	.008	.011	.014	.012	.212	.027	.229	.007	.202	.077	**.231**	**.040**	-	-
Haberman	**.802**	**.029**	.020	.023	-.807	.580	.019	.003	.088	.057	.332	.043	.004	.010
Hayesroth	.057	.033	.015	.011	-	-	.073	.046	**.103**	**.034**	.064	.076	.061	.013
Heart	**.846**	**.046**	.131	.179	-	-	.006	.007	.032	.004	.223	.213	.000	.008
Ionosphere	**.809**	**.040**	.018	.022	-	-	.060	.000	.202	.047	.004	.011	.112	.094
Iris	.550	.065	.311	.083	-.105	.895	**.769**	**.000**	.567	.064	.511	.074	.405	.283
Led7Digit	.013	.006	.005	.003	**.525**	**.036**	.503	.024	.359	.031	.254	.055	-	-
Monk2	.823	.032	.404	.130	**.982**	**.000**	.575	.000	.092	.031	.096	.292	.009	.025
Moons	**.963**	**.022**	.227	.124	-	-	.319	.000	.310	.028	.962	.027	.267	.161
Mov. Libras	.019	.008	.000	.004	.283	.013	**.322**	**.017**	.249	.025	.000	.000	-	-
Newthyroid	.040	.053	.013	.019	-	-	**.791**	**.009**	.370	.150	.695	.206	-.004	.009
Saheart	.788	.025	.163	.092	**.974**	**.000**	.020	.011	.028	.023	.367	.372	.007	.003
Sonar	**.710**	**.068**	.023	.020	-	-	.109	.000	.007	.007	.000	.000	-.003	.007
Soybean	.289	.062	.342	.165	.613	.152	.551	.007	**.635**	**.034**	.000	.000	.076	.087
Spectfheart	**.895**	**.039**	.257	.083	-	-	.014	.000	-.118	.008	.000	.000	-.024	.032
Spiral	**.849**	**.042**	.386	.310	-	-	.042	.000	.015	.009	.049	.041	.036	.006
Tae	.028	.019	.022	.022	-	-	.015	.000	-.002	.004	**.052**	**.022**	.000	.000
Vehicle	.023	.009	.006	.005	-	-	.078	.001	.081	.000	**.250**	**.125**	.020	.013
Wine	.326	.051	.152	.051	-	-	**.397**	**.000**	.368	.002	.376	.097	.010	.008
Zoo	.221	.031	.111	.064	.684	.099	**.710**	**.048**	.438	.110	.399	.116	-	-
Mean	**.485**		.136		-.490		.301		.214		.236		−.158	

Table 6: Experimental results obtained for CS_{10} by DILS$_{CC}$ and the state-of-the-art methods.

the dataset to be analyzed. It is established that in these cases, the ARI value considered for the mean calculation—and the forthcoming statistical analysis of results—is −1.000, which is the worst possible ARI value.

Table 6 shows the results for the SC_{10} constraint set, which are visually summarized in Table 2. It can be observed that DILS$_{CC}$ represents a consistent improvement as compared to all other 6 methods. This is due to the fact that the exploitation capacity of DILS$_{CC}$ allows it to take better advantage the constraints, even with the lowest level of constraint-based information we work with.

The results obtained with the SC_{15} constraint set are presented in Table 7, and summarized in Figure 3. It is noticeable how DILS$_{CC}$ and COPKM are able to obtain ARI values equal to 1, meaning that the resulting partitions of the datasets perfectly match the true partitions.

ARI Results for CS_{15}														
Dataset	DILS$_{CC}$		BRKGA+LS		COPKM		LCVQE		RDPM		TVClust		CECM	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**.957**	**.000**	.418	.368	-	-	.379	.018	.335	.084	.261	.201	-.005	.003
Breast Cancer	.792	.016	.915	.031	**1.00**	**.000**	.979	.000	.502	.000	.096	.285	-.006	.008
Bupa	.993	.006	.994	.006	**1.00**	**.000**	-.002	.000	-.006	.003	.092	.303	.000	.000
Circles	**1.00**	**.000**	.987	.013	**1.00**	**.000**	.011	.010	.408	.292	.910	.270	.066	.049
Ecoli	.091	.026	.024	.012	-	-	.513	.058	.402	.103	**.564**	**.241**	-	-
Glass	.076	.042	.025	.009	-	-	**.286**	**.028**	.248	.016	.244	.066	-	-
Haberman	**1.00**	**.000**	.490	.392	**1.00**	**.000**	.001	.001	.079	.048	.973	.000	.006	.009
Hayesroth	**.478**	**.066**	.131	.057	-	-	.087	.055	.111	.021	.097	.082	.002	.013
Heart	**1.00**	**.000**	.980	.020	**1.00**	**.000**	.053	.000	.030	.012	.435	.433	-.000	.001
Ionosphere	.973	.009	.981	.020	**1.00**	**.000**	.004	.000	.261	.046	.004	.011	.129	.094
Iris	.832	.051	.240	.136	-	-	**.941**	**.000**	.543	.006	.524	.127	.228	.323
Led7Digit	.012	.002	.003	.003	-	-	**.557**	**.029**	.454	.033	.261	.051	-	-
Monk2	.899	.015	.951	.031	**1.00**	**.000**	.671	.000	.144	.028	.098	.301	.100	.025
Moons	**1.00**	**.000**	.982	.018	**1.00**	**.000**	.639	.000	.470	.065	1.00	.000	.299	.161
Mov. Libras	.018	.006	.002	.002	-.742	.516	**.334**	**.025**	.255	.032	.000	.000	-	-
Newthyroid	.390	.113	.104	.076	-	-	.835	.010	.455	.156	**.848**	**.191**	.013	.018
Saheart	.870	.021	.815	.265	**1.00**	**.000**	.017	.000	.030	.023	.808	.384	.003	.003
Sonar	**.981**	**.000**	.947	.067	-	-	.037	.013	.014	.013	.000	.000	-.001	.001
Soybean	.468	.045	.421	.137	**.656**	**.200**	.550	.011	.593	.058	.000	.000	.153	.232
Spectfheart	**1.00**	**.000**	.861	.302	.983	.000	.167	.000	-.116	.008	.000	.000	.032	.032
Spiral	**1.00**	**.000**	.594	.333	-	-	.022	.007	.011	.009	.411	.482	.012	.006
Tae	**.386**	**.045**	.077	.077	-	-	.046	.000	.002	.007	.053	.018	-.000	.000
Vehicle	.066	.014	.023	.014	**1.00**	**.000**	.083	.003	.081	.000	.454	.159	-.006	.007
Wine	**.740**	**.047**	.163	.083	-	-	.395	.000	.369	.001	.430	.129	.001	.002
Zoo	.193	.066	.109	.066	.416	.023	**.724**	**.059**	.434	.112	.423	.138	-	-
Mean	**.649**		.489		.013		.333		.244		.359		−.159	

Table 7: Experimental results obtained for CS_{15} by DILS$_{CC}$ and the state-of-the-art methods.

The fact that DILS$_{CC}$ is always able to output a partition of the datasets is a noteworthy advantage over COPKM. Even in the cases where COPKM is unable to produce a partition, DILS often finds the optimal or a near-optimal partition.

In Table 8, which shows the results obtained with the SC_{20} constraint set, we can see how the quality of the results of DILS$_{CC}$ scales with the number of constraints, as well as that of methods such as BRKGA+LS or COPKM (the latter being more limited in other aspects). This same trend can be confirmed in Figure 4. The TVClust method should also be highlighted, as it is able to find the optimal solution in two cases, which implies a significant improvement over its performance on previous constraint sets. It is also worth noting that DILS$_{CC}$ provides the best average ARI for each of the three constraint sets analyzed in this paper.

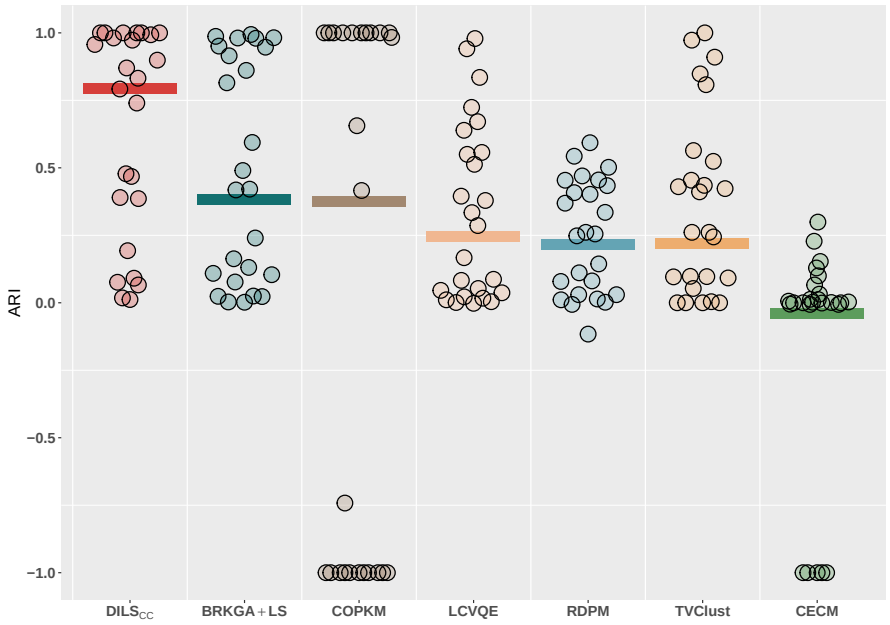


Figure 3: CS$_{15}$ comparative average plot.

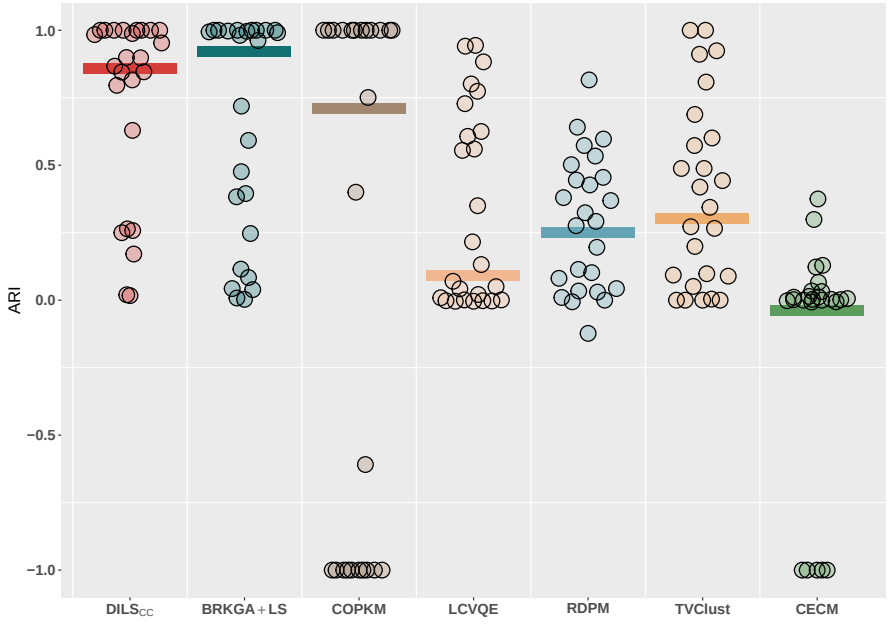


Figure 4: CS$_{20}$ comparative average plot.

Considering that BRKGA and DILS$_{CC}$ are the two metaheuristic approaches to the constrained clustering problem presented in this paper, we want to highlight the clear improvement that the latter constitutes over BRKGA. Both methods are able to scale the quality of the

ARI Results for CS_{20}														
Dataset	DILS$_{CC}$		BRKGA+LS		COPKM		LCVQE		RDPM		TVClust		CECM	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**1.00**	**.000**	**1.00**	**.000**	-	-	.050	.000	.380	.184	.419	.324	-.007	.001
Breast Cancer	.796	.009	.962	.015	**1.00**	**.000**	.944	.000	.502	.000	.098	.291	.014	.013
Bupa	.988	.007	.996	.005	**1.00**	**.000**	-.002	.000	-.006	.003	.093	.302	-.002	.002
Circles	**1.00**	**.000**	**1.00**	**.000**	**1.00**	**.000**	.001	.002	.573	.193	.911	.266	.066	.049
Ecoli	.264	.126	.039	.026	-	-	.607	.072	.427	.071	**.688**	**.243**	-	-
Glass	.258	.131	.084	.040	-	-	.216	.012	.276	.017	**.344**	**.165**	-	-
Haberman	**1.00**	**.000**	**1.00**	**.000**	**1.00**	**.000**	.001	.000	.102	.053	**1.00**	**.000**	.002	.010
Hayesroth	**.816**	**.044**	.395	.136	-	-	.132	.048	.114	.027	.266	.259	.002	.013
Heart	**1.00**	**.000**	**1.00**	**.000**	**1.00**	**.000**	.042	.002	.034	.005	.488	.468	.003	.008
Ionosphere	.984	.006	.994	.006	**1.00**	**.000**	-.002	.000	.324	.042	.004	.011	.129	.094
Iris	**.953**	**.026**	.476	.237	-	-	.941	.000	.597	.074	.573	.138	.375	.283
Led7Digit	.017	.006	.008	.003	-	-	**.555**	**.028**	.534	.044	.272	.057	-	-
Monk2	.899	.017	.991	.008	**1.00**	**.000**	.883	.000	.196	.104	.199	.401	.034	.025
Moons	**1.00**	**.000**	**1.00**	**.000**	**1.00**	**.000**	.774	.000	.816	.095	**1.00**	**.000**	.299	.161
Mov. Libras	.020	.009	.003	.003	-	-	**.350**	**.024**	.292	.029	.000	.000	-	-
Newthyroid	.845	.016	.719	.107	-	-	.801	.001	.445	.151	**.924**	**.138**	.000	.009
Saheart	.867	.006	.981	.006	**1.00**	**.000**	.009	.001	.030	.022	.808	.384	.003	.003
Sonar	**1.00**	**.000**	**1.00**	**.000**	**1.00**	**.000**	-.004	.000	.043	.042	.000	.000	.006	.007
Soybean	.629	.057	.592	.230	-.609	.783	.560	.000	**.641**	**.050**	.000	.000	.123	.087
Spectfheart	**1.00**	**.000**	.997	.006	**1.00**	**.000**	-.004	.000	-.123	.008	.000	.000	.032	.032
Spiral	**1.00**	**.000**	**1.00**	**.000**	.400	.917	-.003	.000	.010	.006	.601	.488	.012	.006
Tae	**.846**	**.031**	.247	.128	-	-	.021	.000	.000	.006	.051	.015	-.000	.000
Vehicle	.171	.049	.043	.024	**1.00**	**.000**	.070	.000	.081	.000	.089	.026	-.006	.007
Wine	**.898**	**.024**	.383	.183	-	-	.625	.000	.369	.002	.488	.165	.011	.008
Zoo	.250	.060	.115	.073	.751	.117	**.728**	**.069**	.455	.111	.443	.131	-	-
Mean	**.740**		.641		.102		.332		.284		.390		−.156	

Table 8: Experimental results obtained for CS_{20} by DILS$_{CC}$ and the state-of-the-art methods.

solutions with the amount of available constraint-based information, although the DILS$_{CC}$ exhibits a better behavior from the smallest constraint set CS_{10}, where the largest average difference between these to methods is found. The aim of the DILS$_{CC}$ method is to preserve a good exploration-exploitation trade-off while maintaining an exploitation-oriented general scheme (ILS). This gives DILS$_{CC}$ the ability to deeply exploit certain regions of the solutions space when needed (which we have found to be beneficial to the constrained clustering problem), whereas BRKGA does not include mechanisms to explicitly exploit a particular local-optima region when it is likely to contain the general optimal solution. Notice that BRKGA does include a local search optimization procedure, making it a memetic algorithm. Although it does not transfer the results of this procedure to the population itself, it is only used to update the best solution found so far if required.

7 Statistical Analysis of Results

With the results obtained by all methods for a total of 75 different datasets—the 25 datasets in combination with the three constraint sets for each one—an empirical analysis can be performed. In doing so, we can statistically determine whether DILS$_{CC}$ represents a significant improvement over previous proposals.

With the notation introduced in Section 5.3 in mind, the results obtained by a given state-of-the-art method will be referred to as sample A, and the results obtained with DILS$_{CC}$ as sample B.

One of the major advantages of the Bayesian sign test is the possibility of obtaining a highly illustrative visual representation of its results. It allows us to produce a representation of the triplet set in the form of a heatmap where each triplet constitutes one point whose location is given by barycentric coordinates. In that context, each of the triplet values will be associated with each of the three vertices of an equilateral triangle. In order to find out where a certain triplet will be placed within the triangle, each of its three values will be considered and a parallel line will delimit the opposing side of the corresponding vertex. The separation between a triangle side and its parallel line will be proportional to the associated triplet value, so that the higher the value, the closer the line will be to the vertex. The location where the three lines intersect is marked with a point. As the values of every triplet describe a probability distribution, and therefore they must add up to one, all triplets will lie in some point within the triangle in all likelihood. The color indicates the density of points in a given region, with yellow representing a high density and red a low density.

Figure 5 shows all six heatmaps obtained when applying the Bayesian sign test to DILS$_{CC}$ and the state-of-the-art methods. All heatmaps suggest that the test assigns a high probability to $A - B < 0$, since most triplets are presented in the lower left third of the heatmap. Bearing this in mind, and considering that ARI is a measure to maximize, it can be observed that the Bayesian sign test provides strong evidence in favor of DILS$_{CC}$. The same test indicates that the results offered by the the state-of-the-art are not equivalent to those offered by DILS$_{CC}$, and are lower in quality.

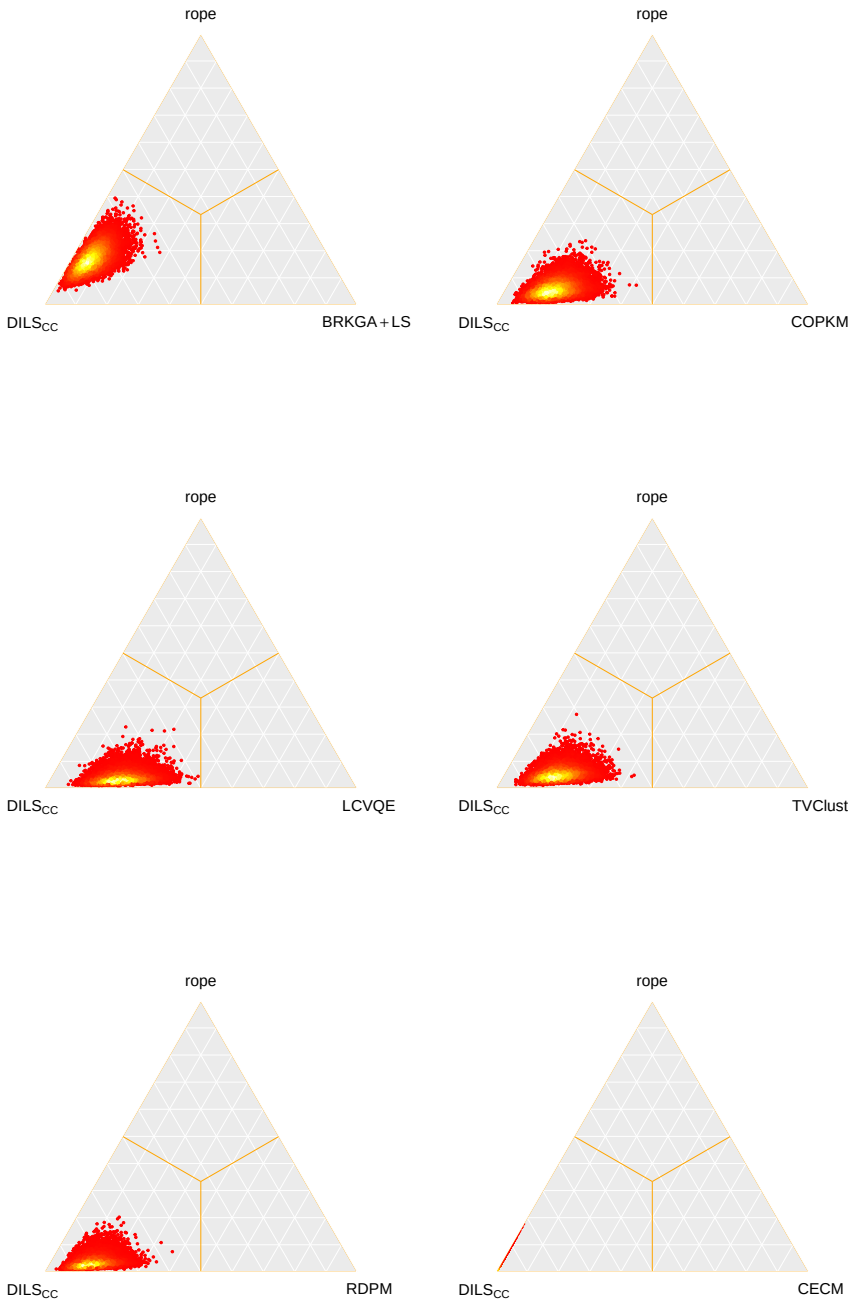


Figure 5: Heatmaps comparing DILS with the state-of-the-art methods. With L being equivalent to A and R being equivalent to B (results obtained by DILS$_{CC}$).

8 Discussion and Future Work

In this Section, a discussion regarding the main discoveries made during the development of our study is presented, as well as future lines of research that may be of interest for the reader. Throughout this paper, the suitability of the proposed method to address highly constrained problems has been analyzed, although in some cases DILS is not able to provide better results than those a random model would produce for certain datasets. Please note that no optimization procedure was used to alter any parameter. As a result, it is possible that with a different parameter configuration, better results could be obtained for these datasets.

Regarding future research lines, our proposal is centered on performing the optimization process, which was previously mentioned, for DILS parameters. This would lead to a better understanding of the influence over the DILS optimization process. Even if the majority of the datasets used in our experiments are real-world datasets, it would be interesting to apply DILS to a real-world problem in which the amount of side information would be probably limited. Performing this action, would allow us to test the robustness of the proposed method in a less controlled environment, i.e., out of the laboratory.

We also want to put clear that the experimental study presented in this paper can be extended in the future. Dozens of constrained clustering methods have been proposed over the years, among the most renowned ones have been chosen for our comparison. Future research lines may include other methods in different frameworks such as GC+PR+LS [35], classic spectral learning approaches [44], density-based clustering [45], and genetic multi-objective optimization [46], among many others.

9 Conclusions

In this paper, the DILS heuristic method has been proposed, along with its application to the SSL constrained clustering problem, which is referred to as DILS$_{CC}$. Focusing on instance-level ML and CL constraints, DILS$_{CC}$ has proven that heuristic techniques constitute a competitive approach to constrained clustering, being able to scale the quality of the results in a way that is directly proportional to the number of constraints.

Supported by the Bayesian statistical tests, it can be objectively proven that the DILS$_{CC}$ approach is significantly better than the state-of-the-art methods. DILS$_{CC}$ has proven to be better when considering the quality and availability of the solutions, namely in cases where large sets of constraints are analyzed.

Acknowledgements

This study is has been funded by the research projects TIN2017-89517-P and PP2016.PRI.I.02.

References

[1] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[2] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[3] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2):245–284, 2015.

[4] Samaneh Hosseini Semnani, Otman A Basir, and Peter Van Beek. Constrained clustering for flocking-based tracking in maneuvering target environment. *Robotics and Autonomous Systems*, 83:243–250, 2016.

[5] Hao Wu and Saurabh Prasad. Semi-supervised deep learning using pseudo labels for hyperspectral image classification. *IEEE Transactions on Image Processing*, 27(3):1259–1270, 2017.

[6] Alex Seret, Thomas Verbraken, and Bart Baesens. A new knowledge-based constrained clustering approach: Theory and application in direct marketing. *Applied Soft Computing*, 24:316–327, 2014.

[7] Son T Mai, Sihem Amer-Yahia, Sébastien Bailly, Jean-Louis Pépin, Ahlame Douzal Chouakria, Ky T Nguyen, and Anh-Duong Nguyen. Evolutionary active constrained clustering for obstructive sleep apnea analysis. *Data Science and Engineering*, 3(4):359–378, 2018.

[8] Firas Saidi, Zouheir Trabelsi, and Henda Ben Ghazela. A novel approach for terrorist sub-communities detection based on constrained evidential clustering. In *2018 12th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–8. IEEE, 2018.

[9] Safa Ben Ayed, Zied Elouedi, and Eric Lefevre. Cevm: Constrained evidential vocabulary maintenance policy for cbr systems. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 579–592. Springer, 2019.

[10] Andreas Brieden, Peter Gritzmann, and Fabian Klemm. Constrained clustering via diagrams: A unified theory and its application to electoral district design. *European Journal of Operational Research*, 263(1):18–34, 2017.

[11] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 577–584. Morgan Kaufmann Publishers Inc., 2001.

[12] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, MSR-TR-2000-65, Microsoft Research, 2000.

[13] Ian Davidson and Sugato Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from data*, 1:1–41, 2007.

[14] Jana Schmidt, Elisabeth Maria Brandle, and Stefan Kramer. Clustering with attribute-level constraints. In *2011 IEEE 11th International Conference on Data Mining*, pages 1206–1211. IEEE, 2011.

[15] Jinlong Wang, Shunyao Wu, and Gang Li. Clustering with instance and attribute level side information. *International Journal of Computational Intelligence Systems*, 3(6):770–785, 2010.

[16] Martin HC Law, Alexander Topchy, and Anil K Jain. Clustering with soft and group constraints. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 662–670. Springer, 2004.

[17] Ian Davidson and SS Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 International Conference on Data Mining*, pages 138–149. SIAM, 2005.

[18] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition, 2010.

[19] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search: Framework and applications. In *Handbook of metaheuristics*, pages 363–397. Springer, 2010.

[20] Claudia Archetti, Dominique Feillet, Andrea Mor, and M Grazia Speranza. An iterated local search for the traveling salesman problem with release dates and completion time minimization. *Computers & Operations Research*, 98:24–37, 2018.

[21] Mustafa Avci and Seyda Topaloglu. A multi-start iterated local search algorithm for the generalized quadratic multiple knapsack problem. *Computers & Operations Research*, 83:54–65, 2017.

[22] Hayet Chentli, Rachid Ouafi, and Wahiba Ramdane Cherif-Khettaf. Impact of iterated local search heuristic hybridization on vehicle routing problems: Application to the capacitated profitable tour problem. In *International Conference on Operations Research and Enterprise Systems*, pages 80–101. Springer, 2018.

[23] Alejandro Estrada-Moreno, M Savelsbergh, Angel A Juan, and Javier Panadero. Biased-randomized iterated local search for a multiperiod vehicle routing problem with price discounts for delivery flexibility. *International Transactions in Operational Research*, 26(4):1293–1314, 2019.

[24] Hassan Zohali, Bahman Naderi, Mohammad Mohammadi, and Vahid Roshanaei. Reformulation, linearization, and a hybrid iterated local search algorithm for economic lot-sizing and sequencing in hybrid flow shop problems. *Computers & Operations Research*, 104:127–138, 2019.

[25] Fuqing Zhao, Xuan He, Guoqiang Yang, Weimin Ma, Chuck Zhang, and Houbin Song. A hybrid iterated local search algorithm with adaptive perturbation mechanism by success-history based parameter adaptation for differential evolution (shade). *Engineering Optimization*, pages 1–17, 2019.

[26] Eduardo Raul Hruschka, Ricardo JGB Campello, Alex A Freitas, et al. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(2):133–155, 2009.

[27] Adán José-García and Wilfrido Gómez-Flores. Automatic clustering using nature-inspired metaheuristics: A survey. *Applied Soft Computing*, 41:192–213, 2016.

[28] Satyasai Jagannath Nanda and Ganapati Panda. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary computation*, 16:1–18, 2014.

[29] Marie-Hélène Masson and Thierry Denoeux. ECM: An evidential version of the fuzzy c-means algorithm. *Pattern Recognition*, 41(4):1384–1397, 2008.

[30] Violaine Antoine, Benjamin Quost, M-H Masson, and Thierry Denoeux. Cecm: Constrained evidential c-means algorithm. *Computational Statistics & Data Analysis*, 56(4):894–914, 2012.

[31] Dan Pelleg and Dorit Baras. K-means with large and noisy constraint sets. In *European Conference on Machine Learning*, pages 674–682. Springer, 2007.

[32] Daniel Khashabi, John Wieting, Jeffrey Yufei Liu, and Feng Liang. Clustering with side information: From a probabilistic model to a deterministic algorithm. *arXiv preprint arXiv:1508.06235*, 2015.

[33] José Fernando Gonçalves and Mauricio GC Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011.

[34] James C Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160, 1994.

[35] Rudinei Martins de Oliveira, Antonio Augusto Chaves, and Luiz Antonio Nogueira Lorena. A comparison of two hybrid methods for constrained clustering problems. *Applied Soft Computing*, 54:256–266, 2017.

[36] Antonio Augusto Chaves, José Fernando Gonçalves, and Luiz Antonio Nogueira Lorena. Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem. *Computers & Industrial Engineering*, 124:331–346, 2018.

[37] William M Spears and Kenneth D De Jong. On the virtues of parameterized uniform crossover. Technical report, NAVAL RESEARCH LAB WASHINGTON DC, 1995.

[38] Isaac Triguero, Sergio González, Jose M Moyano, Salvador García, Jesús Alcalá-Fdez, Julián Luengo, Alberto Fernández, Maria José del Jesús, Luciano Sánchez, and Francisco Herrera. KEEL 3.0: an open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems*, 10(1):1238–1249, 2017.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[41] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[42] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

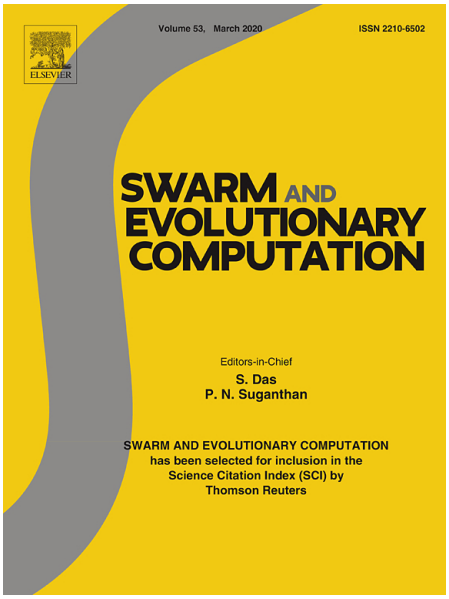
[43] Jacinto Carrasco, Salvador García, María del Mar Rueda, and Francisco Herrera. rnpbst: An r package covering non-parametric and bayesian statistical tests. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 281–292. Springer, 2017.

[44] Kamvar Kamvar, Sepandar Sepandar, Klein Klein, Dan Dan, Manning Manning, and Christopher Christopher. Spectral learning. In *International Joint Conference of Artificial Intelligence*. Stanford InfoLab, 2003.

[45] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. C-dbscan: Density-based clustering with constraints. In *International workshop on rough sets, fuzzy sets, data mining, and granular-soft computing*, pages 216–223. Springer, 2007.

[46] Nobukazu Matake, Tomoyuki Hiroyasu, Mitsunori Miki, and Tomoharu Senda. Multiobjective clustering with automatic k-determination for large-scale data. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 861–868, 2007.

3 ME-MOEA/D_{CC}: Multiobjective Constrained Clustering Through Decomposition-based Memetic Elitism



- **Journal:** Swarm and Evolutionary Computation (SWEVO)
- **JCR Impact Factor:** 10.267
- **Rank:** 14/145
- **Quartile:** Q1
- **Category:** Computer Science, Artificial Intelligence
- **Status:** Published

Ref.: Gonzalez-Almagro, G., Rosales-Perez, A., Luengo, J., Cano, J. R., & Garcia, S. (2021). ME-MEOA/Dcc: Multiobjective constrained clustering through decomposition-based memetic elitism. Swarm and Evolutionary Computation, 66, 100939. DOI: https://doi.org/10.1016/j.swevo.2021.100939.

A preliminary version of this paper was published at the 2020 Genetic and Evolutionary Computation Conference (GECCO) with title: Improving Constrained Clustering Via Decomposition-based Multiobjective Optimization with Memetic Elitism (DOI: https://doi.org/10.1145/3377930.3390187). This conference was held thematically due to the COVID-19 pandemic. This work was **nominated for the best conference paper award**.

ME-MOEA/D$_{CC}$: Multiobjective Constrained Clustering Through Decomposition-based Memetic Elitism

Germán González Almagro [*,a,b] **Alejandro Rosales-Pérez** [c] **Julián Luengo** [a,b]

José-Ramón Cano [d,b] **Salvador García** [a,b]

[a] *Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada, Granada, Spain*
[b] *Andalusian Institute of Data Science and Computational Intelligence (DaSCI)*
[c] *Centro de Investigación en Matemáticas (CIMAT) A.C., Nuevo León, Mexico*
[d] *Department of Computer Science, University of Jaén, Jaén, Spain*

Abstract

Traditionally unsupervised, clustering techniques have received renewed attention recently, as they have been shown to produce better results when provided with incomplete information about the dataset in the form of constraints. Combining classic clustering and constraints leads to constrained clustering, a semi-supervised learning problem still unexplored in many aspects. Based on the exploration-exploitation requirements of constrained clustering, a memetic elitist multiobjective evolutionary algorithm based on decomposition is proposed, which combines classic multiobjective optimization strategies with single-objective optimization procedures. The application scheme of our proposal for the constrained clustering problem is scrutinized and compared to several state-of-the-art methods for 20 datasets with incremental levels of constraint-based information. Experimental results, supported by Bayesian statistical testing, show a consistent improvement in clustering and multiobjective optimization related measures in favor of our proposal over the state-of-the-art.

Keywords Multiobjective Optimization · Memetic Elitism · Constrained Clustering · Instance-level Constraints · Bayesian Statistical Testing

* Corresponding Author (germangalmagro@ugr.es)
Email addresses: julianlm@decsai.ugr.es (Alejandro Rosales-Pérez), arosalesp85@gmail.com (Julián Luengo), jrcano@ujaen.es (José Ramón Cano), salvagl@decsai.ugr.es (Salvador García)

1 Introduction

Clustering has always been a key research area in machine learning. It is able to provide valuable insight within the unsupervised learning paradigm, where no information other than an unlabeled dataset by itself is available for learning. However, new types of information can be added to the classic clustering framework. Subsequently, better and more accurate results are obtained. These new types of information may be given in many forms—not only in the form of labels—, moving the problem from the unsupervised learning framework to the machine learning paradigm known as Semi-Supervised Learning (SSL) [1, 2]. Background knowledge about the problem domain can be incorporated into clustering techniques to improve their capabilities. When the new type of information is given as constraints, the problem is called Constrained Clustering (CC). In CC, a set of constraints is used to guide the clustering process, as the resulting partition of the dataset is required to satisfy as many constraints as possible, in addition to meeting the proper characteristics of a classic clustering partition. CC has been successfully applied in many fields of knowledge, among which it is worth mentioning: satellite image time series [3], storage location assignment in warehouses [4], obstructive sleep apnea analysis [5], electoral district design [6], terrorist sub-communities detection [7], documents clustering [8] and lane finding in GPS data [9] among others.

Many distinctions can be made within the general CC framework. Three main ways to include constraints into the clustering problem can be found in the literature: cluster-level [10], instance-level [11] and feature-level CC [12]. Two main strategies to take constraints into account are commonly used in CC: (1) in *distance-based* methods a new metric reflecting the information contained in the constraint set is learned [13, 14, 15], (2) in *clustering-engine* adapting constraints are used as hints to guide the clustering process by modifying the clustering engine to include them [9, 16, 17]. Finally, the concepts of soft [18] and hard [9] constraints can also be found in the literature. Concerning hard constraints, methods are forced to output partitions satisfying all constraints, whereas soft constraints allow output partitions with some unsatisfied constraints. This study is focused on soft instance-level Must-link (ML) and Cannot-link (CL) constraints, which tell us if two specific instances of a dataset must be placed in the same or in different clusters, respectively.

Using ML and CL constraints makes the CC problem **NP**-complete [19]. Due to this fact, it is easy to understand why approximate methods such as metaheuristics represent a promising approach to the CC problem. Metaheuristics have been successfully applied to many real-world problems such as: crude oil time series [20], COVID-19 disease recognition through X-ray images [21], digital currency forecasting [22] and control of unmanned aerial vehicles [23], among others. The classic clustering problem is not an exception to this trend, with many studies presenting excellent results [24, 25, 26], although very little work has been done on CC. Within the field of metaheuristics, Multiobjective Evolutionary Algorithms (MOEAs) are particularly interesting to approach the clustering problem. Many measures can be used to guide the clustering process towards a quality solution [27], although it is often not straightforward to integrate constraints in a single function that could be optimized by a standard optimizer. Such problem is also found in the CC framework, as even more

quality measures need to be used to include constraints. Constraint-related quality measures often contradict classic clustering quality measures, causing difficulties to integrate them in a single-objective function optimizable by a single-objective evolutionary algorithm. Multiobjective optimization schemes provide us with a powerful tool to overcome all these drawbacks.

A great variety of MOEAs have been developed to solve problems with different characteristics and needs of exploration and exploitation [28]; to name a few: Non-dominated Sorting Genetic Algorithm III (NSGA-III) [29], MultiObjective Evolutionary Algorithm based on Decomposition (MOEA/D) [30], Strength Pareto Evolutionary Algorithm 2 (SPEA2) [31], Pareto Envelope-based Selection Algorithm II (PESA-II) [32] or MultiObjective Genetic Algorithm based on Error Correcting Output Codes (MOGAECOC) [33]. Several studies can be found showing the advantages of using MOEAs in order to approach the classic clustering problem [34, 35, 36], although very little work has been done on investigating their suitability for the CC problem. MOEAs are presented as a promising approach to the CC problem thanks to their capability to optimize objective functions which are often contradictory. The CC problem is an example of the previous, as classic clustering objective functions tend to create hyperspherical clusters and constrained clustering objective functions (such as the infeasibility) are used to deviate the clustering process from this trend. Two relevant studies in this topic are found in the literature: in [37] the MOCK technique (an adaptation of PESA-II [32]) is extended to include constraints, and in [38] an MOEA is used to perform spectral clustering taking into account a set of constraints, and finally K-means is applied to perform clustering. In [39] an MOEA is applied to the self-labeling problem, which is not a clustering problem (as constrained clustering) but an SSL classification problem.

Memetic evolutionary algorithms are widely used to improve and accelerate metaheuristics towards high quality solutions [40, 41]. Guiding the evolutionary process of any given MOEA towards high quality solutions by means of memetic components is a common practice in real-world applications [42]. MOEA/D is not an exception to this trend. Mechanisms ranging from redesigning its selection mechanism [43] or an elitist parallelization for large-scale problems [44], to combinations with local search procedures by means of a hybridization with NSGA-II [45] or direct inclusion methods [46] are used to accelerate MOEA/D.

In this study, we extend our previous work in [47], where a memetic elitist version of MOEA/D [30] is applied to constrained clustering. The general optimization scheme of our proposal ME-MOEA/D (Memetic Elitist - Multiobjective Optimization Evolutionary Algorithm based on Decomposition) is examined in detail. Unlike MOEA/D, our proposal implements memetic elitism by means of any single-objective optimization procedure and a dominance-guided sorting mechanism, which is applied to the full population to select elite individuals. ME-MOEA/D can adaptively select the single-objective function to apply the single-objective optimization procedure on, at any given stage of the optimization process, while also allowing the user to control this feature through parameter settings.

ME-MOEA/D is able to fuse classic multiobjective optimization methods with single-objective procedures such as the Local Search (LS) by adaptively choosing a single-objective function to optimize for each individual. For the purposes of this study, we improve the ap-

plication of ME-MOEA/D to the CC problem, which we call ME-MOEA/D$_{CC}$. We include a new initialization method, a biased crossover operator and an external population limiting method. To the best of our knowledge, no previous study combines these genetic components and a memetic elitist version of MOEA/D to approach the CC problem. We further prove the capabilities of ME-MOEA/D$_{CC}$ to produce high quality results for the CC problem by means of a more extensive experimentation than the one presented in [47]. The quality of the results from the point of view of multiobjective optimization is compared throughout this study.

The rest of this paper is structured as follows: background related to CC and multiobjective optimization is introduced in Section 2; our proposal ME-MOEA/D is described in Section 3, and its application scheme for the CC problem (ME-MOEA/D$_{CC}$) is presented in 4; the experimental setup is explained in Section 5; results and their analysis are discussed in Sections 6 and 7, respectively; finally, conclusions are presented in Section 8.

2 Background

In this section, we explore the background knowledge concerning CC (Section 2.1), along with its computational complexity (Section 2.2). We will also present the basis of multiobjective optimization (Section 2.3).

2.1 Constrained Clustering

Partitional clustering can be defined as the task of grouping the n instances of a dataset X into k clusters. Each instance is described by u features. More formally, $X = \{x_1, \cdots, x_n\}$, with the ith instance noted as $x_i = (x_i^1, \cdots, x_i^u)$. A class label l_i is assigned to each instance $x_i \in X$. As a result, the set of labels $L = \{l_1, \cdots, l_n\}$, with $l_i \in \{1, \cdots, k\}$ is obtained. L effectively splits X into k non-overlapping clusters c_i to form a partition called C. Assigning an instance to a cluster depends on the similarity to the rest of elements in that cluster, and the dissimilarity to the rest of instances of the dataset. The similarity between two instances can be obtained with some kind of distance measurement [48].

In instance-level Constrained Clustering (CC) a set of constraints is given to guide the clustering process. Constraints consist of pairs of instances that state whether the two instances must be placed in the same cluster or in different clusters. Better results can be obtained by integrating these constraints into the clustering process. Constraints can be formalized as follows:

- Must-link (ML) constraints $C_=(x_i, x_j)$: instances x_i and x_j from X must be placed in the same cluster.
- Cannot-link (CL) constraints $C_\neq(x_i, x_j)$: instances x_i and x_j from X cannot be assigned to the same cluster.

In constrained clustering, a partition of k clusters $C = \{c_1, \cdots, c_k\}$ must be found given a dataset X and two constraint sets $C_=(x_i, x_j)$ and $C_\neq(x_i, x_j)$. The number of instances in each

cluster c_i must add up to the total number of instances in the dataset X, which is defined as $n = |X| = \sum_{i=1}^{k} |c_i|$.

ML constraints are reflexive, transitive and symmetric, and therefore they constitute an equivalence relation. Given constraints $C_{=}(x_a, x_b)$ and $C_{=}(x_b, x_c)$ then $C_{=}(x_a, x_c)$ is verified. In addition, if $x_a \in c_i$ and $x_b \in c_j$ are related by $C_{=}(x_a, x_b)$, then $C_{=}(x_c, x_d)$ is verified for any $x_c \in c_i$ and $x_d \in c_j$ [11].

CL constraints do not constitute an equivalence relation. However, analogously, given $x_a \in c_i$ and $x_b \in c_j$, and the constraint $C_{\neq}(x_a, x_b)$, then it is also true that $C_{\neq}(x_c, x_d)$ for any $x_c \in c_i$ and $x_d \in c_j$ [11].

2.2 The Feasibility Problem

A relevant aspect to consider is to what extent constraints affect the complexity of the classic clustering problem. The feasibility problem for non-hierarchical instance-level constrained clustering can be formulated as in Definition 1 [19].

Definition 1 ***Feasibility Problem****: given a dataset X, a constraint set CS, and the bounds on the number of clusters $k_l \leq k \leq k_u$, does there exist a partition C of X with k clusters such that all constraints in CS are satisfied? [19]*

When $k_l = 1$ and $k_u \geq 3$, the feasibility problem for constrained clustering is **NP**-complete, which can be proven by reducing it from the Graph K-Colorability problem (it can also be proven that it is not harder, and thus both have the same complexity) [19]. Table 1 shows the complexity of the feasibility for different types of constraints.

Constraints	Complexity
Must-Link	**P**
Cannot-Link	**NP**-complete
ML and CL	**NP**-complete

Table 1: Feasibility problem complexity [19].

These complexity results show that the feasibility problem with CL constraints is intractable and hence constrained clustering is intractable too. For more details on the complexity of constrained clustering see [19].

Solving intractable problems with exacts methods is not feasible. Furthermore, many partition-related features can be optimized to obtain a good partition of a given dataset [27]. This provides an explanation for why multiobjective evolutionary algorithms constitute a good approach to the constrained clustering problem.

2.3 Multiobjective Optimization

The Multiobjective Optimization Problem (MOP) is formalized as in Equation 1:

$$\begin{aligned} &\text{minimize} \;\; F(y) = (f_1(y), \cdots, f_m(y)) \\ &\textbf{s.t.} \;\; y \in \Omega \end{aligned}, \tag{1}$$

where Ω is the variable space and $F : \Omega \rightarrow R^m$ consists of m real-valued functions (objective functions). R^m is known as the objective space and $\{F(y)|y \in \Omega\}$ defines the attainable object set. If $y \in R^n$ and Ω is defined as in Equation 2, with h_j being continuous functions, then the MOP in Equation 1 is said to be continuous.

$$\Omega = \{y \in R^n | h_j(y) \leq 0, j = 1, \cdots, m\}. \tag{2}$$

MOP techniques aim to balance all objective functions in Equation 1; this task is not trivial in the general case due to conflicts between the objective functions. A MOP technique finds a trade-off which can be defined in terms of Pareto optimality. Let $v, w \in R^m$, then, v dominates w if and only if $f_i(v) \leq f_i(w) \forall i \in \{1, \cdots, m\}$ and if $\exists j | f_j(v) < f_j(w), j \in \{1, \cdots, m\}$. This is: v dominates w if and only if v is better than w in at least one objective and as good as w in the rest, and is denoted as $w \prec v$.

A point $y^* \in \Omega$ is said to be Pareto optimal if there is no other point $y \in \Omega$ such that y dominates y^*. The set of Pareto optimal points is referred to as the Pareto Set (PS). The Pareto Front (PF) is formed by the objective vectors associated with the points in PS. A MOP technique aims to find the best possible approximation to the PF for any given optimization problem. A MOP definition for maximization problems can be obtained by reversing all inequalities.

In real-life applications of multiobjective optimization, a PF needs to be obtained so that a decision maker can later select the preferred solution. Being the MOP defined as above, there are no restrictions in the size of the PF, so in theory, for some problems, very large or even infinite Pareto optimal vectors could be found. This is why obtaining the full PF is usually not feasible, or at least very time-consuming. Moreover, if the Pareto approximation to the PF is too large the decision maker would have trouble choosing a solution from it due to the excess of information. Most multiobjective optimization methods struggle to find a set of well-distributed Pareto optimal vectors with a reasonable size that constitutes a good approximation to the entire PF. Evolutionary algorithms have proven to be excellent at finding this approximation to the PF [49], resulting in a whole new family of algorithms called Multiobjective Evolutionary Algorithms (MOEA) [50].

3 Memetic Elitist MOEA/D

In this section, we describe the general optimization scheme of Memetic Elitist MultiObjective Evolutionary Algorithm Based on Decomposition (ME-MOEA/D), which is based on

the classic MOEA/D method presented in [30]. ME-MOEA/D uses a biased selection operator to introduce elitism into the exploration process. Our proposal also takes advantage of the decomposition approach of MOEA/D to MOP to apply single-objective optimization methods.

ME-MOEA/D maintains a population P of $|P|$ individuals and a weight vector λ for each one of them. Each individual in P is referred to as $p_i = \{p_i^1, \cdots, p_i^n\}$, and its associated weight vector as $\lambda_i = \{\lambda_i^1, \cdots, \lambda_i^m\}$. The set of weight vectors is referred to as $\Lambda = \{\lambda_i, \cdots, \lambda_{|P|}\}$. Every λ_i is composed of m values such that $\lambda_i^j \geq 0 | j \in \{1, \cdots, m\}$ and $\sum_{j=1}^{m} \lambda_i^j = 1$. By doing this, the m-dimensional λ-space is defined. ME-MOEA/D also uses an external archive EP (External Population) to store the set of non-dominated solutions found so far (the approximation to the PF).

Multiobjective Problem Decomposition ME-MOEA/D uses the set of weight vectors Λ to introduce the decomposition factor into the optimization process as it is done in the classic MOEA/D. ME-MOEA/D decomposes the problem by approximating the PF as separated scalar problems. Each individual $p_i \in P$ is associated with a particular scalar problem. Three decomposition schemes were originally proposed in [30] for minimization problems (all of them can be reformulated for maximization problems). Selecting an appropriate decomposition scheme is of first importance in order to obtain high quality results for the problem to which we are applying ME-MOEA/D to. This is an utterly problem-dependent decision and needs to be carefully addressed. Here we introduce the Tchebycheff decomposition scheme. However, other approaches can be found in [51, 52, 49, 30].

The *Tchebycheff approach* [49] is shown in Equation 3, where $z^* = (z_1^*, \cdots, z_m^*)$ is the reference point, defined as $z_j^* = \min\{f_j(p_i) | j \in 1, \cdots, m\}$ with $p_i \in \Omega$ for minimization problems. For each Pareto Optimal point y^*, there is a weight vector λ such that y^* is optimal for Equation 3 and each optimal solution of Equation 3 is optimal for Equation 1. This way, the weight vectors $\{\lambda_1, \cdots, \lambda_{|P|}\}$ can be modified to obtain different PFs.

$$\begin{array}{l} \text{minimize } \; g^{te}(p_i|\lambda_i, z^*) = max\{\lambda_i^j | f_j(p_i) - z_j|\} \\ \textbf{s.t.} \;\; p_i \in \Omega \end{array} , \tag{3}$$

A key concept in ME-MOEA/D is the neighborhood in the λ-space. Without loss of generalization, let us assume that the optimal solution of $g^{te}(p_i|\lambda_i, z^*)$ should be close to the one for $g^{te}(p_j|\lambda_j, z^*)$ if λ_i and λ_j are close in the λ-space. Then, any information about g^{te}s with weight vectors close to λ_i can be used to improve $g^{te}(p_i|\lambda_i, z^*)$. The concept of neighborhood must be defined in order to do so: the neighborhood of λ_i is composed by its δ closest weight vectors in Λ, with distances between weight vectors calculated using the Euclidean distance.

Genetic Operators Given that ME-MOEA/D is still a genetic algorithm, it relies on the crossover, mutation and selection operators to explore the solution space. For each individual p_i in the population P, the crossover operator is used to generate a new individual

combining characteristics from two already explored individuals. Therefore, the mutation operator randomly modifies this new individual to introduce diversity in the population P. These problem-dependent procedures need to be defined bearing in mind the particularities of the problem to solve.

For each individual p_i two individuals p_a and p_b need to be selected from P for combination by the crossover operator. For this purpose, ME-MOEA/D uses a biased selection operator, which always randomly chooses the first individual p_a from the λ-neighborhood of p_i. A second individual p_b is (also randomly) chosen from the same λ-neighborhood or from the external population (EP), which is the set of non-dominated solutions, with a certain probability given by a parameter $\gamma \in (0, 1)$. This parameter γ is meant to control the exploration-exploitation trade-off of ME-MOEA/D. When $\gamma = 0$, the unbiased (regular) selection operator is used, so p_b is always chosen from P, whereas if $\gamma = 1$, p_b is always chosen from EP (the elitist part of this biased selection operator is applied). Figure 1 shows a graphical representation of the biased selection operator.

Memetic Elitism ME-MOEA/D introduces a bias towards high quality individuals by means of an LS procedure. In each generation, the elite of the population must be obtained. In order to do so, the *dominance index* ψ_i of each individual p_i has to be computed as in Equation 4.

$$\psi_i = |\{p_j | p_i \prec p_j \wedge p_j \in P\}| \tag{4}$$

For every individual p_i, its dominance index ψ_i is computed as the number of individuals in the population P dominating it. The set of ν individuals featuring the lowest dominance index are selected as the elite of P. The elite of the population is improved by applying an LS procedure to each individual in it. Since LS procedures are typically single-objective optimization procedures, we cannot apply it to optimize all m objective functions at the same time. Here is where we again use the λ-space in our favor: we can decide which objective function to optimize for every p_i depending on its associated λ_i. There are multiple options:

- Optimize the least relevant target function: we can determine what would be the least significant objective function when computing $g^{te}(p_i|\lambda_i, z^*)$, which would be $f_\alpha|\alpha = \text{argmin}_{i=1}^{m}\{\lambda_i^1, \cdots, \lambda_i^m\}$. Applying the LS procedure to the least relevant function for every p_i effectively makes the evolutionary part of ME-MOEA/D responsible for the optimization of the more relevant target functions for every p_i.
- Optimize the most relevant target function: similarly, the most relevant target function for every p_i can be determined as $f_\alpha|\alpha = \text{argmax}_{i=1}^{m}\{\lambda_i^1, \cdots, \lambda_i^m\}$. Applying the LS procedure to the most relevant function reinforces the evolutionary algorithm in the optimization of the same target functions, which effectively intensifies the exploitation stages.
- Optimize any other target function: any decision maker can be used to determine which target function to optimize when a λ-space is available. Adaptive methods

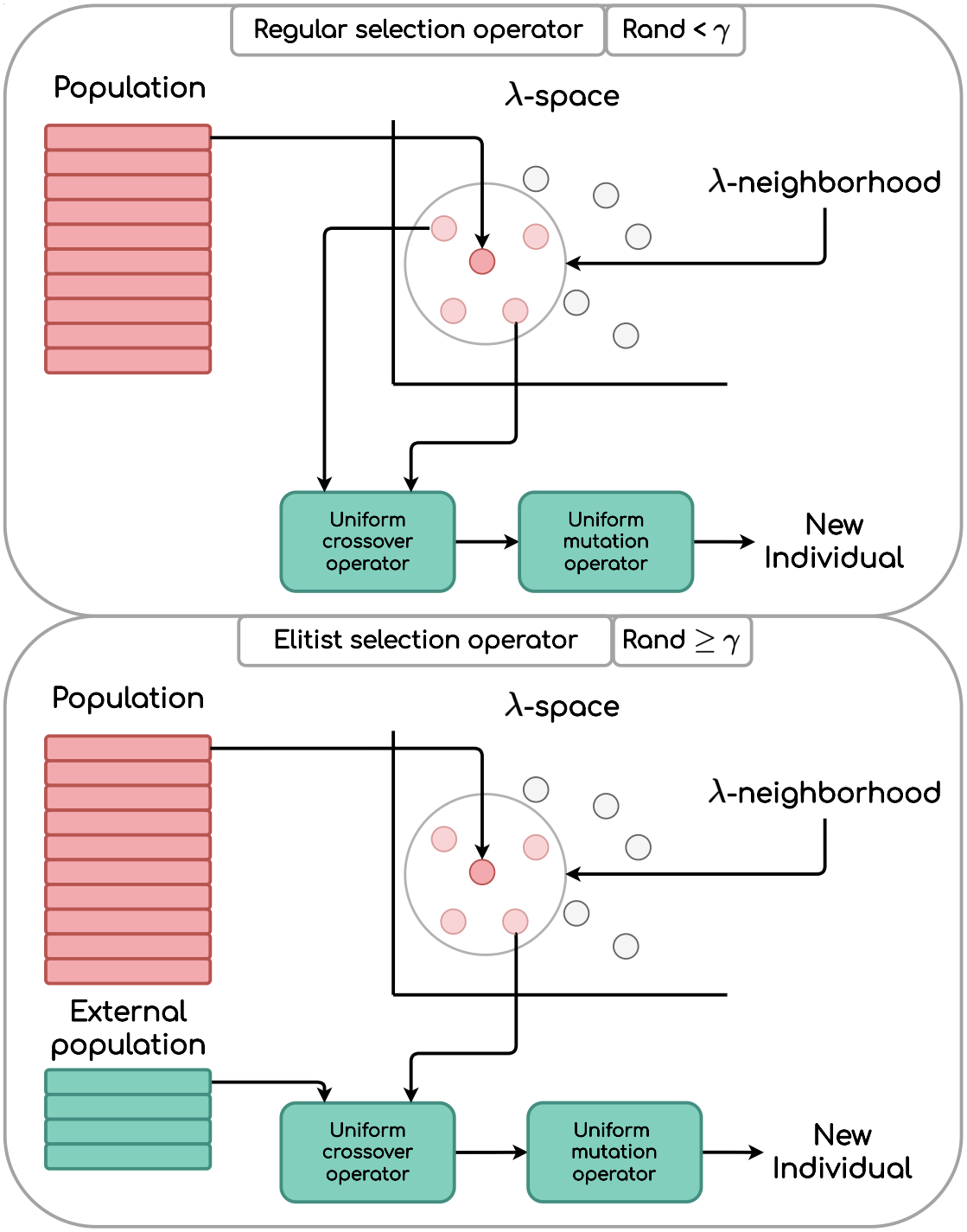


Figure 1: Visual representation for the regular and elitist behaviors of the biased selection operator.

could even be used to better balance the exploration-exploitation trade-off of ME-MOEA/D. This is an exceedingly problem-dependent aspect of our proposal and needs to be carefully considered before applying ME-MOEA/D to real-world problems.

The LS procedure is a problem-dependent algorithm, although a condition must be fulfilled: given an initial individual p_i to apply it to, a local change is made to get p'_i and this individual replaces p_i if and only if $p_i \prec p'_i$. By doing so, we make sure that true improvement is always achieved and that the original solution p_i never degenerates into worse solutions, so that its dominance index ψ_i is never higher after applying the LS. As we have already mentioned, ME-MOEA/D maintains a set of non-dominated solutions called EP. Results obtained by the LS procedure are transferred only to EP and not to the population itself, as this would strongly bias the exploration of the solutions space. Note that the LS procedure indirectly helps the population to converge to high quality solutions, as the EP population takes part in our proposed biased selection operator.

In each generation, the current population is composed of the best solution found so far for each individual p_i (subproblem), and an external population (EP) is maintained to store all non-dominated solutions found. It is also necessary to keep track of the target function values for each individual. To this end, ME-MOEA/D uses the set $\{FV_1, \cdots, FV_{|P|}\}$, where FV_i is $F(p_i)$. Algorithm 1 summarizes the overall ME-MOEA/D optimization process and Figure 2 shows a pictorial representation of it.

Initialization of Λ, z and P are problem-specific procedures. A proper initialization step for P is crucial to the evolutionary process. Also note that the size of the external population EP can be limited, preventing it from growing indefinitely. This is a problem-dependent procedure too, and the influence of EP in our proposed biased selection operator has to be taken into account when designing it.

4 Constrained Clustering Through ME-MOEA/D

In this section, a description of all problem-dependent elements that need to be defined to apply ME-MOEA/D to the constrained clustering problem is provided. The combination of these elements has never been applied to the CC problem before. ME-MOEA/D is able to produce high quality results when provided with this components to solve the CC problem. The particularization of ME-MOEA/D for CC is referred as ME-MOEA/D$_{CC}$.

Label-based Representation Scheme Metaheuristic methods have already been successfully applied to automatic clustering. Several representations schemes have been proposed: binary encoding, label-based encoding, graph-based encoding, fixed-length real encoding or variable-length real encoding, among others [26, 53]. All of them have advantages in specific application scenarios. We have selected label-based encoding, as it allows a specific control of the number of clusters in a partition.

In label-based encoding, each individual p_i from P defines a partition of the dataset X by explicitly assigning a label to each one of its instances. With this, we have that $p_i = [p_i^1, \cdots, p_i^n]$

Algorithm 1: Memetic Elitist MOEA/D

Input: Size of the population $|P|$, maximum size of the external population |EP|, size of the λ-neighborhoods δ, selection operator bias probability γ, size of the elite of the population ν, set of weight vectors Λ.

`// Initialization Step`

[1] $EP = \emptyset$

[2] Initialize Λ and $z = [z_i, \cdots, z_m]$

[3] Obtain the λ-neighborhood for every $\lambda_i \in \Lambda$ as $\{\lambda_i^1, \cdots, \lambda_{|P|}^{\delta}\}$ by computing the Euclidean distances in Λ. Then, for every $i \in \{i, \cdots, |P|\}$ set $\Delta(i) = \{i_1, \cdots i_\delta\}$

[4] Generate the initial population $P = \{p_1, \cdots, p_{|P|}\}$ and get their fitness values as $FV_i = F(p_i) \forall i \in \{i, \cdots, |P|\}$

[5] **while** *stopping criteria are not met* **do**

[6] | **for** $i \in \{i, \cdots, |P|\}$ **do**

| | `// Selection Operator`

[7] | | **if** `Rand(0,1)` $< \gamma$ **then**

[8] | | | Select p_a randomly from the p_i λ-neighborhood and select p_b randomly from EP

[9] | | **else**

[10] | | | Select p_a and p_b randomly from the p_i λ-neighborhood.

[11] | | **end**

| | `// Crossover Operator`

[12] | | Obtain a new individual p_{new} by applying the crossover operator to p_a and p_b.

| | `// Mutation Operator`

[13] | | Mutate p_{new} by applying the mutation operator

| | `// Update reference point` z

[14] | | For each $j \in \{1, \cdots, m\}$ set $z_j = f_j(p_{\text{new}})$ if $f_j(p_{\text{new}}) < z_j$

| | `// Update the neighborhood of` p_i

[15] | | For each $j \in \Delta(i)$ set $p_j = p_{\text{new}}$ and $FV_j = F(p_{\text{new}})$ if $g^{te}(p_{\text{new}}|\lambda_j, z) \leq g^{te}(p_j|\lambda_j, z)$

| | `// Update the external population EP`

[16] | | Remove from EP all vectors dominated by $F(p_{\text{new}})$

[17] | | Add $F(p_{\text{new}})$ to EP if it is not dominated by any vector in EP

[18] | | Remove elements from EP if its maximum size is exceeded

[19] | **end**

| `// Memetic Elitism`

[20] | Obtain the indices *EliteIndices* of the best ν individuals in P by computing the dominance index ψ_i for every individual

[21] | **for** $j \in$ *EliteIndices* **do**

[22] | | Apply LS to p_j having into account λ_j to get an improved individual p_{LS}

[23] | | Remove from EP all vectors dominated by $F(p_{LS})$

[24] | | Add $F(p_{LS})$ to EP if it is not dominated by any vector in EP

[25] | **end**

[26] **end**

[27] **return** EP

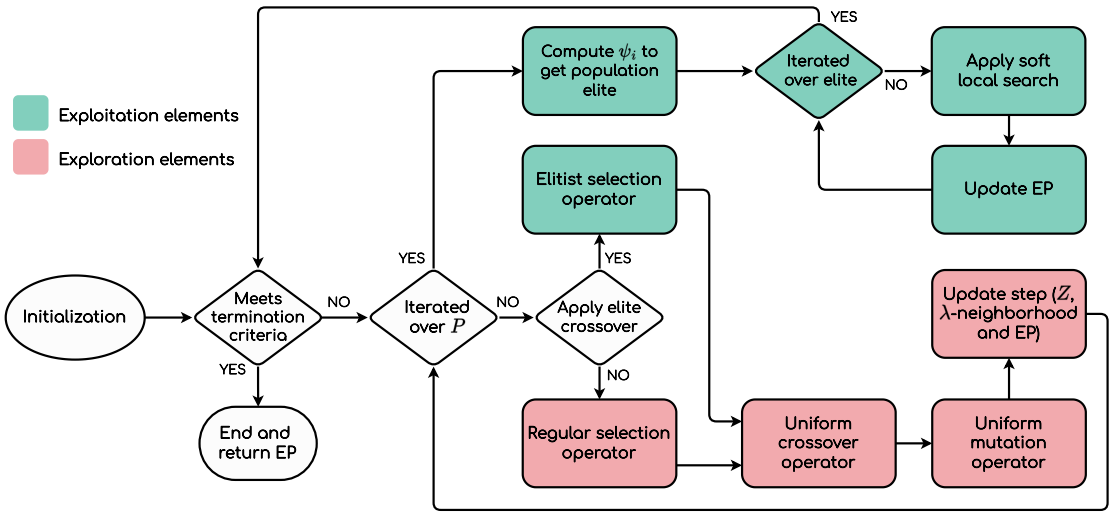


Figure 2: Diagram summarizing ME-MOEA/D optimization process.

where $p_i^j = l | l \in \{1, \cdots, k\} \forall j \in \{i, \cdots, n\}$. This means that every position p_i^j of p_i contains the label of the jth instance of the dataset X. Figure 3 shows an example of the label-based representation scheme.

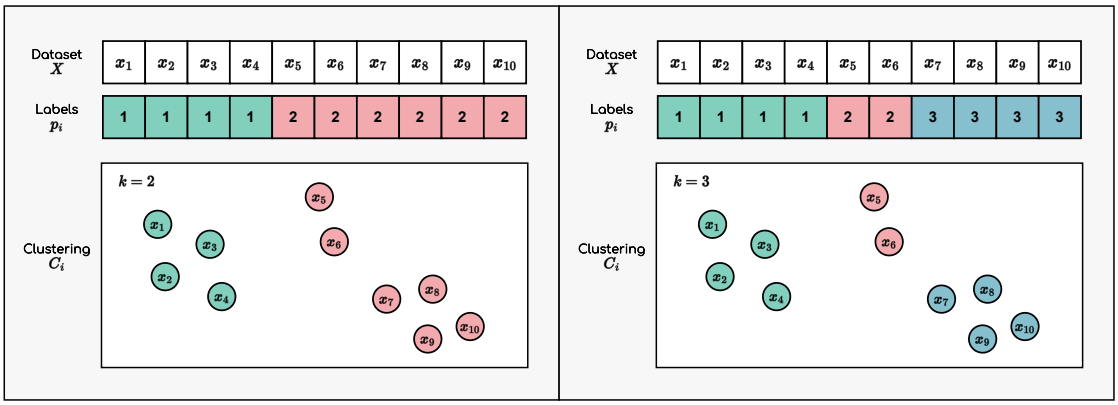


Figure 3: Label-based representation scheme.

Target Functions for the CC Problem The following three functions ($m = 3$) are used to find high quality solutions to the CC problem. The ratio of the within-cluster mean distance to the between-cluster separation is known as the *Davies-Bouldin* function [54]. It is used to keep clusters compact and separated from each other. Equation 5 defines the within-cluster mean distance for a cluster c_i.

$$\overline{c_i} = \frac{1}{|c_i|} \sum_{x_j \in c_i} \|x_j - \mu_i\|^2, \tag{5}$$

where μ_i is the centroid of cluster c_i. The distance between two clusters c_i and c_j is computed as the Euclidean distance between their centroids: $d_{i,j} = \|\mu_i - \mu_j\|^2$. Then, $R_i = \max_{j,j\neq i}\{(\overline{c_i} + \overline{c_j})/d_{i,j}\}$, so that the Davies-Bouldin function can be defined as in Equation 6 for a partition C and a dataset X. Quality results are obtained by minimizing DB(C, X).

$$\text{DB}(C, X) = \frac{1}{k}\sum_{i=1}^{k} R_i. \tag{6}$$

Secondly, the *Connectedness* function is utilized to keep the number of clusters of the generated solutions under control [55]. Every instance in the dataset is related to the number of neighboring instances placed in different clusters. Given a partition C and a dataset X, the *Connectedness* is computed as in Equation 7.

$$\text{Conn}(C, X) = \sum_{i=1}^{n}\left(\sum_{j=1}^{\epsilon} x_{i,nn_i(j)}\right), \tag{7}$$

where $nn_i(j)$ is the jth neighbor of instance x_i, ϵ defines the size of the neighborhood for every instance and $x_{i,nn_i(j)}$ is computed as in Equation 8. A good partition would minimize Conn(C, X).

$$x_{i,nn_i(j)} = \begin{cases} \frac{1}{j} & \nexists c_k | x_i, nn_i(j) \in c_k \\ 0 & \text{otherwise} \end{cases}. \tag{8}$$

Lastly, the *Infeasibility* function integrates constraints into the clustering process. It measures the number of violated constraints in a given partition C for a set of ML constraints $C_=$ and a set of CL constraints $C_\neq$, as in Equation 9.

$$\text{Infs}(C, C_=, C_\neq) = \sum_{i=0}^{n}\sum_{j=0}^{n} V(C_=(x_i, x_j)) + V(C_\neq(x_i, x_j)), \tag{9}$$

where $V(\cdot)$ is a function that returns 1 if the constraint given as argument exists and is violated and 0 otherwise. Clearly Infs$(C, C_=, C_\neq)$ is a function to minimize.

All three target functions described above need to be minimized, and thus, the CC problem is approached from the point of view of minimization. The lower bound for all functions is 0, so that the reference point is set as $z^* = (0, 0, 0)$. ME-MOEA/D$_{CC}$ uses the Tchebycheff decomposition scheme to optimize these functions.

Local Search Procedure for the CC Problem ME-MOEA/D uses an LS procedure to locally improve individuals p_i from P in a non-exhaustive way. The LS procedure proposed for the CC problem randomly chooses an index j from an individual p_i (instance j from the dataset X) and assigns it to different clusters iteratively. The target function to optimize for

each individual is determined by the most relevant target function, which is referred in this study as f_{alpha}, as we aim to reinforce the exploitation part of the evolutionary algorithm.

Every time an instance is moved from one cluster to another in p_i, a new individual p_i' is generated. This individual substitutes p_i if $f_\alpha(p_i') < f_\alpha(p_i)$ and $p_i \prec p_i'$. When there is no possible improvement for a particular index j from p_i, the LS is said to fail and the fails counter is increased. If the LS reaches the maximum number of fails, it stops. This maximum number of fails is given in the form of a proportion $\xi \in (0, 1)$ of the number of instances in the dataset X (equivalent to the length of each individual p_i). Parameter ξ allows for an effective control of the exploration-exploitation trade-off and helps the evolution process to converge. If ξ is properly set then a lot of target function evaluations will be spent in the LS in the early stages of the algorithm, as the individuals in the population are more likely to be locally improved. However, in late stages of the evolution process individuals in the population are arduous to locally improve. The maximum number of fails will prevent the LS from wasting target function evaluations that could be employed by the exploration components of the algorithm. Algorithm 2 summarizes the LS procedure described above.

Algorithm 2: Local Search

Input: Individual to be locally improved p_i, weights vector λ_i, fail percent ξ, number of clusters k.

```
// Find the most significant cost function index
```
[1] $\alpha \leftarrow \text{argmax}_{i=1}^{m}\{\lambda_i^1, \cdots, \lambda_i^m\}$
[2] `fails` $\leftarrow 0$
[3] **while** *improvement* ***or*** $fails < n \times \xi$ **do**
[4] `improvement` $\leftarrow$ `false`
[5] $j \leftarrow$ `RandInt`$(\{1, \cdots, n\})$
`// Random shuffle labels set`
[6] $RSL \leftarrow$ `RandomShuffle`$(\{1, \cdots, K\})$
[7] **for** $l \in RSL$ ***and while not*** *improvement* **do**
[8] $p_i' \leftarrow p_i$
`// Move instance` j `to the cluster associated with label` l
[9] $\left[p_i^j\right]' \leftarrow l$
[10] **if** $f_\alpha(p_i') < f_\alpha(p_i)$ ***and*** $p_i \prec p_i'$ **then**
[11] $p_i \leftarrow p_i'$
[12] `improvement` $\leftarrow$ `true`
[13] **end**
[14] **end**
[15] **if** ***not*** *improvement* **then**
[16] `fails` $\leftarrow$`fails+1`
[17] **end**
[18] **end**
[19] **return** p_i

EP Limiting Method The size of the external archive storing non-dominated vectors needs to be kept from growing indefinitely. Adaptive archiving methods can be used to do so [56]. Such methods involve removing non-dominated solutions from EP, and have the potential to worsen the PF approximation found by ME-MEOA/D. Nonetheless, we can take advantage of one of the characteristics of the CC problem: it is a multimodal problem. This means that two different individuals p_i and p_j, distant from each other in the solutions space, can have very similar values for all objective functions $F(p_i) \approx F(p_j)$ and thus their projection in the objective space would be very close to each other. These two individuals p_i and p_j are unlikely to dominate each other $F(p_i) \approx F(p_j) \nrightarrow p_i \prec p_j \wedge p_j \prec p_i$, in which case both need to be kept in the external population, even if they encode very similar partitions. Figure 4 shows an example of this.

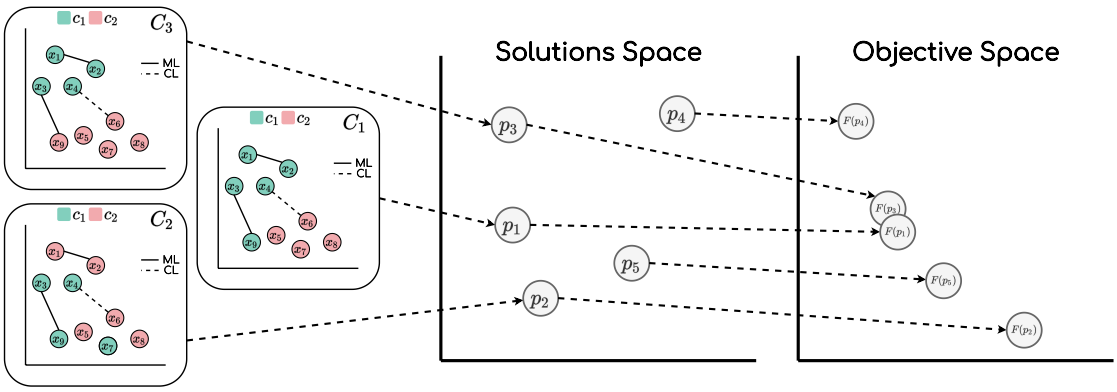


Figure 4: Multimodality in the CC problem.

In Figure 4, partitions C_1 and C_3 are very similar, they only differ on the label of instance x_9, which belongs to cluster c_1 (in green) in partition C_1, and to cluster c_2 (in red) in partition C_3. Individuals p_1 and p_3 encode partitions C_1 and C_3 respectively. On the one hand, note how clusters in C_3 are more compact than clusters in C_1, thus $\text{DB}(p_3) < \text{DB}(p_1)$. On the other hand, achieving more compact clusters C_3 violates a constraint, whereas C_3 does not, and thus $\text{Infs}(p_3, \text{ML}, \text{CL}) > \text{Infs}(p_1, \text{ML}, \text{CL})$. This is a clear example of the effects of multimodality: p_1 and p_3 are different partitions of X and thus they are distant in the solutions space, whereas their values for all objective functions are very similar $F(p_1) \approx F(p_3)$ but they do not dominate each other $p_1 \nprec p_3 \wedge p_3 \nprec p_1$, as p_1 features a better value than p_3 for the Infs function and vice versa for the DB function.

To reduce the effect of the multimodality of the CC problem, a density-based storing method for non-dominated solutions can be used: when a new solution needs to be stored and the maximum size |EP| for the EP has been reached, a solution from the most crowded region in the objective space is removed [57]. In order to do so, an adaptive m-dimensional mesh is extended over the objective space. This effectively divides the objective space into m-dimensional cells, with each one of them enclosing a region of the objective space. Every dimension of the objective space is divided into φ equal intervals, so the total number of cells is given by φ^m. The size of the interval dividing every dimension of the objective space is

computed for every $f_a|a \in \{1, \cdots, m\}$ as $(\max_{i=1}^{n}\{f_a(p_i)\} - \min_{i=1}^{n}\{f_a(p_i)\})/\varphi$. The crowding of every cell is computed as the number of solutions enclosed by it in the objective space. Figure 5 shows an example of this with $\varphi = 4$. Once the crowding for every cell has been computed, a random cell is selected among those with the highest crowding, and a random individual belonging to this cell is removed in order to make room for a new solution entering EP.

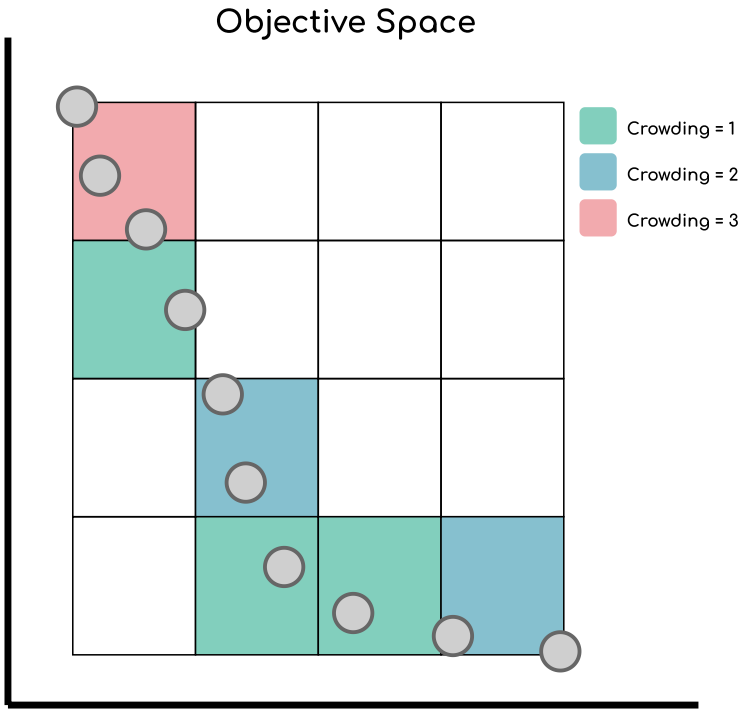


Figure 5: Adaptive mesh over an objective space with $\varphi = 4$.

This EP archiving strategy prevents cell overcrowding and thus lessens the effects of multimodality in the CC problem. Additionally, let us remember that the EP does not only stores non-dominated solutions, but also takes part in the evolutionary process of ME-MOEA/D by means of the biased selection operator. Once the elitist part of the biased selection operator is applied, a solution is randomly chosen from EP to be fed into the crossover operator. By using this EP archiving strategy, a better-distributed approximation to the PF is achieved, thus reducing the probability of the same solution being randomly selected multiple times from EP for the biased selection operator. As a result a better exploration capability for ME-MOEA/D_{CC} is achieved.

Biased Crossover Operator and Uniform Mutation Operator The crossover operator is always given two individuals (parents) p_a and p_b to generate a new individual (offspring) p_{new} combining their features. The uniform crossover operator randomly selects a given percentage (typically 50%) of features from the first parent p_a and copies them into the offspring p_{new} without change, and then fills the rest with features from the second parent p_b.

However, if any difference can be established between p_a and p_b, the percentage of features inherited from each parent can be tuned to affect the convergence of the evolutionary process. This is the case of ME-MOEA/D$_{CC}$.

With respect to the CC problem, the new individual p_{new} always inherits a higher percentage ζ of features from p_b than from p_a, and therefore p_b is considered to be the elite parent. Now, there is the issue of deciding which one in a pair of individuals will be the elite parent p_b. Once the elitist half of the biased selection operator is applied, the decision is easy: the elite parent is always the individual randomly selected from EP. When the regular half of the operator is used, this decision becomes a bit more complex. Given the set of weight vectors Λ, a label l_i can be assigned to every λ_i such that $l_i = \text{argmax}\{\lambda_i^1, \cdots, \lambda_i^m\}$, effectively splitting the λ-space into m clusters. The elite parent p_b will be the individual whose associated weight vector λ_b is closer (Euclidean distance) to the centroid of its associated cluster label l_b in the λ-space. Cluster membership and centroids in the λ-space have to be computed only once, since the weight vectors are never changed in ME-MOEA/D.

Once the new individual p_{new} has been generated, it is fed into the classic uniform mutation operator, which randomly changes membership of instances to clusters to introduce diversity. The mutation probability for each instance (gene) is controlled by the parameter ϖ.

Population and Λ Initialization Procedure Initializing the population P with a good mixture of random solutions and quality solutions is of paramount importance in order for the evolutionary process of the population to converge and ultimately find better solutions. To achieve this, a K-means based initialization method is used for the CC problem [37]. A parameter ϱ specifies the percentage of the population to be initialized with a random number in $[10, 20]$ of iterations of the K-means algorithm. This lets us generate different solutions and favor diversity. The rest of the population is randomly initialized only by filling the individual (array) p_i with random integers in the interval $[1, k]$. We have found this to produce more advantageous results than a full random initialization. Regarding Λ initialization, all weight vectors λ_i are randomly initialized following a normal distribution. Advantages of random initialization for the λ-space are discussed in [58].

5 Experimental Setup

Table 2 displays a summary of the datasets used in our experiments. Results obtained by ME-MOEA/D$_{CC}$ and the state-of-the-art are compared over 20 real-world datasets and 3 constraint sets for each one of them. These datasets can be found at the Keel-dataset repository[1] [59], and at the `scikit-learn` Python package[2] [60]. Datasets marked with $*$ have been undersampled to 30%, as they have already been solved in the literature and are used here just as benchmarks.

[1] https://sci2s.ugr.es/keel/category.php?cat=clas
[2] https://scikit-learn.org/stable/datasets/index.html

Name	No. Instances	No. Classes	No. Features
Appendicitis	106	2	7
Banana*	1590	2	2
Breast Cancer	569	2	30
Bupa	345	16	5
Contraceptive	1473	3	9
Heart	270	2	13
Ionosphere	351	2	33
Iris	150	3	4
Monk2	432	2	6
Newthyroid	215	3	5
Phoneme*	1622	2	5
Pima	768	2	8
Saheart	462	2	9
Soybean	47	4	35
Spectfheart	267	2	44
Tae	151	3	5
Titanic*	661	2	3
Vowel	990	11	13
Wdbc	569	2	30
Wine	178	3	13

Table 2: Summary of datasets used for the experiments.

Classification datasets allow for an easy building of the constraint set. Labels are used as an oracle which is queried with two instances. If they belong to the same class, an ML constraint is set between the two instances; a CL constraint is set otherwise (see Section 5.1). Classification datasets also provide an easy evaluation of constrained clustering methods, as the true labels can be used to assess the quality of the results (see Section 5.2). All datasets used in our experiments are commonly used as benchmarks, covering a wide range of values for all three features displayed in Table 2.

5.1 Constraint Generation

Three constraint sets with incremental levels of constraint-based information are generated for every dataset in Table 2 following the method proposed in [9]. This method builds the constraint set by randomly selecting two instances from the dataset and setting a ML or CL constraint depending on the labels of the instances.

The three constraint sets CS_{10}, CS_{15} and CS_{20} are associated with a small percentage of the size of the dataset: 10%, 15% and 20%, respectively. To get the number of constraints generated for every constraint set associated with every dataset, the formula $\frac{n_f(n_f-1)}{2}$ is used, which is the number of edges in a complete graph with n_f vertices.

Dataset	CS_{10}		CS_{15}		CS_{20}	
	ML	CL	ML	CL	ML	CL
Appendicitis	37	18	76	44	154	77
Banana*	6368	6193	14311	14130	25509	24894
Breast Cancer	846	750	1954	1701	3292	3149
Bupa	91	504	217	1109	374	1972
Contraceptive	3893	6985	8620	15690	15372	27993
Heart	173	178	436	384	747	684
Ionosphere	338	292	738	640	1357	1128
Iris	28	77	92	161	132	303
Monk2	484	462	1064	1016	1835	1906
Newthyroid	125	106	273	255	488	415
Phoneme*	7817	5386	17270	12376	30826	21824
Pima	1572	1354	3601	3069	6443	5338
Saheart	613	468	1281	1134	2368	1910
Soybean	1	9	11	17	5	40
Spectfheart	230	121	563	257	952	479
Tae	31	89	88	165	164	301
Titanic*	1249	962	2721	2229	4908	3870
Vowel	445	4406	1020	10006	1786	17717
Wdbc	864	732	1975	1680	3448	2993
Wine	57	96	105	246	210	420

Table 3: Number of constraints used in experiments.

By randomly allocating the constraints as described above, the bias introduced by using subsets of labeled data can be avoided, as the risk of biasing the constraint set towards classes with poor representation is lower. The number of constraints of each type obtained for every dataset and constraint set is displayed in Table 3. All these constraint sets can be found in this link [3].

5.2 Evaluation Method

In this study, a series of MOEAs and classic CC algorithms are compared with our proposal, ME-MOEA/D$_{CC}$. All methods can be compared from the point of view of CC, as ultimately they all output a partition for a given dataset. Concerning the comparison of multiple MOEAs, the approximation to the PF produced by each one of them can be compared by using Pareto-specific measures.

[3] https://drive.google.com/drive/u/1/folders/1sjnPYitey8q9zrPKa_YpFS7iNKTT15QH

Comparing Partitions When it comes to evaluating the quality of the partition obtained by any given constrained clustering method, the Adjusted Rand Index (ARI) can be used [61]. As classification datasets are used, we have the true labels at our disposal. By using the ARI measure, the degree of similarity of two partitions for the same dataset can be computed; therefore, if one of these partitions corresponds to the true labels, the degree of similarity with the ground truth can be obtained.

ARI is the corrected-for-chance version of the basic Rand Index, which computes the degree of agreement between two partitions C_1 and C_2 of a given dataset X. C_1 and C_2 can be viewed as collections of $n(n-1)/2$ pairwise decisions [62]. For each x_i and x_j in X, they are assigned to the same cluster or to different clusters by a partition. The number of pairings where x_i is in the same cluster as x_j in both C_1 and C_2 is taken as a; conversely, b represents the number of pairings where x_i and x_j are in different clusters. The degree of similarity between C_1 and C_2 is computed as in Equation 10.

$$\text{Rand}(C_1, C_2) = \frac{a+b}{n(n-1)/2} \tag{10}$$

Equation 10 can be corrected for chance by taking into account the expected similarity of all comparisons between partitions specified by the random model to establish a baseline. Equation (11) shows the expression used to compute ARI.

$$\text{ARI}(C_1, C_2) = \frac{\text{Rand}(C_1, C_2) - \text{Expected Index}}{\text{Maximum Index} - \text{Expected Index}}, \tag{11}$$

where Expected Index is the degree of similarity with a random model and Maximum Index is assumed to be 1. With this, it is clear that $\text{ARI}(C_1, C_2) \in [-1, 1]$, with a value close to 1 meaning a high degree of agreement between C_1 and C_2, positive values close to 0 meaning no agreement, and a value below 0 meaning that the $\text{Rand}(C_1, C_2)$ is less than expected when compared with the results of a random model. To summarize: the higher the ARI, the greater the degree of similarity between C_1 and C_2. For more details on ARI, see [61].

The ability of each method to integrate constraints into the clustering process needs to be evaluated as well. To this end, we define the Unsat measure, which, given a partition C and the constraint sets $C_=$ and $C_{\neq}$, computes the percentage of unsatisfied constraints as in Equation 12.

$$\text{Unsat}(C, C_=, C_{\neq}) = \frac{\text{Infs}(C, C_=, C_{\neq})}{|C_=| + |C_{\neq}|}, \tag{12}$$

Comparing PF Approximations An in-depth analysis of MO evaluation methods can be found in [63], where authors insist on the importance of using more than one measure to evaluate the quality of the PF approximation found by a given method. In this study we use the hypervolume unary quality indicator $H(\cdot)$ to evaluate the performance of every

MOEA individually, and the additive ϵ-indicator (ϵ^+-indicator) $I_{\epsilon+}(\cdot,\cdot)$ to carry out paired comparisons.

The hypervolume measure was proposed in [64] and can be defined as in [65] by Definition 2 assuming minimization.

Definition 2 ***Hypervolume*** *[65]: Given a set A of points in the positive orthant $\mathbb{R}^d_{\geq 0}$, the hypervolume indicator $H(A)$ is defined as the d-dimensional volume of the hole-free orthogonal polytope*

$$\Pi^d = \left\{x \in \mathbb{R}^d_{\geq 0} | a \preceq x \wedge x \leq r \ \ \forall a \in A\right\}$$

This polytope corresponds to the space dominated by at least one point in A, and r is the reference point used to compute the volume of Π^d.

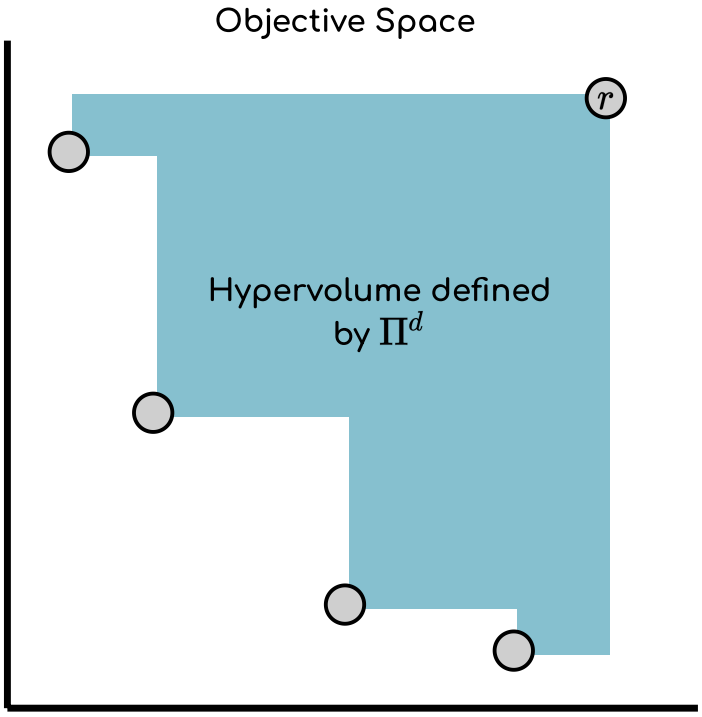


Figure 6: 2-dimensional polytope defining the hypervolume of a set of solutions in a Pareto approximation.

The ϵ^+-indicator was proposed in [64]. It is a binary indicator comparing two sets of non-dominated solutions and can be defined as in Definition 3.

Definition 3 ***ϵ^+-indicator*** *[64]: Without loss of generality, let us suppose a minimization problem with m positive objectives $Z \subseteq \mathbb{R}^{+^n}$. Given two objective vectors $z_1 = (z_1^1, z_1^2, \cdots, z_1^m) \in Z$ and $z_2 = (z_2^1, z_2^2, \cdots, z_2^m) \in Z$ then:*

$$z_1 \succeq_{\epsilon^+} z_2 \leftrightarrow \forall i | 1 \leq i \leq m | z_1^i \leq \epsilon + z_2^i$$

for a given $\epsilon > 0$. Then z_1 is said to ϵ^+-dominate z_2. Given two approximation sets A and B, the binary ϵ^+-indicator $I_{\epsilon^+}(\cdot,\cdot)$ is defined as:

$$I_{\epsilon^+}(A,B) = \inf_{\epsilon \in \mathbb{R}} \{\forall z_2 \in B \;\; \exists z_1 \in A | z_1 \succeq_{\epsilon^+} z_2\}$$

Intuitively, the ϵ^+-indicator shows the factor by which an approximation set is worse than another in all objectives. In other words: $I_{\epsilon^+}(A,B)$ gives the minimum factor ϵ such that any vector in B is ϵ^+-dominated by at least one vector in A. This can be computed in $\mathcal{O}(m \times |A| \times |B|)$ as in Equation 13.

$$I_{\epsilon^+}(A,B) = \max_{z_2 \in B} \min_{z_1 \in A} \max_{1 \leq i \leq m} (z_1^i - z_2^i) \tag{13}$$

Note that $I_{\epsilon^+}(\cdot,\cdot)$ is not a symmetric function, so $I_{\epsilon^+}(A,B) \neq I_{\epsilon^+}(B,A)$. In order to truly compare A and B, both $I_{\epsilon^+}(A,B)$ and $I_{\epsilon^+}(B,A)$ have to be computed. Having said that, if $I_{\epsilon^+}(A,B) > I_{\epsilon^+}(B,A)$ it means that the factor ϵ_1^+ by which A ϵ^+-dominates B is higher than the factor ϵ_2^+ by which B ϵ^+-dominates A, indicating an advantage of A over B, and $I_{\epsilon^+}(A,B) < I_{\epsilon^+}(B,A)$ indicates exactly the opposite.

5.3 Validation of results

An in-depth analysis of the disadvantages of the classic Null Hypothesis Statistical Tests (NHST) can be found in [66], where a new model is proposed to carry out statistical comparisons between two sets of results. In [66], the authors summarize the drawbacks of NHST as *"In a nutshell: NHST do not answer the question we ask"*. These drawbacks are based on the trap of black-and-white thinking, that is: to reject, or not to reject?

Bayesian tests can be used to overcome the drawbacks found in NHST. The Bayesian sign test is the Bayesian version of the NHST non-parametric sign test. It can be easily used to compare two sets of results by employing the `rNPBST` R package in [67]. A general use guide to Bayesian statistical testing can be found in [68].

The Bayesian sign test obtains the statistical Dirichlet distribution of a parameter ρ according to the differences between two sets of results. The number of cases where $A - B < 0$, the number of cases where no significant differences are found, and the number of cases where $A - B > 0$ are counted to get the distribution of ρ. The region of practical equivalence (rope) $[r_{\min}, r_{\max}]$ needs to be defined in order to identify cases where there are no significant differences, so that $P(A \approx B) = P(\rho \in \text{rope})$. Finally, the weights of the Dirichlet distribution are computed so it can be sampled to get triplets with the following form:

$$[P(\rho < r_{\min}) = P(A - B < 0), \;\; P(\rho \in \text{rope}), \;\; P(\rho > r_{\max}) = P(A - B > 0)]$$

5.4 Competing Methods

Our proposal, ME-MOEA/D$_{CC}$, is compared with other 8 CC algorithms, including the base method MOEA/D, the state-of-the-art in MOEA applied to the CC problem, a single-objective EA, and other 4 well-known non-evolutionary CC methods. These 8 methods can be briefly summarized as follows:

- **MOEA/D**: The MOEA/D method is the base algorithm used to build ME-MOEA/D. The same application scheme used for ME-MOEA/D (described in Section 4) can be used (excluding memetic elements) to apply MOEA/D to the CC problem to get MOEA/D$_{CC}$.
- **MOCK**: The Multiobjective Clustering with automatic K-determination (MOCK) algorithm is the application scheme of PESA-II [32] to the clustering problem, which is further extended to CC. It uses a graph-based representation scheme, along with a selection operator based on crowding [32] and a biased mutation operator to optimize compactness and connectedness. It was originally designed for the classic clustering problem, and extended to the CC problem by computing the ARI of a subset of labeled data with the labels that the algorithm finds for those same instances of the dataset. The K-means algorithm and a minimum spanning trees based method are used to initialize the population [37].
- **PESA-II**: Although the Pareto Envelope-based Selection Algorithm II (PESA-II) [32] is used by the MOCK application scheme as the base algorithm, the capabilities of this MO strategy need to be tested when provided with the same information as our proposal ME-MOEA/D$_{CC}$. In order to do so, the application scheme used for ME-MOEA/D (Section 4) can be used (excluding memetic elements) to apply PESA-II to the CC problem, thus obtaining PESA-II$_{CC}$.
- **SHADE**: In [69] the single-objective Success History based Adaptive Differential Evolution (SHADE) algorithm is applied to the CC problem, resulting in SHADE$_{CC}$. It uses the label-based representation scheme, as well as memetics elements such as an elitist selection operator and a local search procedure, to produce quality results for the CC problem.
- **COPKM**: COnstrained Partitional K-means (COPKM) modifies the assignment rule of instances to clusters of the classic K-means algorithm in a way such that an instance can be assigned to a cluster only if no constraints are violated by the assignment [9].
- **LCVQE**: The Linear Constrained Vector Quantization Error algorithm introduces a modification of the cost function of CVQE to make it less computationally complex [70].
- **TVClust**: Two Views Clustering applies clustering methods over the dataset and the constraint set separately, and try to find a consensus between the results [71]. It makes a soft interpretation of the constraints.

- **RDPM**: Relation Dirichlet Process - Means can be viewed as an extension of K-means that includes side information (constraints). It is a deterministic derivation of the TVClust model. The number of clusters (k) does not need to be specified [71].

5.5 Parameters Setting

The parameters used for every competing method are described in this section. The parameters used for the three MOEAs compared in this study are presented in Table 4. The stopping criterion for the evolutionary process is the maximum number of target functions evaluations in all three cases. As classification datasets are used for our experiments, the k parameter (which determines the number of clusters in the output partition) can always be set to the optimal value, with the exception of the MOCK method, which does not need k to be specified. The optimal value of k for each dataset is featured in Table 2 under the column "No. Classes". Please note that this is the only parameter whose value depends on the dataset and cannot be computed solely from the information contained in it (which does not interfere with the semi-supervised foundations).

Parameter	Meaning	ME-MOEA/D$_{CC}$	MOEA/D$_{CC}$	MOCK	PESA-II$_{CC}$
$\|P\|$	Population size	100	100	100	100
|EP|	External population maximum size	1000	1000	1000	1000
Evals	Maximum target functions evaluations	300000	300000	300000	300000
ϵ	Connectivity neighborhood size	$\sqrt{n}$	$\sqrt{n}$	$\sqrt{n}$	$\sqrt{n}$
φ	Grid size	16	16	16	16
ϖ	Mutation probability	7%	7%	-	7%
δ	λ-neighborhood size	$\lfloor\|P\|/10\rfloor$	$\lfloor\|P\|/10\rfloor$	-	-
ν	Population elite size	$0.2 * \|P\|$	-	-	-
ζ	Probability that a feature is inherited from an elite parent	60%	-	-	-
γ	Probability for the elitist selection operator to be applied	20%	-	-	-
ξ	Maximum number of fails allowed in LS	$n \times 0.15$	-	-	-
k	Output partition number of clusters	No. Classes (Table 2)	No. Classes (Table 2)	-	No. Classes (Table 2)

Table 4: Parameters setup used for ME-MOEA/D$_{CC}$, MOEA/D$_{CC}$, MOCK and PESA-II$_{CC}$.

Table 5 describes parameter setup for the non-evolutionary state-of-the-art methods mentioned in Section 5.4. A Python implementation for these methods can be found at GitHub[4]. In all cases the value for k is equal to the number of classes displayed in Table 2. All parameter values for the non-evolutionary methods have been set following the guidelines of their authors.

Since our goal is not to optimize parameters in a case-by-case basis, but to get the fairest and most general comparison possible, we have not included any parameter tuning step for any

[4]https://github.com/GermangUgr/TFG/tree/master/Software

Method name	Parameters name and values
SHADE$_{CC}$	`max_eval` = 300000 `population_size` = 100 `prt_elite` = 0.25
COPKM	`max_iter` = 300 `tolerance` = $1 * 10^{-4}$ `init_mode` = ``rand''
LCVQE	`max_iter` = 300; `initial_centroids` = $\emptyset$
RDPM	`max_iter` = 300; $\xi_0 = 0.1$; $\xi_{\text{rate}} = 1$ λ is calculated on the basis of the mean distances in the dataset.
TVClust	`max_iter` = 300; $\alpha_0 = 1.2$ `stop_threshold` = $5 * 10^{-4}$

Table 5: Parameters setup used for non-evolutionary state-of-the-art methods.

competing method, not even for our proposal ME-MOEA/D$_{CC}$. Also, the high number of datasets used in our experiments makes tuning each parameter specifically for each dataset unfeasible in a reasonable time. The final purpose of this study is to provide a fair comparison between algorithms, assessing their robustness in a common environment with multiple datasets. All parameters concerning MOEAs have been set within reasonable values following the specific guidelines given by their authors and the general ones given in [50]

All our experiments have been carried out in the Hercules computing server of the University of Granada, which features 19 computing nodes with two 2.2 GHz Intel®Xeon Silver 4214 processor, 256 GB of RAM, a 6 TB SATA2 HDD and a 1 TB SSD in each node. Two Gigabit Ethernet internal nets are used to interconnect nodes. Ubuntu 20 LTS is installed in each node.

6 Experimental Results

In this section experimental results are reported. For the sake of readability, all the tables have been moved to Appendices A and B. Tables 6 to 11 in Appendix A present ARI and Unsat results, thereby comparing overall quality for all methods from the point of view of clustering. Tables 13 to 16 in Appendix B present Pareto approximations-related measures, enabling comparisons from the point of view of MOEAs. In order for clustering-quality comparisons to be made, a single solution from Pareto approximations found by the three MOEAs compared need to be chosen. As true labels are available, the chosen solution is one featuring the highest ARI in for all cases. Note that this would not be possible in an out-of-lab application or any other decision maker should be applied [72]. However, this study aims to prove general applicability of MOEAs to the CC problem, as well as the capability of our proposal, ME-MOEA/D$_{CC}$, to find high quality solutions. Extracting these solutions

from the Pareto approximation is a problem-dependent procedure, an expert or a specifically designed decision maker must be used in out-of-lab applications.

Given that all methods compared in this study rely on non-deterministic procedures, variations in the results can be found between different runs. Every method is run 30 times for every dataset and constraint set to mitigate the effects these non-deterministic procedures could have on the results. This results in a total of 16200 experiments. In all tables, the "Avg." column shows average results for every compared method in every dataset, while the "SD" column shows the Standard Deviation.

6.1 Clustering Quality Comparison

Tables from 6 to 11 (in Appendix A) present ARI and Unsat results for the CS_{10}, CS_{15}, CS_{20} constraint sets generated for every dataset. Note that, for the COPKM method, some results are missing. This is because COPKM is highly dependent on the order in which constraints are analyzed and it makes a hard interpretation of constraints. COPKM may not be able to find a solution for a given constraint set, even if this solution always exists, as constraints are generated on the basis of the true labels and no noise is introduced in the constraint set. This is a major drawback for the COPKM method, and cases where COPKM does not output a partition are considered to produce the worst benchmark values for plotting and averaging purposes.

Figures 7, 8 and 9 are used to compare average results for all methods, we refer to them as average plots. They allow a quick view of the distribution of ARI results produced for each method within the ARI output range $[-1, 1]$, with black marks representing average results for each method.

Tables 6 and 7 display results obtained by every method for the CS_{10} constraint set generated for every dataset. We can observe how our proposal, ME-MOEA/D$_{CC}$, represents a consistent improvement in average results over every one of the other 7 methods compared. It is able to achieve near-optimal results for datasets such as Breast Cancer and Appendicitis. Also note that, even if the COPKM method is able to obtain optimal results for a few datasets, the standard deviation is more than twice the average, making it very unreliable when looking for consistent quality. This can also be observed in Figure 7, where a clear polarization of results obtained by COPKM is featured. This will be the tendency for COPKM as the amount of constraint-based information is increased.

Results obtained for the CS$_{15}$ are presented in Tables 8 and 9. We continue to observe how MOEA/D$_{CC}$ outperforms most of the state-of-the-art methods, and offers results as competitive as those obtained by RDPM. Regarding the comparison with previous MOEAs, all of them are able to scale the quality of the results with the amount of constraint-based information, with our proposal and SHADE$_{CC}$ being the ones scaling to a greater extent, and SHADE$_{CC}$. Notably, SHADE$_{CC}$ goes from being one of the worst performing methods in Table 7 (with average results worst than 0.2) to being one of the best in Table 9 (with average close to 0.5). This can be clearly observed in Figure 8 when compared with Figure 7.

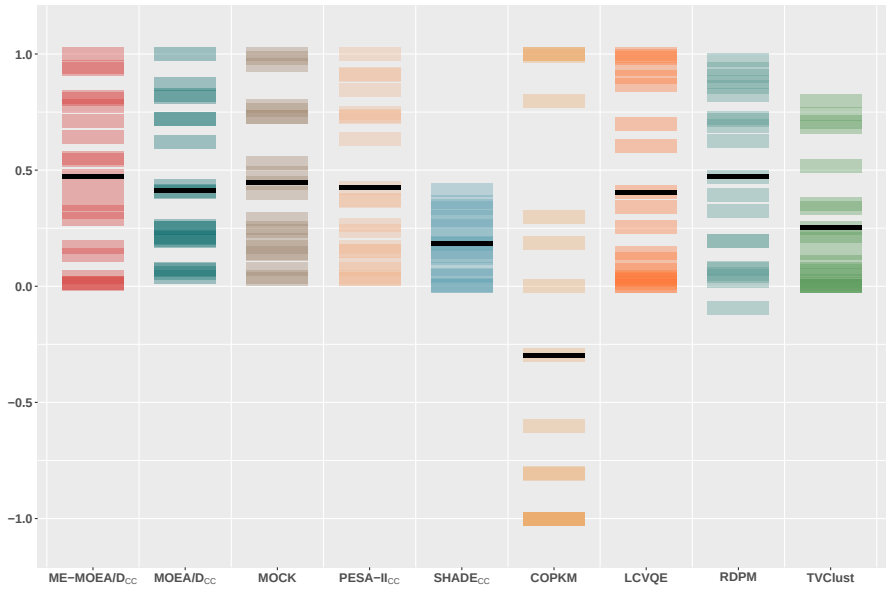


Figure 7: CS$_{10}$ comparative average plot.

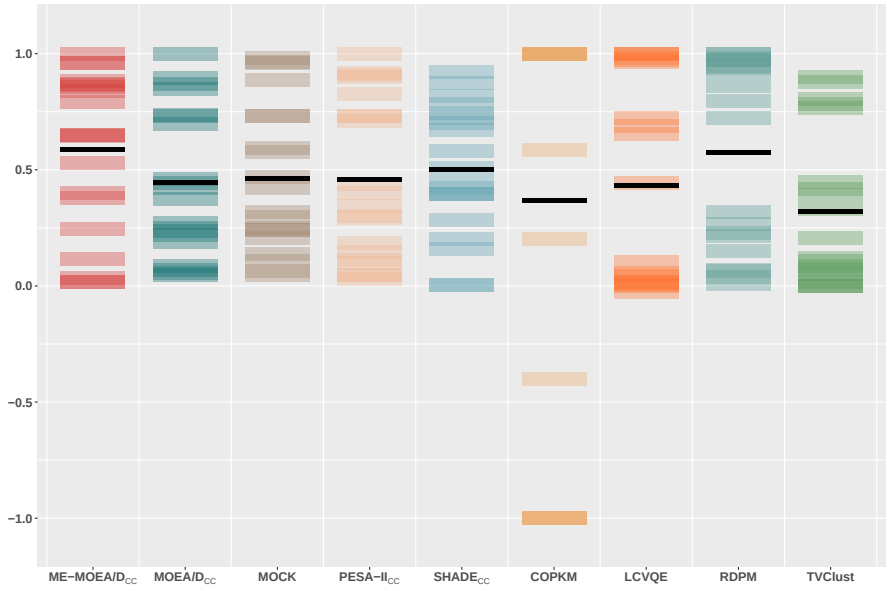


Figure 8: CS$_{15}$ comparative average plot.

In Tables 10 and 11, featuring results obtained for the CS_{20} constraint set, the scaling tendency observed in previous tables remains. Once again, our proposal outperforms the state-of-the-art, with RDPM and SHADE$_{CC}$ being the more disputed comparison. ME-MOEA/D$_{CC}$ continues to scale the quality of its results with the amount of constraint-based information. This is not the case for the other MOEAs, whose average ARI results increase to

a lesser extent. This is a sign of a proper constraint-integration scheme in ME-MOEA/D$_{CC}$; it is as well indicative of memetic elitism being beneficial for the CC problem. The scaling differences can be, once more, clearly observed by comparing Figure 8 with Figure 9.

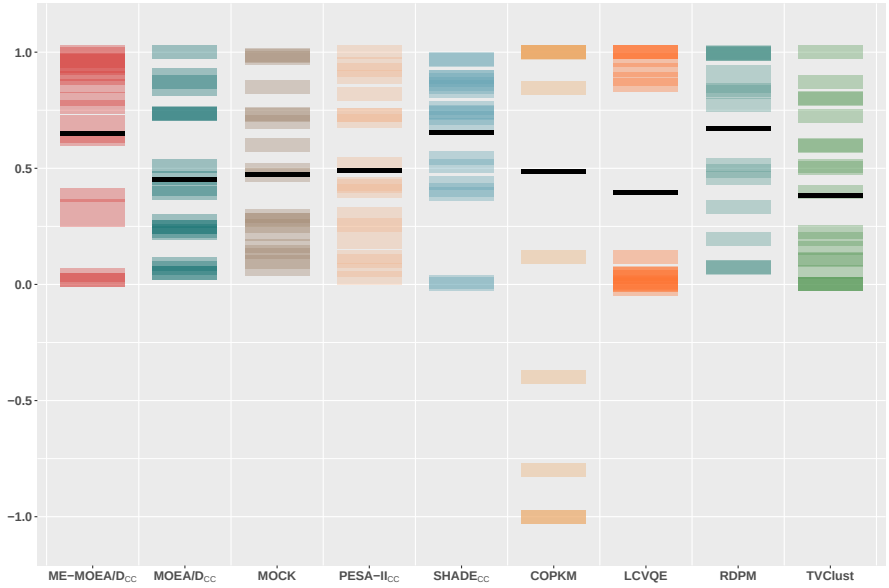


Figure 9: CS$_{20}$ comparative average plot.

It is also worth noting how PESA-II$_{CC}$—which is the label-based version of MOCK—obtains results consistently worst than those obtained by MOCK for all three levels of constraints. This is indicative of the capabilities of the PESA-II optimization scheme being more suitable for a graph-based representations scheme than for a label-based one. A significant improvement of ME-MOEA/D$_{CC}$ over PESA-II$_{CC}$ can also be observed, which is another evidence in favor of the CC problem benefiting from the specific ME-MOEA/D exploration-exploitation capabilities.

6.2 Pareto Approximation Quality Comparison

Tables 12 to 17 (in Appendix B) show values for 3 MOEAs-oriented measures. Hypervolume and $I_{\epsilon+}$ were introduced in Section 5.2. The size of the Pareto approximation found by every method has also been included. In order for comparisons to be made, all Pareto approximations have been normalized, thus the range for both the hypervolume and the $I_{\epsilon+}$ measures is set to $[0, 1]$.

With regards to the hypervolume measure, in the first four columns of Tables 13, 15 and 17 we observe how ME-MOEA/D$_{CC}$ consistently achieves more compact Pareto approximations than the other two competing methods. No compared method features a consistent increase or decrease in the hypervolume measure when the amount of constraint-based information is increased. However, by looking at the $I_{\epsilon+}$ measure, in Tables 12, 14 and 16, we observe how it contains very high quality solutions, even if the Pareto approximations

found by ME-MOEA/D$_{CC}$ are more compact than those obtained by other methods. The $I_{\epsilon+}$ measure indicates that Pareto approximation found by our proposal ϵ^+-dominate approximations found by other competing methods by a large margin in the majority of the cases, whereas the opposite situation does not happen. We also observe how this margin increases as we move from CS_{10} to CS_{20}, indicating that ME-MOEA/D$_{CC}$ is much more capable of integrating constraints into the clustering process than the other methods. It is also worth noting how PESA-II$_{CC}$ is able to obtain ϵ^+-indices as competitive as those obtained by ME-MOEA/D$_{CC}$ for CS$_{10}$, although it is quickly outscaled by ME-MOEA/D$_{CC}$ as the level of constraint-based information is increased.

Regarding the comparison of the Pareto approximation sizes in the last four columns of Tables 13, 15 and 17, we observe how the more populated Pareto approximations are always achieved by MOCK. This is clearly due to its graph-based representation scheme, which derives in a representation-related multimodality issue resulting in a partition having many different representations. This, in combination with MOCK actively trying to populate the less crowded regions of the Pareto approximation, by giving up resources that could be used to further improve promising regions, makes Pareto approximations achieve by MOCK excessively crowded. These conclusions are supported by the well-known fact that a more populated Pareto approximation does not mean a higher quality Pareto approximation.

Regarding the other three compared MOEAs—ME-MOEA/D$_{CC}$, MOEA/D$_{CC}$ and PESA-II$_{CC}$—, we observe how ME-MOEA/D$_{CC}$, even if it implements procedures to intensify the search in promising areas, finds and appropriate balance between the crowding of the Pareto approximation and the quality of the solutions in it. This is why it is able to find better solutions than any other MOEA with a better balanced Pareto approximation.

To summarize, the results presented in this section prove that our proposal ME-MOEA/D$_{CC}$ obtains average-sized Pareto approximations, with them being more compact than those obtained by the previous proposal MOCK and the base method MOEA/D$_{CC}$. Even if Pareto approximations found by our proposal are more compact, they contain very high quality solutions, as they ϵ^+-dominate Pareto approximations found by other proposals by a large margin in the vast majority of cases.

7 Statistical Analysis of Results

An empirical analysis comparing all ARI results obtained by every method—20 datasets combined with 3 constraint sets for a total of 60 results—can be performed by using the Bayesian signed-rank test. This test is similar to the Bayesian sign test described in Section 5.3, the only difference being that it sorts (ranks) differences by absolute values in the process of obtaining the Dirichlet distribution of the parameter ρ. Bearing this in mind the notation introduced in Section 5.3, we take the results obtained by a given state-of-the-art method as sample A, and the results obtained by ME-MOEA/D$_{CC}$ as sample B. As ARI is a measure to maximize, a higher value for $P(\rho < r_{\min}) = P(A - B < 0)$ would give the advantage to ME-MOEA/D$_{CC}$, whereas a high value for $P(\rho > r_{\max}) = P(A - B > 0)$ would mean

the opposite. The rope area has been set to rope $= [-0.02, 0.02]$, following the guidelines in [68].

A very illustrative visual representation of the results obtained by the Bayesian signed-rank test can be created. Once the result distribution has been sampled, it can be represented in the form of a heatmap by plotting every triplet in it as a point in barycentric coordinates in an equilateral triangle. In order to do so, each triplet value is associated with each of the three vertices of the triangle; the higher the value, the closer it is to its associated vertex. Values of every triplet must add up to one, since they describe a probability distribution, so all triplets lie within the triangle.

Figure 10 shows heatmaps comparing our proposal with all other 8 competing methods in a 1vs1 style, with color indicating the density of points in a given region: yellow represents high density and red is associated to low density. We can observe that no single triplet represented in the top region of any heatmap is shown. This means that the Bayesian signed-rank test assigns a very low probability to our proposal being equivalent to any other method. Heatmaps 10a, 10b and 10c compare our proposal with the three other MOEAs considered. They are all very similar, giving a clear advantage to our proposal ME-MOEA/D$_{CC}$ obtaining significantly better results than MOEA/D$_{CC}$, MOCK and PESA-II$_{CC}$ respectively. The same can be said for Heatmaps 10d, 10e, 10f and 10h, noting how in some cases the COPKM obtains competitive results. The most contested comparison corresponds to ME-MOEA/D$_{CC}$ vs RDPM, shown in Heatmap 10g, with both methods being statistically different, but with the advantage corresponding to one or the other depending on the dataset. However, we observe a slight tendency to the left on the point cloud, which means that the test confers a slight advantage in favor of our proposal ME-MOEA/D$_{CC}$.

Figure 11 shows a general Bayesian signed-rank test comparison, with the lower triangular matrix indicating the method given the advantage by the test, and the upper triangular matrix displaying rope probability values for those same comparisons. As Figure 11 shows, ME-MOEA/D$_{CC}$ is given the advantage in all cases. The Bayesian signed-rank test suggests to never use COPKM, due to its unreliability to obtain results. It also indicates that the previous MOEAs obtain competitive results depending on the case, with the comparisons regarding MOEA/D$_{CC}$, MOCK and PESA-II$_{CC}$ featuring a high value for rope, indicating a non-negligible probability of both methods being statistically identical.

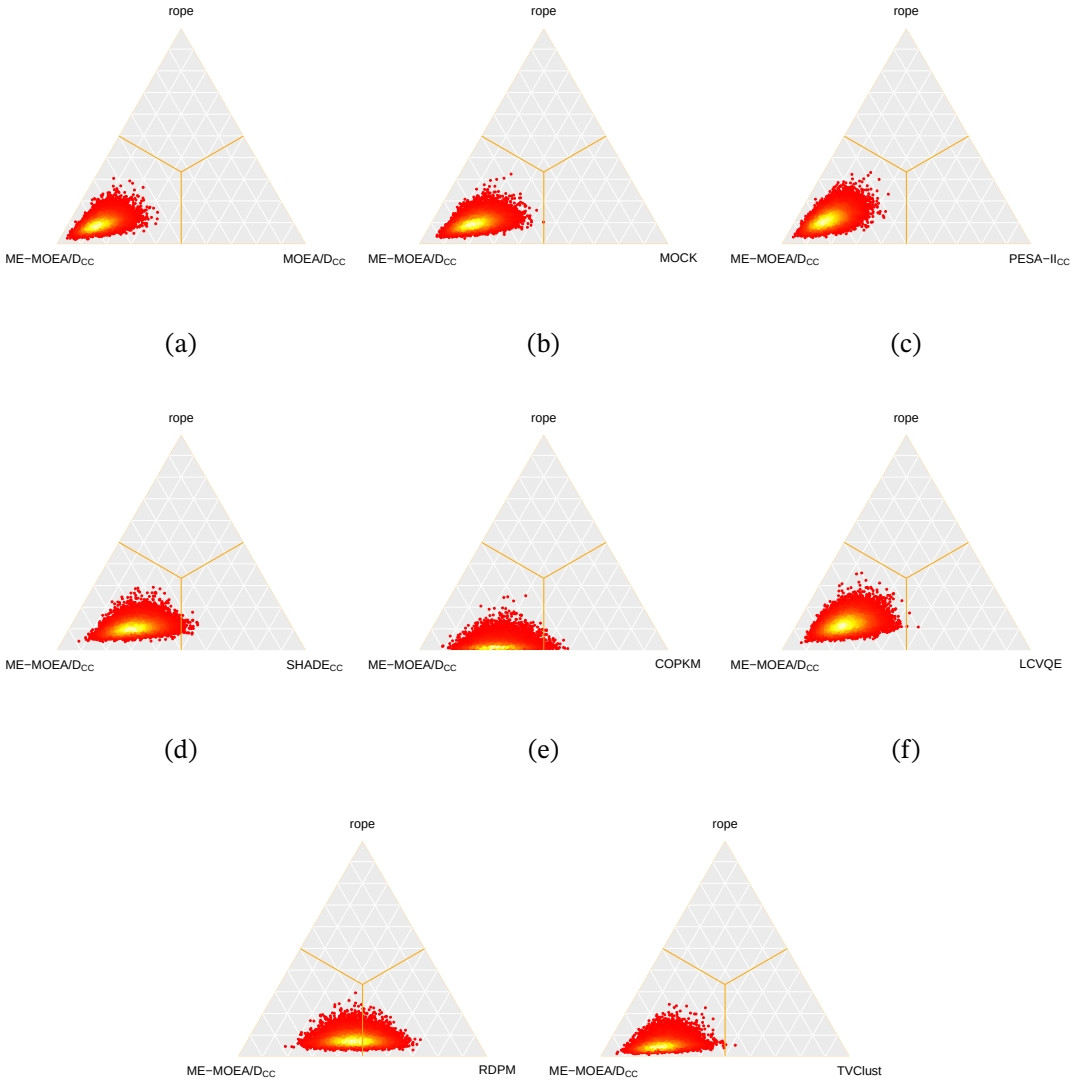


Figure 10: Heatmaps comparing ME-MOEA/D$_{CC}$ with all other compared methods.

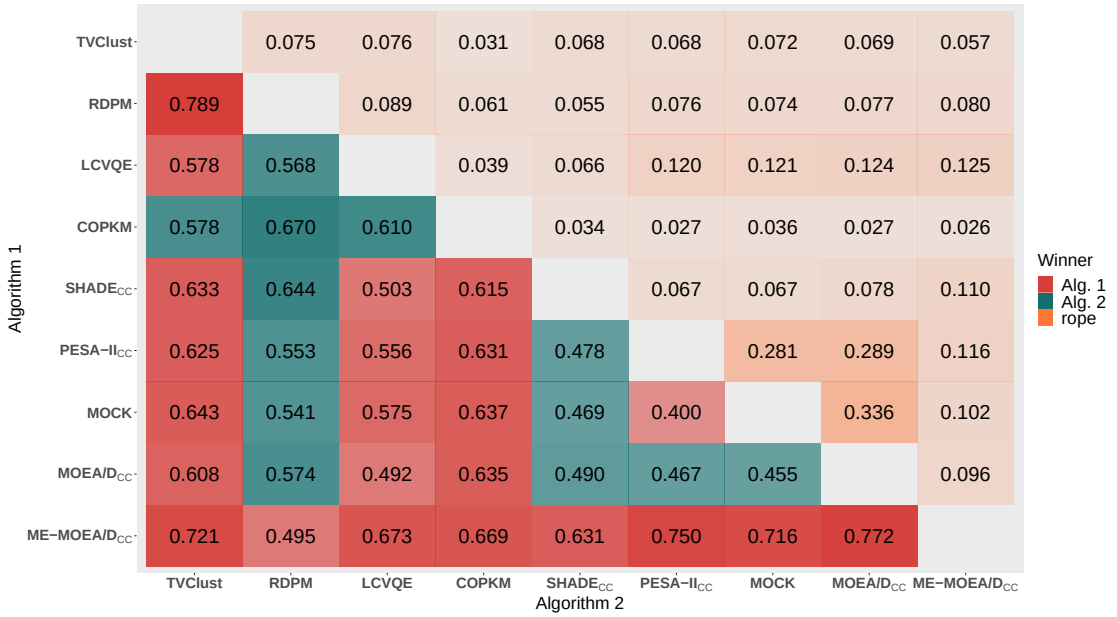


Figure 11: Overall Bayesian signed-rank test comparisons.

8 Conclusions

In this study the ME-MOEA/D (Memetic Elitist - Multiobjective Optimization Evolutionary Algorithm based on Decomposition) is presented. It implements exploitation procedures able to improve the quality of the solutions stored in the Pareto approximation without seriously interfering with the exploration capabilities of the base method MOEA/D. ME-MOEA/D$_{CC}$ is the result of integrating a constrained clustering application scheme, which we describe in the study, into ME-MOEA/D. A biased crossover operator is used to bias the search towards high quality solutions and an adaptive archiving method lessens the effects of multimodality in the constrained clustering problem.

A total of 8 constrained clustering methods (including 3 other MOEAs) are compared in this study. Experimental results over 20 datasets and 3 constraint sets for every dataset (for a total of 60) show a consistent advantage in favor of ME-MOEA/D$_{CC}$ with respect to the rest of the methods. Regarding the quality comparisons for Pareto approximations, our proposal has proven to produce Pareto approximations containing better individual solutions than those obtained by previous approaches, notwithstanding the Pareto approximations obtained by ME-MOEA/D$_{CC}$ being more compact. Bayesian statistical tests have been used to validate these results. The Bayesian signed-rank test assigns a higher probability to our proposal being better on average than any other method presented in this study. The broad scope of the experimentation carried out makes the conclusions drawn in this study generalizable to real-world applications, as real-world benchmark datasets are used to test all compared methods. Therefore, our proposed ME-MOEA/D$_{CC}$ could be implemented in any of the

CC applications introduced in Section 1, after a detailed study of the particularities of the problem analyzed and with a specific parameter tuning step.

To bring this study to and end, the following conclusions can be extracted:

- In the vast majority of the cases, our proposal ME-MOEA/D$_{CC}$ obtains better average results than the 8 previous approaches to the CC problem presented in this study.
- The Bayesian signed-rank test assigns a high probability to the differences between sets of results being statistically relevant.
- The Bayesian signed-rank test assigns a high probability to this differences being in favor of ME-MOEA/D$_{CC}$.
- ME-MOEA/D$_{CC}$ generates Pareto approximations containing better solutions than previous multiobjective approaches to the CC problem.
- ME-MOEA/D$_{CC}$ generates more compact Pareto approximations than previous multiobjective approaches to the CC problem.

Acknowledgements

This study is has been funded by the research projects TIN2017-89517-P and PP2019.PRI.I.06.

A Clustering Quality Comparison - Tables

This appendix presents result tables for the clustering quality comparison. Tables 6, 8 and 10 show results for ARI measure. Tables 7, 9 and 11 present results for Unsat measure. All these tables are referenced and analyzed in Section 6.1.

ARI Results for CS_{10}																		
Dataset	ME-MOEA/D_{CC}		MOEA/D_{CC}		MOCK		PESA-II_{CC}		SHADE$_{CC}$		COPKM		LCVQE		RDPM		TVClust	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	.817	.067	.622	.031	.758	.078	**.908**	**.016**	.073	.054	-	-	.257	.000	.322	.087	.106	.227
Banana*	.170	.024	.075	.016	.240	.010	.075	.008	.180	.134	**1.00**	**.000**	.012	.000	.973	.006	.202	.399
Breast Cancer	**.937**	**.020**	.721	.002	.730	.000	.730	.000	.416	.249	.000	1.00	.903	.000	.859	.048	.023	.049
Bupa	.013	.006	.056	.004	.073	.015	.033	.006	.005	.004	**.188**	**.018**	.031	.005	.074	.039	.065	.024
Contraceptive	.013	.003	.039	.006	.039	.005	.033	.003	.003	.003	-	-	.042	.008	**.688**	**.282**	.011	.017
Heart	**.712**	**.032**	.432	.027	.486	.012	.632	.027	.123	.158	-	-	.603	.000	.197	.061	.216	.254
Ionosphere	**.544**	**.138**	.261	.054	.532	.062	.372	.013	.345	.290	-.807	.580	.342	.000	.195	.061	.000	.000
Iris	.777	.067	.813	.069	**.998**	**.004**	.847	.024	.199	.106	.297	.849	.868	.000	.626	.088	.516	.051
Monk2	.008	.005	.213	.123	.032	.025	.039	.070	.260	.258	-	-	**.699**	**.004**	.023	.007	.336	.439
Newthyroid	.643	.022	.823	.025	.776	.052	.748	.024	.048	.067	-.802	.595	**.937**	**.000**	.715	.099	.743	.134
Phoneme*	.321	.052	.253	.003	.171	.025	.153	.015	.132	.050	**1.00**	**.000**	.001	.003	.936	.035	.707	.448
Pima	.415	.062	.209	.017	.249	.016	.239	.019	.186	.217	-.600	.800	.027	.000	**.878**	**.062**	.799	.391
Saheart	.288	.072	.194	.025	.200	.020	.265	.022	.363	.235	-	-	.072	.000	.078	.043	**.685**	**.334**
Soybean	**1.00**	**.000**	**1.00**	**.000**	.982	.009	**1.00**	**.000**	.238	.087	.798	.251	**1.00**	**.000**	.822	.086	.000	.000
Spectfheart	.355	.136	**.404**	**.039**	.401	.071	.154	.048	.298	.184	-	-	.002	.000	-.093	.035	.000	.000
Tae	.136	.020	**.199**	**.041**	.142	.010	.170	.019	.044	.030	-	-	.116	.007	.052	.022	.050	.010
Titanic*	.553	.073	.252	.010	.288	.004	.366	.018	.337	.191	**1.00**	**.000**	.145	.021	.393	.136	.145	.099
Vowel	.038	.008	.068	.007	**.137**	**.000**	.095	.004	.002	.002	-	-	.056	.006	.043	.015	.076	.023
Wdbc	.944	.008	.722	.012	.730	.000	.732	.003	.285	.219	**.993**	**.000**	**.993**	**.000**	.906	.055	.000	.000
Wine	.805	.063	.870	.022	.953	.009	.916	.036	.148	.052	-	-	**.982**	**.000**	.725	.086	.356	.104
Average	**.474**	**.335**	0.411	0.305	0.446	0.327	0.425	0.338	0.184	0.127	-0.297	0.826	0.405	0.400	0.471	0.365	0.252	0.277

Table 6: Experimental results for the ARI measure obtained by ME-MOEA/D_{CC} and 8 previous approaches in the CS_{10} constraint set.

Unsat Results for CS_{10}																		
Dataset	ME-MOEA/D$_{CC}$		MOEA/D$_{CC}$		MOCK		PESA-II$_{CC}$		SHADE$_{CC}$		COPKM		LCVQE		RDPM		TVClust	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	4.36	3.37	16.36	3.63	8.00	4.95	**0.00**	**0.00**	**0.00**	**0.00**	-	-	21.81	0.00	32.72	5.07	38.54	10.96
Banana*	38.78	0.89	45.87	0.72	37.77	0.63	43.40	0.24	33.90	4.27	**0.00**	**0.00**	48.48	0.01	0.55	0.12	38.94	19.48
Breast Cancer	**2.58**	**0.72**	13.48	0.36	13.65	0.00	13.65	0.00	17.62	6.14	50.00	50.00	4.07	0.00	4.74	1.82	45.99	2.30
Bupa	16.94	0.54	27.49	8.11	29.64	7.36	17.27	2.14	12.98	0.24	**0.00**	**0.00**	7.96	0.75	13.76	3.10	21.71	8.16
Contraceptive	44.27	1.05	42.78	2.76	42.49	0.86	41.50	0.67	36.81	2.98	-	-	43.75	1.11	**10.31**	**9.84**	51.15	3.17
Heart	**7.74**	**1.52**	27.23	2.25	23.13	0.89	12.19	0.79	12.39	2.76	-	-	14.81	0.00	36.09	4.10	36.92	14.75
Ionosphere	14.98	4.74	31.23	2.64	20.95	1.93	21.23	0.91	**12.34**	**5.56**	90.00	30.00	27.46	0.00	34.52	3.68	46.34	0.00
Iris	6.66	2.33	5.90	2.20	**0.00**	**0.00**	4.19	2.86	**0.00**	**0.00**	30.00	45.82	2.85	0.00	14.00	5.65	18.66	3.16
Monk2	36.82	3.45	37.88	5.66	39.36	3.80	37.18	4.93	18.20	5.93	-	-	**11.20**	**0.00**	46.14	0.81	32.56	22.17
Newthyroid	10.82	2.33	7.79	1.22	8.39	2.09	5.80	1.53	8.22	1.75	90.00	30.00	**0.86**	**0.00**	8.31	2.47	8.26	5.03
Phoneme*	32.81	2.47	36.91	0.18	35.40	0.49	40.33	0.79	35.68	1.13	**0.00**	**0.00**	48.83	0.13	2.13	1.33	14.37	21.98
Pima	25.27	2.46	38.25	0.81	33.07	0.78	33.52	0.94	27.12	5.21	80.00	40.00	46.03	0.00	**3.30**	**1.75**	9.64	19.02
Saheart	26.51	2.92	36.37	1.97	33.71	0.75	28.60	1.37	16.79	5.16	-	-	39.31	0.00	42.14	2.73	**11.69**	**16.30**
Soybean	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	2.00	6.00	90.00	0.00
Spectfheart	17.66	2.97	20.22	1.82	17.26	2.76	22.33	2.59	**9.82**	**2.81**	-	-	33.90	0.00	41.28	2.52	34.47	0.00
Tae	18.83	2.50	18.00	3.18	17.66	1.22	14.66	2.96	**2.08**	**1.25**	-	-	14.08	0.25	48.75	8.55	47.91	8.69
Titanic*	18.67	3.35	34.12	1.03	32.56	0.29	27.32	0.44	20.62	4.90	**0.00**	**0.00**	39.48	0.54	26.18	6.26	39.69	4.20
Vowel	16.49	0.43	17.62	1.57	10.36	0.00	15.44	0.36	13.25	0.08	-	-	**9.84**	**0.44**	18.01	2.28	18.61	2.92
Wdbc	2.36	0.39	14.24	0.65	13.65	0.00	12.74	1.17	21.85	4.97	**0.00**	**0.00**	**0.00**	**0.00**	3.46	2.63	45.86	0.00
Wine	5.35	2.96	4.83	1.46	2.35	0.97	2.09	2.00	1.47	0.71	-	-	**0.00**	**0.00**	8.62	3.15	23.79	8.33
Average	17.39	12.77	23.83	13.44	20.97	13.43	19.67	13.81	**15.06**	**11.45**	62.00	44.22	20.74	17.86	19.85	16.38	33.76	18.72

Table 7: Experimental results for the Unsat measure obtained by ME-MOEA/D$_{CC}$ and 8 previous approaches in the CS_{10} constraint set.

ARI Results for CS_{15}																		
Dataset	ME-MOEA/D$_{CC}$		MOEA/D$_{CC}$		MOCK		PESA-II$_{CC}$		SHADE$_{CC}$		COPKM		LCVQE		RDPM		TVClust	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	.881	.041	.699	.071	.595	.033	**.898**	**.020**	.395	.357	-	-	.443	.000	.265	.144	.050	.166
Banana*	.247	.054	.084	.017	.242	.018	.112	.015	.508	.033	**1.00**	**.000**	.004	.000	**1.00**	**.000**	.208	.396
Breast Cancer	.961	.029	.731	.007	.730	.000	.730	.000	.723	.041	**1.00**	**.000**	.965	.000	.971	.021	.015	.040
Bupa	.015	.004	.056	.003	.066	.007	.029	.003	.002	.005	**.585**	**.008**	.042	.008	.150	.046	.120	.071
Contraceptive	.018	.006	.047	.003	.048	.008	.045	.004	.203	.037	**1.00**	**.000**	.015	.006	.971	.015	.072	.206
Heart	.858	.050	.442	.025	.469	.062	.710	.031	.874	.103	**1.00**	**.000**	.687	.000	.213	.072	.333	.389
Ionosphere	.839	.022	.422	.018	.578	.023	.437	.025	.701	.176	**1.00**	**.000**	.656	.000	.319	.067	.000	.000
Iris	.863	.082	.848	.013	.966	.021	.918	.032	.420	.197	-	-	**.980**	**.000**	.797	.137	.785	.211
Monk2	.402	.280	.373	.083	.318	.043	.046	.031	.819	.045	**1.00**	**.000**	.725	.000	.037	.009	.416	.479
Newthyroid	.792	.061	.868	.023	.887	.045	.825	.017	.283	.249	-.400	.917	**1.00**	**.000**	.982	.019	.876	.186
Phoneme*	.378	.063	.251	.003	.207	.009	.185	.006	.446	.042	**1.00**	**.000**	-.026	.000	1.00	.000	.898	.307
Pima	.529	.103	.235	.015	.251	.016	.293	.014	.580	.226	**1.00**	**.000**	.017	.000	.937	.057	.804	.391
Saheart	.647	.039	.219	.012	.242	.022	.343	.041	.783	.044	**1.00**	**.000**	-.001	.000	.230	.108	.767	.372
Soybean	**1.00**	**.000**	**1.00**	**.000**	.983	.015	**1.00**	**.000**	.397	.125	.200	.980	**1.00**	**.000**	.862	.110	.000	.000
Spectfheart	.652	.046	.460	.031	.423	.044	.303	.032	.921	.023	**1.00**	**.000**	.010	.013	.008	.246	.000	.000
Tae	.118	.025	**.188**	**.024**	.106	.017	.145	.020	.157	.101	-	-	.103	.000	.068	.032	.058	.014
Titanic*	.648	.078	.273	.014	.297	.008	.401	.012	.670	.074	**1.00**	**.000**	.056	.000	.920	.048	.448	.331
Vowel	.033	.005	.069	.006	**.138**	**.001**	.099	.004	.002	.002	-	-	.042	.006	.064	.010	.086	.028
Wdbc	.959	.013	.734	.012	.730	.000	.730	.000	.745	.026	**1.00**	**.000**	.972	.000	.947	.036	.108	.288
Wine	.869	.029	.897	.017	.958	.009	.906	.013	.395	.123	-	-	**1.00**	**.000**	.723	.094	.391	.132
Average	**0.585**	**0.336**	.445	.308	.462	.312	.458	.336	.501	.272	.369	.861	.434	.430	.573	.394	.322	.322

Table 8: Experimental results for the ARI measure obtained by ME-MOEA/D$_{CC}$ and 8 previous approaches in the CS_{15} constraint set.

Unsat Results for CS_{15}																		
Dataset	ME-MOEA/D$_{CC}$		MOEA/D$_{CC}$		MOCK		PESA-II$_{CC}$		SHADE$_{CC}$		COPKM		LCVQE		RDPM		TVClust	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**0.83**	**0.74**	7.50	1.74	10.33	0.40	1.00	0.33	6.25	4.87	-	-	19.16	0.00	29.25	7.09	40.75	11.14
Banana*	37.30	2.49	45.61	0.93	37.90	0.74	43.20	0.89	23.50	1.50	**0.00**	**0.00**	50.22	0.00	**0.00**	**0.00**	39.32	19.66
Breast Cancer	1.92	1.40	13.61	0.40	13.51	0.00	13.38	0.26	12.48	1.85	**0.00**	**0.00**	1.91	0.00	1.06	0.72	46.17	1.78
Bupa	18.91	1.24	32.00	8.67	45.09	10.99	17.51	1.95	16.13	0.40	**0.00**	**0.00**	12.11	0.56	12.31	1.73	15.46	3.64
Contraceptive	43.96	0.44	43.53	2.61	45.06	4.41	41.47	0.21	30.95	1.27	**0.00**	**0.00**	46.31	0.71	0.88	0.50	46.72	11.10
Heart	5.80	2.18	27.17	0.96	25.61	2.41	12.82	1.81	4.69	4.09	**0.00**	**0.00**	11.82	0.00	35.64	4.89	31.14	18.84
Ionosphere	6.48	0.85	28.47	0.67	20.33	1.37	24.58	1.78	10.88	5.60	**0.00**	**0.00**	15.53	0.00	32.38	3.60	46.44	0.00
Iris	4.11	0.85	6.32	0.75	1.10	0.38	3.08	1.26	4.54	1.59	-	-	**0.00**	**0.00**	7.74	8.17	8.18	9.93
Monk2	25.56	11.10	31.01	3.77	32.15	1.65	39.72	1.24	7.50	2.20	**0.00**	**0.00**	12.78	0.00	45.48	1.01	28.56	23.49
Newthyroid	8.59	3.53	6.02	1.28	5.56	2.01	6.89	0.90	15.10	7.73	70.00	45.82	**0.00**	**0.00**	0.39	0.47	5.34	9.46
Phoneme*	30.72	3.24	37.85	0.15	35.78	0.26	40.03	0.42	26.05	2.08	**0.00**	**0.00**	49.73	0.00	0.00	0.00	5.01	15.03
Pima	22.12	4.38	37.55	0.44	34.73	0.64	32.92	0.88	17.94	8.91	**0.00**	**0.00**	48.84	0.00	1.78	1.73	9.65	19.31
Saheart	14.94	1.56	36.18	0.98	34.24	1.15	28.43	2.02	8.99	2.09	**0.00**	**0.00**	50.51	0.00	33.61	5.15	11.39	18.44
Soybean	**0.00**	**0.00**	**0.00**	**0.00**	0.71	1.42	**0.00**	**0.00**	**0.00**	**0.00**	40.00	48.99	**0.00**	**0.00**	6.07	4.24	60.71	0.00
Spectfheart	10.39	1.19	21.07	0.90	17.17	1.95	23.80	1.08	1.86	0.72	**0.00**	**0.00**	41.26	0.41	37.22	8.61	31.34	0.00
Tae	25.77	3.59	32.33	3.55	30.67	0.64	21.50	1.77	**8.20**	**1.40**	-	-	33.59	0.00	39.92	7.82	44.66	3.18
Titanic*	15.85	3.39	34.33	0.37	33.30	0.25	27.23	0.59	14.25	2.95	**0.00**	**0.00**	46.28	0.00	3.42	2.26	26.41	16.03
Vowel	16.07	0.25	18.13	0.95	**10.76**	**0.53**	15.32	0.18	14.40	0.14	-	-	13.73	0.18	12.27	0.66	17.87	2.13
Wdbc	1.74	0.54	13.81	0.38	14.33	0.00	14.33	0.00	10.84	1.03	**0.00**	**0.00**	1.01	0.00	2.06	1.35	41.22	13.09
Wine	4.16	0.73	3.02	0.81	1.08	0.33	2.16	0.29	6.12	0.31	-	-	**0.00**	**0.00**	7.37	1.86	28.43	9.54
Average	14.76	12.46	23.77	13.91	22.47	14.40	20.47	13.79	**12.03**	**7.84**	30.50	43.52	22.74	19.94	15.44	15.89	29.24	16.02

Table 9: Experimental results for the Unsat measure obtained by ME-MOEA/D$_{CC}$ and 8 previous approaches in the CS_{15} constraint set.

ARI Results for CS_{20}																		
Dataset	ME-MOEA/D_{CC}		MOEA/D_{CC}		MOCK		PESA-II_{CC}		SHADE$_{CC}$		COPKM		LCVQE		RDPM		TVClust	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**.991**	**.017**	.735	.054	.701	.061	.948	.033	.759	.417	-	-	.117	.000	.459	.097	.158	.282
Banana*	.337	.033	.088	.017	.253	.032	.120	.017	.545	.017	**1.00**	**.000**	.000	.000	**1.00**	**.000**	.110	.297
Breast Cancer	.965	.014	.737	.009	.730	.000	.731	.002	.831	.020	**1.00**	**.000**	**1.00**	**.000**	.992	.007	.194	.389
Bupa	.018	.006	.051	.004	.067	.008	.029	.003	.009	.005	**.844**	**.007**	.032	.006	.334	.110	.400	.083
Contraceptive	.018	.002	.048	.005	.096	.067	.061	.006	.391	.115	**1.00**	**.000**	.010	.011	.997	.002	.226	.190
Heart	.930	.043	.458	.020	.471	.020	.703	.051	.971	.027	**1.00**	**.000**	.857	.000	.515	.176	.502	.454
Ionosphere	.904	.040	.412	.030	.601	.018	.518	.035	.891	.022	**1.00**	**.000**	.007	.000	.490	.086	.000	.000
Iris	.941	.037	.841	.025	**.984**	**.019**	.925	.025	.893	.128	-	-	.886	.000	.837	.203	.597	.141
Monk2	.762	.137	.394	.066	.276	.041	.257	.111	.879	.046	**1.00**	**.000**	.925	.009	.077	.032	.727	.423
Newthyroid	.798	.039	.870	.012	.848	.022	.822	.018	.743	.123	-.400	.917	.968	.000	**.992**	**.014**	.871	.169
Phoneme*	.384	.046	.246	.013	.164	.034	.182	.008	.511	.057	**1.00**	**.000**	-.021	.000	.991	.026	.799	.401
Pima	.626	.055	.232	.025	.276	.009	.302	.020	.736	.041	**1.00**	**.000**	.043	.000	.997	.002	**1.00**	**.000**
Saheart	.637	.066	.249	.018	.216	.016	.402	.033	.861	.018	**1.00**	**.000**	-.001	.000	.853	.020	.803	.394
Soybean	**1.00**	**.000**	**1.00**	**.000**	.987	.011	**1.00**	**.000**	.435	.155	.118	.925	**1.00**	**.000**	.829	.114	.000	.000
Spectfheart	.854	.050	.510	.042	.494	.049	.423	.065	.967	.039	**1.00**	**.000**	-.004	.001	.197	.438	.000	.000
Tae	.278	.036	.220	.028	.125	.007	.245	.017	**.407**	**.171**	-	-	.048	.007	.069	.042	.055	.034
Titanic*	.701	.059	.275	.006	.293	.004	.432	.013	.716	.053	**1.00**	**.000**	.045	.000	.913	.039	.512	.291
Vowel	.039	.007	.070	.006	.140	.002	.101	.003	.002	.002	**.998**	**.000**	.031	.004	.077	.010	.104	.024
Wdbc	.964	.015	.732	.009	.733	.005	.730	.000	.848	.030	**1.00**	**.000**	**1.00**	**.000**	.995	.005	.000	.000
Wine	.888	.048	.903	.007	.976	.011	.893	.023	.686	.152	-.800	.600	**1.00**	**.000**	.772	.060	.600	.232
Average	.652	.338	.454	.308	.472	.315	.491	.319	.654	.279	.488	.799	.397	.457	**.669**	**.346**	.383	.329

Table 10: Experimental results for the ARI measure obtained by ME-MOEA/D_{CC} and 8 previous approaches in the CS_{20} constraint set.

Unsat Results for CS_{20}																		
Dataset	ME-MOEA/D$_{CC}$		MOEA/D$_{CC}$		MOCK		PESA-II$_{CC}$		SHADE$_{CC}$		COPKM		LCVQE		RDPM		TVClust	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**0.17**	**0.34**	11.42	3.72	12.29	2.54	1.21	0.63	4.43	7.68	-	-	41.12	0.00	19.91	3.16	32.38	11.22
Banana*	33.10	1.63	45.66	0.74	37.13	1.55	43.16	0.83	22.18	0.87	**0.00**	**0.00**	49.85	0.02	**0.00**	**0.00**	44.28	14.76
Breast Cancer	1.61	0.75	12.60	0.56	13.15	0.00	13.03	0.22	7.54	0.72	**0.00**	**0.00**	**0.00**	**0.00**	0.30	0.26	39.37	19.02
Bupa	19.15	0.71	27.98	6.96	40.13	13.78	17.96	0.77	16.91	0.50	**0.00**	**0.00**	15.35	0.36	8.52	1.69	11.87	4.03
Contraceptive	43.70	0.19	45.17	2.53	39.18	3.32	41.33	0.30	25.72	4.35	**0.00**	**0.00**	46.93	1.54	0.07	0.03	37.56	9.39
Heart	2.64	1.66	26.30	1.16	25.36	0.55	12.63	2.72	1.15	1.21	**0.00**	**0.00**	6.70	0.00	20.47	8.61	24.83	22.58
Ionosphere	4.08	1.77	28.92	1.52	19.59	0.70	21.94	1.51	4.74	1.15	**0.00**	**0.00**	48.65	0.00	24.42	4.44	45.39	0.00
Iris	1.88	1.34	6.16	1.43	**0.69**	**0.84**	2.94	0.76	1.66	2.09	-	-	3.67	0.00	5.21	7.85	19.49	7.81
Monk2	11.33	6.48	29.15	3.08	34.60	1.85	33.61	4.70	5.48	2.06	**0.00**	**0.00**	3.37	0.44	42.20	1.81	13.49	20.90
Newthyroid	8.48	1.54	5.71	0.26	6.93	1.43	7.33	1.02	7.91	4.13	70.00	45.82	2.10	0.00	**0.29**	**0.49**	5.62	7.69
Phoneme*	30.36	2.22	37.28	0.86	36.77	0.91	40.42	0.37	23.54	2.64	**0.00**	**0.00**	49.48	0.00	0.30	0.92	9.88	19.76
Pima	18.26	2.77	38.01	1.23	33.91	0.31	33.40	0.58	12.33	1.99	**0.00**	**0.00**	47.78	0.00	0.05	0.05	**0.00**	**0.00**
Saheart	16.74	3.49	36.03	1.21	35.79	1.17	28.04	1.18	6.01	0.83	**0.00**	**0.00**	49.11	0.00	6.28	0.58	9.28	18.56
Soybean	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	40.00	48.99	**0.00**	**0.00**	2.00	2.09	88.88	0.00
Spectfheart	5.38	1.94	21.27	1.72	18.14	1.90	21.50	1.53	1.11	1.24	**0.00**	**0.00**	51.82	0.67	30.07	16.49	33.47	0.00
Tae	21.46	2.10	27.22	1.90	31.91	1.29	25.54	1.85	**10.05**	**2.50**	-	-	40.71	0.45	39.37	7.69	46.96	5.74
Titanic*	14.12	2.99	34.40	0.40	32.84	0.13	27.24	0.75	13.11	2.32	**0.00**	**0.00**	48.30	0.00	3.97	1.84	23.52	13.07
Vowel	16.11	0.11	18.10	1.03	10.88	0.35	15.38	0.19	14.95	0.03	**0.00**	**0.00**	15.90	0.34	11.24	0.47	17.00	1.26
Wdbc	1.54	0.67	13.60	0.50	13.60	0.39	13.80	0.00	6.75	1.37	**0.00**	**0.00**	**0.00**	**0.00**	0.19	0.15	46.46	0.00
Wine	3.23	1.17	3.49	0.68	0.66	0.55	3.23	0.59	5.95	2.81	90.00	30.00	**0.00**	**0.00**	6.15	1.61	18.07	10.92
Average	12.67	12.01	23.42	13.61	22.18	13.77	20.18	13.37	**9.58**	**7.48**	25.00	40.06	26.04	21.86	11.05	13.28	28.39	19.99

Table 11: Experimental results for the Unsat measure obtained by ME-MOEA/D$_{CC}$ and 8 previous approaches in the CS_{20} constraint set.

B Pareto Approximation Quality Comparison - Tables

This appendix presents result tables for the Pareto front approximation quality comparison. Tables 12, 14 and 16 show results for different setups of the ϵ^+-indicator. Tables 13, 15 and 17 present results for the hypervolume and the Pareto front size. All these tables are referenced and analyzed in Section 6.2.

Dataset	$I_{\epsilon^+}(\cdot,\cdot) : A \leftarrow$ ME-MOEA/D$_{CC}$ for CS_{10}											
	$B \leftarrow$ MOEA/D$_{CC}$				$B \leftarrow$ MOCK				$B \leftarrow$ PESA-II$_{CC}$			
	$I_{\epsilon^+}(A,B)$		$I_{\epsilon^+}(B,A)$		$I_{\epsilon^+}(A,B)$		$I_{\epsilon^+}(B,A)$		$I_{\epsilon^+}(A,B)$		$I_{\epsilon^+}(B,A)$	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**.212**	**.031**	.028	.031	**.332**	**.034**	.273	.073	.064	.017	**.105**	**.017**
Banana*	**.532**	**.095**	.054	.021	**.493**	**.105**	.049	.054	**.251**	**.099**	.101	.037
Breast Cancer	**.543**	**.134**	.305	.120	**.425**	**.105**	.243	.137	.087	.099	**.451**	**.154**
Bupa	**.286**	**.028**	.070	.020	**.244**	**.038**	.086	.019	**.183**	**.035**	.120	.038
Contraceptive	**.385**	**.076**	.046	.027	**.319**	**.035**	.213	.042	**.198**	**.120**	.157	.057
Heart	**.224**	**.028**	.103	.021	**.294**	**.025**	.052	.023	**.136**	**.032**	.070	.029
Ionosphere	**.275**	**.039**	.165	.095	**.372**	**.035**	.153	.128	.144	.039	**.229**	**.133**
Iris	**.267**	**.088**	.088	.058	**.303**	**.131**	.171	.067	.159	.077	**.254**	**.078**
Monk2	**.239**	**.036**	.103	.038	**.241**	**.037**	.059	.022	.051	.036	**.179**	**.038**
Newthyroid	**.191**	**.102**	.130	.056	**.175**	**.082**	.151	.075	.116	.036	**.399**	**.088**
Phoneme*	**.467**	**.132**	.297	.147	**.368**	**.089**	.149	.146	.222	.129	**.236**	**.191**
Pima	**.478**	**.031**	.093	.028	**.382**	**.055**	.097	.062	**.195**	**.043**	.148	.046
Saheart	**.358**	**.050**	.064	.019	**.335**	**.051**	.014	.006	.073	.028	**.114**	**.025**
Soybean	**.384**	**.025**	.131	.051	**.430**	**.028**	.136	.018	**.010**	**.036**	.000	.000
Spectfheart	.093	.035	**.208**	**.049**	**.153**	**.059**	.118	.040	.092	.040	**.114**	**.045**
Tae	**.213**	**.035**	.041	.023	**.295**	**.032**	.213	.081	.097	.025	**.104**	**.027**
Titanic*	**.469**	**.035**	.071	.015	**.410**	**.059**	.160	.015	.154	.061	**.164**	**.043**
Vowel	**.297**	**.083**	.155	.026	**.270**	**.084**	.061	.043	.136	.072	**.187**	**.064**
Wdbc	**.550**	**.053**	.278	.094	**.445**	**.068**	.151	.129	.166	.111	**.384**	**.089**
Wine	**.294**	**.103**	.136	.063	**.285**	**.106**	.154	.083	.196	.114	**.323**	**.102**
Average	**.338**	**.062**	.128	.050	**.329**	**.063**	.135	.063	.137	.062	**.192**	**.065**

Table 12: Experimental results for the ϵ^+-indicator obtained by ME-MOEA/D$_{CC}$ and 3 previous MO approaches in the CS_{10} constraint set.

Dataset	Hypervolume for CS$_{10}$								Pareto Approximation Size for CS$_{10}$							
	ME-MOEA/D$_{CC}$		MOEA/D$_{CC}$		MOCK		PESA-II$_{CC}$		ME-MOEA/D$_{CC}$		MOEA/D$_{CC}$		MOCK		PESA-II$_{CC}$	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	.489	.056	.659	.020	.681	.083	.407	.004	1000.0	0.0	254.8	26.1	995.2	9.6	997.2	5.6
Banana*	.199	.068	.637	.083	.645	.064	.312	.030	10.8	3.3	180.6	36.1	485.4	209.5	94.8	18.8
Breast Cancer	.324	.177	.494	.055	.429	.104	.064	.050	25.2	11.1	67.6	10.7	945.4	54.9	28.0	1.4
Bupa	.479	.058	.764	.009	.624	.039	.548	.055	39.6	11.8	240.4	16.0	1000.0	0.0	113.6	24.4
Contraceptive	.443	.103	.827	.016	.564	.029	.384	.041	35.8	13.7	303.4	32.4	999.8	0.4	111.2	1.3
Heart	.416	.015	.559	.021	.660	.042	.475	.066	542.8	199.0	100.2	16.4	998.2	2.7	66.2	7.7
Ionosphere	.447	.085	.594	.044	.624	.024	.369	.046	331.2	70.1	96.8	18.9	999.8	0.4	88.0	6.6
Iris	.425	.132	.633	.071	.539	.127	.320	.075	45.2	17.4	48.2	5.7	927.2	129.2	39.8	5.8
Monk2	.462	.053	.575	.050	.631	.036	.333	.051	143.4	39.7	148.6	8.5	997.4	2.3	74.4	16.9
Newthyroid	.701	.148	.801	.035	.746	.064	.263	.061	35.4	19.0	60.0	13.7	1000.0	0.0	44.0	4.3
Phoneme*	.288	.128	.503	.030	.532	.032	.275	.024	11.8	4.7	83.6	10.9	1000.0	0.0	81.2	6.9
Pima	.300	.055	.561	.046	.427	.020	.271	.020	31.6	24.1	115.0	9.0	992.0	11.1	90.0	5.4
Saheart	.379	.047	.585	.041	.652	.049	.313	.011	180.2	81.8	119.0	6.2	858.4	160.0	93.0	11.4
Soybean	.418	.045	.697	.072	.651	.047	.441	.000	1000.0	0.0	1000.0	0.0	1000.0	0.0	1000.0	0.0
Spectfheart	.523	.057	.405	.029	.542	.034	.532	.040	459.4	122.7	83.4	22.6	999.6	0.8	85.8	15.6
Tae	.551	.031	.760	.050	.752	.065	.504	.045	127.8	58.9	133.8	18.8	999.6	0.8	127.6	5.0
Titanic*	.345	.048	.822	.037	.746	.033	.264	.040	59.6	41.4	95.2	13.5	760.2	114.1	96.0	8.5
Vowel	.448	.138	.824	.030	.726	.032	.460	.049	19.8	2.4	193.6	41.5	997.6	4.8	61.8	3.8
Wdbc	.299	.140	.434	.031	.425	.086	.137	.078	25.0	6.3	66.8	26.5	945.2	54.8	32.8	5.6
Wine	.491	.171	.682	.058	.610	.065	.302	.120	28.2	15.4	46.2	4.0	1000.0	0.0	20.2	2.1
Average	.421	.109	.641	.127	.610	.100	.349	.122	207.6	302.9	171.8	202.6	945.0	121.1	167.2	278.6

Table 13: Experimental results for the Hypervolume and the Pareto Approximation Size measures obtained by ME-MOEA/D$_{CC}$ and 3 previous MO approaches in the CS_{10} constraint set.

Dataset	$I_{\epsilon^+}(\cdot,\cdot) : A \leftarrow$ ME-MOEA/D$_{CC}$ for CS_{15}											
	$B \leftarrow$ MOEA/D$_{CC}$				$B \leftarrow$ MOCK				$B \leftarrow$ PESA-II$_{CC}$			
	$I_{\epsilon^+}(A,B)$		$I_{\epsilon^+}(B,A)$		$I_{\epsilon^+}(A,B)$		$I_{\epsilon^+}(B,A)$		$I_{\epsilon^+}(A,B)$		$I_{\epsilon^+}(B,A)$	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**.247**	**.028**	.052	.023	.353	.026	**.356**	**.061**	**.097**	**.028**	.038	.010
Banana*	**.378**	**.081**	.142	.112	**.324**	**.094**	.095	.110	.084	.040	**.219**	**.165**
Breast Cancer	**.482**	**.131**	.303	.139	**.420**	**.082**	.268	.166	**.417**	**.171**	.111	.198
Bupa	**.235**	**.043**	.077	.030	**.187**	**.051**	.111	.049	**.171**	**.042**	.118	.048
Contraceptive	**.391**	**.052**	.054	.010	**.358**	**.028**	.119	.027	.093	.063	**.099**	**.047**
Heart	**.247**	**.045**	.153	.036	**.277**	**.034**	.109	.045	.094	.048	**.119**	**.037**
Ionosphere	.363	.125	**.381**	**.061**	**.401**	**.060**	.357	.050	.351	.156	**.406**	**.051**
Iris	.204	.060	**.221**	**.049**	.167	.051	**.246**	**.081**	.135	.040	**.223**	**.090**
Monk2	**.428**	**.122**	.091	.071	**.372**	**.120**	.048	.040	**.208**	**.104**	.047	.031
Newthyroid	**.279**	**.071**	.150	.051	**.232**	**.056**	.109	.064	.122	.048	**.312**	**.057**
Phoneme*	**.472**	**.112**	.197	.032	**.399**	**.146**	.074	.053	.127	.102	**.179**	**.068**
Pima	**.382**	**.096**	.136	.082	**.279**	**.106**	.138	.088	.129	.095	**.184**	**.098**
Saheart	**.313**	**.055**	.062	.019	**.344**	**.026**	.054	.022	.083	.030	**.141**	**.022**
Soybean	**.524**	**.165**	.103	.025	**.542**	**.107**	.121	.064	**.245**	**.210**	.051	.050
Spectfheart	.186	.096	**.242**	**.047**	**.194**	**.106**	.194	.046	**.291**	**.117**	.120	.039
Tae	**.262**	**.045**	.067	.021	**.320**	**.029**	.239	.044	.099	.045	**.118**	**.022**
Titanic*	**.433**	**.058**	.069	.023	**.349**	**.032**	.164	.051	.121	.091	**.181**	**.031**
Vowel	**.363**	**.127**	.108	.030	**.315**	**.167**	.122	.072	**.206**	**.117**	.143	.100
Wdbc	.343	.251	**.397**	**.161**	.233	.194	**.335**	**.178**	**.248**	**.279**	.173	.260
Wine	**.345**	**.128**	.164	.051	**.316**	**.129**	.162	.046	.145	.109	**.347**	**.082**
Average	**.344**	**.095**	.158	.054	**.319**	**.082**	.171	.068	**.173**	**.097**	.166	.075

Table 14: Experimental results for the ϵ^+-indicator obtained by ME-MOEA/D$_{CC}$ and 3 previous MO approaches in the CS_{15} constraint set.

Dataset	Hypervolume for CS_{15}								Pareto Approximation Size for CS_{15}							
	ME-MOEA/D$_{CC}$		MOEA/D$_{CC}$		MOCK		PESA-II$_{CC}$		ME-MOEA/D$_{CC}$		MOEA/D$_{CC}$		MOCK		PESA-II$_{CC}$	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	.543	.052	.724	.021	.588	.057	.624	.008	1000.0	0.0	84.6	9.0	995.2	8.1	996.4	7.2
Banana*	.404	.189	.632	.064	.601	.041	.275	.005	21.4	11.7	234.8	33.6	790.0	257.8	132.8	9.8
Breast Cancer	.358	.171	.501	.100	.413	.082	.807	.387	43.6	44.8	59.2	8.5	887.2	151.6	26.2	2.4
Bupa	.471	.083	.758	.004	.606	.023	.548	.035	58.6	12.2	281.6	23.3	999.0	2.0	132.0	20.2
Contraceptive	.364	.061	.866	.020	.589	.005	.358	.043	30.0	6.9	320.4	44.7	999.6	0.8	112.8	12.7
Heart	.464	.060	.615	.045	.598	.024	.403	.057	575.2	222.6	87.6	20.3	995.4	7.3	71.4	7.4
Ionosphere	.452	.098	.477	.032	.535	.016	.344	.031	313.8	108.5	147.6	18.8	998.2	3.1	120.0	13.9
Iris	.625	.100	.620	.021	.575	.115	.492	.065	44.2	13.9	46.2	5.7	998.0	4.0	45.0	3.0
Monk2	.268	.128	.431	.069	.476	.028	.426	.028	173.0	85.4	158.0	29.1	939.4	110.8	94.8	8.3
Newthyroid	.599	.093	.692	.069	.671	.090	.280	.015	30.2	8.2	52.6	11.5	1000.0	0.0	55.6	6.4
Phoneme*	.295	.146	.496	.025	.576	.063	.264	.035	10.4	3.2	72.2	6.8	984.6	22.5	82.6	9.2
Pima	.395	.186	.597	.021	.430	.037	.285	.042	54.6	29.2	152.2	18.9	955.6	88.3	95.6	14.2
Saheart	.387	.020	.545	.031	.562	.035	.314	.019	217.4	42.6	166.4	19.9	823.8	193.4	102.0	14.3
Soybean	.108	.137	.686	.072	.627	.048	.289	.000	1000.0	0.0	982.0	36.0	1000.0	0.0	1000.0	0.0
Spectfheart	.496	.106	.417	.073	.450	.024	.665	.033	456.8	113.8	70.6	23.5	1000.0	0.0	141.8	28.4
Tae	.516	.048	.750	.027	.735	.039	.450	.022	126.2	31.5	135.8	29.3	1000.0	0.0	124.0	15.0
Titanic*	.385	.064	.852	.050	.715	.067	.259	.019	89.4	34.4	88.2	11.7	672.4	61.1	91.8	5.3
Vowel	.454	.202	.882	.014	.674	.008	.536	.020	17.4	5.4	221.6	24.7	993.0	7.9	59.0	6.7
Wdbc	.452	.287	.443	.038	.378	.077	.818	.364	40.0	35.7	66.6	15.3	998.0	4.0	25.6	1.2
Wine	.483	.092	.620	.059	.643	.033	.234	.104	19.8	3.4	60.0	10.0	995.6	8.8	18.0	3.7
Average	.426	.114	.630	.142	.572	.097	.434	.177	216.1	301.6	174.4	200.9	951.2	87.6	176.3	276.3

Table 15: Experimental results for the Hypervolume and the Pareto Approximation Size measures obtained by ME-MOEA/D$_{CC}$ and 3 previous MO approaches in the CS_{15} constraint set.

Dataset	$I_{\epsilon^+}(\cdot,\cdot) : A \leftarrow$ ME-MOEA/D$_{CC}$ for CS_{20}											
	$B \leftarrow$ MOEA/D$_{CC}$				$B \leftarrow$ MOCK				$B \leftarrow$ PESA-II$_{CC}$			
	$I_{\epsilon^+}(A,B)$		$I_{\epsilon^+}(B,A)$		$I_{\epsilon^+}(A,B)$		$I_{\epsilon^+}(B,A)$		$I_{\epsilon^+}(A,B)$		$I_{\epsilon^+}(B,A)$	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**.237**	**.029**	.016	.016	**.380**	**.010**	.258	.113	**.091**	**.031**	.042	.012
Banana*	**.365**	**.120**	.126	.118	**.350**	**.111**	.080	.117	.142	.060	**.238**	**.153**
Breast Cancer	**.640**	**.295**	.297	.206	**.581**	**.194**	.146	.139	**.625**	**.298**	.043	.160
Bupa	**.264**	**.037**	.051	.033	**.239**	**.028**	.142	.045	**.212**	**.045**	.095	.033
Contraceptive	**.448**	**.097**	.052	.012	**.429**	**.073**	.062	.069	.121	.069	**.126**	**.031**
Heart	**.296**	**.028**	.154	.046	**.333**	**.027**	.104	.027	**.113**	**.032**	.076	.022
Ionosphere	.370	.070	**.435**	**.038**	**.434**	**.037**	.416	.028	.377	.073	**.466**	**.034**
Iris	**.484**	**.213**	.259	.119	**.446**	**.226**	.218	.156	**.404**	**.246**	.251	.115
Monk2	**.614**	**.063**	.039	.011	**.524**	**.061**	.045	.041	**.278**	**.094**	.080	.055
Newthyroid	**.283**	**.103**	.237	.076	**.316**	**.165**	.185	.071	.214	.193	**.427**	**.083**
Phoneme*	**.539**	**.116**	.157	.016	**.419**	**.120**	.038	.055	**.191**	**.103**	.103	.089
Pima	**.428**	**.065**	.110	.026	**.298**	**.075**	.098	.044	.122	.058	**.134**	**.043**
Saheart	**.364**	**.035**	.064	.021	**.349**	**.035**	.030	.020	**.126**	**.050**	.102	.042
Soybean	**.382**	**.069**	.183	.079	**.460**	**.066**	.180	.076	.008	.013	**.079**	**.087**
Spectfheart	.202	.011	**.252**	**.055**	**.216**	**.040**	.170	.044	**.273**	**.090**	.121	.055
Tae	**.220**	**.052**	.053	.024	**.296**	**.027**	.274	.025	.077	.027	**.169**	**.067**
Titanic*	**.431**	**.019**	.059	.018	**.383**	**.036**	.212	.037	.086	.040	**.195**	**.046**
Vowel	**.397**	**.053**	.101	.054	**.334**	**.053**	.071	.060	**.297**	**.035**	.109	.077
Wdbc	**.492**	**.108**	.366	.127	**.418**	**.149**	.278	.142	**.606**	**.225**	.000	.000
Wine	**.419**	**.243**	.277	.097	**.427**	**.241**	.247	.067	.327	.282	**.485**	**.144**
Average	**.394**	**.091**	.164	.060	**.382**	**.089**	.163	.069	**.234**	**.103**	.167	.067

Table 16: Experimental results for the ϵ^+-indicator obtained by ME-MOEA/D$_{CC}$ and 3 previous MO approaches in the CS_{20} constraint set.

Dataset	Hypervolume for CS_{20}								Pareto Approximation Size for CS_{20}							
	ME-MOEA/D$_{CC}$		MOEA/D$_{CC}$		MOCK		PESA-II$_{CC}$		ME-MOEA/D$_{CC}$		MOEA/D$_{CC}$		MOCK		PESA-II$_{CC}$	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	.482	.020	.698	.051	.650	.114	.511	.008	1000.0	0.0	100.4	10.4	1000.0	0.0	1000.0	0.0
Banana*	.406	.173	.613	.086	.711	.065	.275	.010	12.8	1.9	241.0	36.9	890.4	219.2	131.0	9.4
Breast Cancer	.268	.271	.441	.066	.418	.063	.800	.400	25.2	33.5	47.4	6.8	873.0	65.2	25.4	0.8
Bupa	.484	.052	.792	.017	.582	.033	.573	.038	46.8	9.0	274.4	23.1	998.4	3.2	128.4	14.3
Contraceptive	.304	.097	.881	.019	.695	.141	.316	.021	29.8	3.1	314.6	49.1	1000.0	0.0	101.8	7.0
Heart	.392	.036	.550	.058	.625	.018	.387	.024	742.2	116.9	99.0	9.7	999.8	0.4	73.4	3.5
Ionosphere	.460	.030	.435	.066	.521	.012	.318	.021	367.6	100.9	129.2	28.0	990.8	11.4	124.6	17.0
Iris	.399	.249	.670	.130	.537	.164	.506	.124	59.0	58.0	45.0	8.1	986.2	18.9	42.6	1.6
Monk2	.118	.032	.522	.014	.424	.025	.280	.088	178.8	56.1	200.4	37.8	995.0	9.0	121.6	26.6
Newthyroid	.653	.145	.690	.051	.715	.081	.230	.035	15.6	6.3	48.4	14.1	999.2	1.6	50.4	11.0
Phoneme*	.236	.107	.509	.052	.564	.036	.277	.022	9.6	5.1	74.2	14.0	968.6	38.0	83.8	3.4
Pima	.293	.054	.600	.031	.423	.036	.257	.038	62.6	27.8	150.8	26.0	999.2	1.1	98.4	12.1
Saheart	.322	.057	.579	.088	.566	.067	.325	.039	302.0	96.3	140.6	17.1	816.2	163.6	117.8	5.3
Soybean	.420	.101	.623	.145	.674	.057	.339	.000	1000.0	0.0	981.6	36.8	999.4	1.2	1000.0	0.0
Spectfheart	.476	.075	.333	.045	.469	.042	.609	.025	708.8	109.8	95.0	8.7	999.4	0.8	150.6	21.9
Tae	.570	.059	.771	.036	.714	.032	.410	.048	145.0	40.4	138.8	31.8	1000.0	0.0	147.2	29.0
Titanic*	.376	.061	.848	.022	.656	.067	.242	.026	59.6	21.0	85.4	18.3	730.0	97.6	111.8	7.3
Vowel	.363	.133	.843	.056	.705	.029	.573	.047	22.0	5.6	248.4	54.4	994.0	8.0	49.2	7.5
Wdbc	.296	.214	.402	.057	.443	.079	.000	.000	50.8	42.3	56.0	15.3	834.0	99.8	25.0	0.0
Wine	.405	.224	.644	.039	.623	.051	.187	.111	14.6	8.1	42.4	9.9	997.2	5.6	15.8	2.3
Average	.386	.117	.622	.152	.586	.104	.421	.203	242.6	329.6	175.6	201.5	953.5	77.6	179.9	276.3

Table 17: Experimental results for the Hypervolume and the Pareto Approximation Size measures obtained by ME-MOEA/D$_{CC}$ and 3 previous MO approaches in the CS_{20} constraint set.

References

[1] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[2] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[3] Baptiste Lafabregue, Jonathan Weber, Pierre Gançarski, and Germain Forestier. Deep constrained clustering applied to satellite image time series. In *ECML/PKDD Workshop on Machine Learning for Earth Observation Data (MACLEAN)*, Würzburg, Germany, 2019.

[4] Chao-Lung Yang and Thi Phuong Quyen Nguyen. Constrained clustering method for class-based storage location assignment in warehouse. *Industrial Management & Data Systems*, 116(4):667–689, 2016.

[5] Son T Mai, Sihem Amer-Yahia, Sébastien Bailly, Jean-Louis Pépin, Ahlame Douzal Chouakria, Ky T Nguyen, and Anh-Duong Nguyen. Evolutionary active constrained clustering for obstructive sleep apnea analysis. *Data Science and Engineering*, 3(4):359–378, 2018.

[6] Andreas Brieden, Peter Gritzmann, and Fabian Klemm. Constrained clustering via diagrams: A unified theory and its application to electoral district design. *European Journal of Operational Research*, 263(1):18–34, 2017.

[7] Firas Saidi, Zouheir Trabelsi, and Henda Ben Ghazela. A novel approach for terrorist sub-communities detection based on constrained evidential clustering. In *2018 12th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–8. IEEE, 2018.

[8] MA Balafar, R Hazratgholizadeh, and MRF Derakhshi. Active learning for constrained document clustering with uncertainty region. *Complexity*, 2020, 2020.

[9] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 577–584. Morgan Kaufmann Publishers Inc., 2001.

[10] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, MSR-TR-2000-65, Microsoft Research, 2000.

[11] Ian Davidson and Sugato Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from data*, 1:1–41, 2007.

[12] Jana Schmidt, Elisabeth Maria Brandle, and Stefan Kramer. Clustering with attribute-level constraints. In *2011 IEEE 11th International Conference on Data Mining*, pages 1206–1211. IEEE, 2011.

[13] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 521–528, 2003.

[14] Dit-Yan Yeung and Hong Chang. A kernel approach for semisupervised metric learning. *IEEE Transactions on Neural Networks*, 18(1):141–149, 2007.

[15] Xuesong Yin, Songcan Chen, Enliang Hu, and Daoqiang Zhang. Semi-supervised clustering with metric learning: An adaptive kernel method. *Pattern Recognition*, 43(4):1320–1333, 2010.

[16] Ian Davidson and SS Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 59–70. Springer, 2005.

[17] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. C-dbscan: Density-based clustering with constraints. In *International workshop on rough sets, fuzzy sets, data mining, and granular-soft computing*, pages 216–223. Springer, 2007.

[18] Martin HC Law, Alexander Topchy, and Anil K Jain. Clustering with soft and group constraints. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 662–670. Springer, 2004.

[19] Ian Davidson and SS Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 International Conference on Data Mining*, pages 138–149. SIAM, 2005.

[20] Seçkin Karasu, Aytaç Altan, Stelios Bekiros, and Wasim Ahmad. A new forecasting model with wrapper-based feature selection approach using multi-objective optimization technique for chaotic crude oil time series. *Energy*, 212:118750, 2020.

[21] Aytaç Altan and Seçkin Karasu. Recognition of covid-19 disease from x-ray images by hybrid model consisting of 2d curvelet transform, chaotic salp swarm algorithm and deep learning technique. *Chaos, Solitons & Fractals*, 140:110071, 2020.

[22] Aytaç Altan, Seçkin Karasu, and Stelios Bekiros. Digital currency forecasting with chaotic meta-heuristic bio-inspired signal processing techniques. *Chaos, Solitons & Fractals*, 126:325–336, 2019.

[23] Aytaç Altan. Performance of metaheuristic optimization algorithms based on swarm intelligence in attitude and altitude control of unmanned aerial vehicle for path following. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–6. IEEE, 2020.

[24] Satyasai Jagannath Nanda and Ganapati Panda. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary computation*, 16:1–18, 2014.

[25] Lijun He, Wenfeng Li, Yu Zhang, and Yulian Cao. A discrete multi-objective fireworks algorithm for flowshop scheduling with sequence-dependent setup times. *Swarm and Evolutionary Computation*, 51:100575, 2019.

[26] Adán José-García and Wilfrido Gómez-Flores. Automatic clustering using nature-inspired metaheuristics: A survey. *Applied Soft Computing*, 41:192–213, 2016.

[27] Weiguo Sheng, Stephen Swift, Leishi Zhang, and Xiaohui Liu. A weighted sum validity function for clustering with a hybrid niching genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(6):1156–1167, 2005.

[28] Javier Del Ser, Eneko Osaba, Daniel Molina, Xin-She Yang, Sancho Salcedo-Sanz, David Camacho, Swagatam Das, Ponnuthurai N Suganthan, Carlos A Coello Coello, and Francisco Herrera. Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation*, 48:220–250, 2019.

[29] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2013.

[30] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[31] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

[32] David W Corne, Nick R Jerram, Joshua D Knowles, and Martin J Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 283–290, 2001.

[33] A novel multi-objective genetic algorithm based error correcting output codes. *Swarm and Evolutionary Computation*, 57:100709, 2020.

[34] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos Artemio Coello Coello. A survey of multiobjective evolutionary algorithms for data mining: Part i. *IEEE Transactions on Evolutionary Computation*, 18(1):4–19, 2013.

[35] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos A Coello Coello. Survey of multiobjective evolutionary algorithms for data mining: Part ii. *IEEE Transactions on Evolutionary Computation*, 18(1):20–35, 2013.

[36] Lisong Wang, Guonan Cui, Qing Zhou, and Kui Li. A multi-clustering method based on evolutionary multiobjective optimization with grid decomposition. *Swarm and Evolutionary Computation*, page 100691, 2020.

[37] Julia Handl and Joshua Knowles. On semi-supervised clustering via multiobjective optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1465–1472, 2006.

[38] Juanjuan Luo, Licheng Jiao, and Jose A Lozano. A sparse spectral clustering framework via multiobjective evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, 20(3):418–433, 2015.

[39] Zahra Donyavi and Shahrokh Asadi. Using decomposition-based multi-objective evolutionary algorithm as synthetic example optimization for self-labeling. *Swarm and Evolutionary Computation*, page 100736, 2020.

[40] Ferrante Neri, Carlos Cotta, and Pablo Moscato. *Handbook of memetic algorithms*, volume 379. Springer, 2011.

[41] Pablo Moscato and Carlos Cotta. A gentle introduction to memetic algorithms. In *Handbook of metaheuristics*, pages 105–144. Springer, 2003.

[42] Francia Jiménez, Claudio Sanhueza, Regina Berretta, and Pablo Moscato. Accelerating a multi-objective memetic algorithm for feature selection using hierarchical k-means indexes. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 181–182, 2018.

[43] Bilel Derbel, Arnaud Liefooghe, Qingfu Zhang, Sébastien Verel, Hernan Aguirre, and Kiyoshi Tanaka. A set-oriented MOEA/D. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 617–624, 2018.

[44] Weiqin Ying, Yuehong Xie, Yu Wu, Bingshen Wu, Shiyun Chen, and Weipeng He. Universal partially evolved parallelization of MOEA/D for multi-objective optimization on message-passing clusters. *Soft Computing*, 21(18):5399–5412, 2017.

[45] Alexandre Sawczuk Da Silva, Hui Ma, Yi Mei, and Mengjie Zhang. A hybrid memetic approach for fully automated multi-objective web service composition. In *2018 IEEE International Conference on Web Services (ICWS)*, pages 26–33. IEEE, 2018.

[46] Ahmad Alhindi, Abrar Alhindi, Atif Alhejali, Abdullah Alsheddy, Nasser Tairan, and Hosam Alhakami. MOEA/D-GLS: a multiobjective memetic algorithm using decomposition and guided local search. *Soft Computing*, 23(19):9605–9615, 2019.

[47] Germán González-Almagro, Alejandro Rosales-Pérez, Julián Luengo, José-Ramón Cano, and Salvador García. Improving constrained clustering via decomposition-based multiobjective optimization with memetic elitism. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 333–341, 2020.

[48] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[49] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[50] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2014.

[51] Indraneel Das and John E Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM journal on optimization*, 8(3):631–657, 1998.

[52] Achille Messac, Amir Ismail-Yahaya, and Christopher A Mattson. The normalized normal constraint method for generating the pareto frontier. *Structural and multidisciplinary optimization*, 25(2):86–98, 2003.

[53] Alvaro Garcia-Piquer, Albert Fornells, Jaume Bacardit, Albert Orriols-Puig, and Elisabet Golobardes. Large-scale experimental evaluation of cluster representations for multiobjective evolutionary clustering. *IEEE Transactions on Evolutionary Computation*, 18(1):36–53, 2013.

[54] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):224–227, 1979.

[55] Julia Handl and Joshua Knowles. Exploiting the trade-off: the benefits of multiple objectives in data clustering. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 547–560. Springer, 2005.

[56] Joshua Knowles and David Corne. Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Transactions on Evolutionary Computation*, 7(2):100–116, 2003.

[57] Ryoji Tanabe and Hisao Ishibuchi. A review of evolutionary multimodal multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 24(1):193–200, 2019.

[58] Hui Li, Min Ding, Jingda Deng, and Qingfu Zhang. On the use of random weights in MOEA/D. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 978–985. IEEE, 2015.

[59] Isaac Triguero, Sergio González, Jose M Moyano, Salvador García, Jesús Alcalá-Fdez, Julián Luengo, Alberto Fernández, Maria José del Jesús, Luciano Sánchez, and Francisco Herrera. KEEL 3.0: an open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems*, 10(1):1238–1249, 2017.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[61] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[62] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[63] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[64] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

[65] Nicola Beume, Carlos M Fonseca, Manuel López-Ibáñez, Luís Paquete, and Jan Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082, 2009.

[66] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[67] Jacinto Carrasco, Salvador García, María del Mar Rueda, and Francisco Herrera. rnpbst: An r package covering non-parametric and bayesian statistical tests. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 281–292. Springer, 2017.

[68] Jacinto Carrasco, Salvador García, MM Rueda, S Das, and Francisco Herrera. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, 54:100665, 2020.

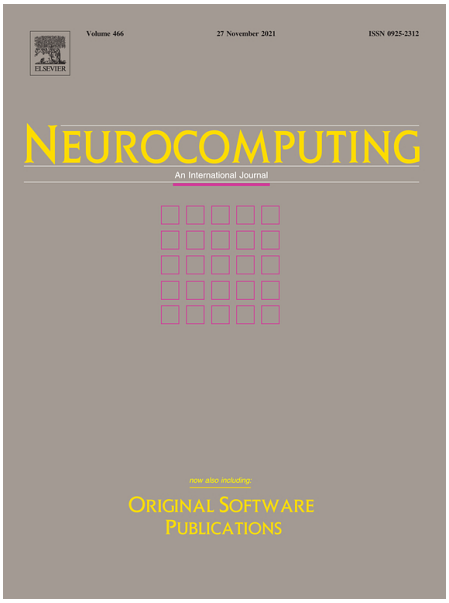
[69] Germán González-Almagro, Julián Luengo, José-Ramón Cano, and Salvador García. Enhancing instance-level constrained clustering through differential evolution. *Applied Soft Computing*, page 107435, 2021.

[70] Dan Pelleg and Dorit Baras. K-means with large and noisy constraint sets. In *European Conference on Machine Learning*, pages 674–682. Springer, 2007.

[71] Daniel Khashabi, John Wieting, Jeffrey Yufei Liu, and Feng Liang. Clustering with side information: From a probabilistic model to a deterministic algorithm. *arXiv preprint arXiv:1508.06235*, 2015.

[72] Tamara Ulrich, Dimo Brockhoff, and Eckart Zitzler. Pattern identification in pareto-set approximations. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 737–744, 2008.

4 3SHACC: Three Stages Hybrid Agglomerative Constrained Clustering



- **Journal:** Neurocomputing (NEUCOM)
- **JCR Impact Factor:** 5.779
- **Rank:** 39/145
- **Quartile:** Q2
- **Category:** Computer Science, Artificial Intelligence
- **Status:** Published

Ref.: Gonzalez-Almagro, G., Suárez, J. L., Luengo, J., Cano, J. R., & García, S. (2022). 3SHACC: Three stages hybrid agglomerative constrained clustering. Neurocomputing, 490, 441-461. DOI: https://doi.org/10.1016/j.neucom.2021.12.018.

A preliminary version of this paper was published at the 2020 Hybrid Artificial Intelligence Conference (HAIS) with title: Agglomerative Constrained Clustering Through Similarity and Distance Recalculation (DOI: https://doi.org/10.1007/978-3-030-61705-9_35). This conference was held thematically due to the COVID-19 pandemic.

3SHACC: Three Stages Hybrid Agglomerative Constrained Clustering

Germán González Almagro [*,a,b] **Juan Luis Suárez** [,a,b] **Julián Luengo** [a,b]

José-Ramón Cano [c,b] **Salvador García** [a,b]

[a] *Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada, Granada, Spain*
[b] *Andalusian Institute of Data Science and Computational Intelligence (DaSCI)*
[c] *Department of Computer Science, University of Jaén, Jaén, Spain*

Abstract

Traditionally within the unsupervised learning paradigm, hierarchical and partitional clustering techniques have been shown to produce better results when provided with partial information, leading to a renewed attention towards this topic. Constrained clustering is a semi-supervised learning problem that combines classic clustering techniques with background knowledge given in the form of a set of constraints. In this paper, we propose to incorporate constraints into the clustering process in three phases: the first phase is devoted to quantify constraint relevance and to learn a metric matrix according to such relevance, a second phase computing similarities between instances by means of the reconstruction coefficient and pairwise distances, and a third stage performing agglomerative hierarchical clustering with a reward-style stepped affinity function favoring merges satisfying the higher possible number of constraints. Experimental results, supported by Bayesian statistical testing, show a consistent improvement in favor of our proposal over previous approaches to the constrained clustering problem.

Keywords Constrained Clustering · Instance-level Constraints · Constraint Relevance Quantification · Weighted Distance Metric Learning.

* Corresponding Author (`germangalmagro@ugr.es`)
Email addresses: `jlsuarezdiaz@ugr.es` (Juan Luis Suárez), `julianlm@decsai.ugr.es` (Julián Luengo), `jrcano@ujaen.es` (José Ramón Cano), `salvagl@decsai.ugr.es` (Salvador García)

1 Introduction

Clustering has always been a key research area in machine learning. It is able to provide valuable insight within the unsupervised learning paradigm, where no information other than an unlabeled dataset by itself is available for learning. However, background knowledge can be added to the classic clustering framework, leading to better and more accurate results and moving the problem from the unsupervised learning framework to the machine learning paradigm known as Semi-Supervised Learning (SSL) [1, 2]. When the background knowledge is given in the form or constraints, the problem is called Constrained Clustering (CC). In CC, a set of constraints is used to guide the clustering process, as the resulting partition of the dataset is required to satisfy as many constraints as possible, in addition to meeting the proper characteristics of a classic clustering partition. CC has been successfully applied in many fields of knowledge, among which it is worth mentioning: satellite image time series [3], storage location assignment in warehouses [4], obstructive sleep apnea analysis [5], electoral district design [6], lane finding in GPS data [7] and classic benchmark applications [8, 9].

Many distinctions can be made within the general CC framework. Three main ways to include constraints into the clustering problem can be found in the literature: cluster-level [10], instance-level [11] and feature-level CC [12]. CC algorithms commonly use two main strategies to take constraints into account: (1) in *distance-based* methods a new metric reflecting the information contained in the constraint set is learned [13, 9, 14], (2) in *clustering-engine* adapting constraints are used as hints to guide the clustering process by modifying the clustering engine to include them [7, 15, 16, 17]. Finally, the concepts of soft [18] and hard [7] constraints can also be found in the literature. With a hard interpretation of constraints, methods are forced to output partitions satisfying all constraints, whereas soft constraints allow output partitions with some unsatisfied constraints. This study is focused on soft instance-level Must-link (ML) and Cannot-link (CL) constraints, which tell us if two specific instances of a dataset must be placed in the same or in different clusters, respectively. Soft constraints pose several general advantages over hard constraints, such as noise resilience and less computational complexity, while maintaining the advantages considering background knowledge. A soft interpretation of constraints does not only brings the advantages mentioned above, but also avoids specific problems related to hierarchical CC, such as not being able to always provide a full dendrogram and reaching dead-ends in the dendrogram-building process, as it will be explained in detail in Section 2.3.

Using ML and CL constraints makes the CC problem **NP**-complete [19]. This is the reason why approximate methods represent a promising approach to the CC problem. Within the classic clustering paradigm, two broad categories can be found in the literature: partitional clustering and hierarchical clustering. In partitional clustering, a partition assigning every instance from the dataset to a specific cluster from among a fixed number of them is built, while hierarchical clustering obtains a tree-like hierarchical structure coding a set of partitions that allows the user to choose any cluster granularity between one and the number of instances in the dataset. Both of them have been applied to many real-world problems [20],

although when it comes to CC a significant imbalance favoring partitional methods can be observed; very little work has been done to integrate constraints into hierarchical clustering methods [15, 21, 22, 23] with respect to the number of existing partitional CC methods, whose analysis would require the length of a monograph. Conceptually different approaches to the CC problem have also been proposed, specially within the metaheuristics field, with methods ranging from classic metaheuristic approaches [24, 8] to innovative iterated local search variants [25] or evolutionary multiobjective optimization [26, 27].

In this study we aim to investigate the use of hybrid agglomerative hierarchical clustering methods for the CC problem, which should combine distance-based techniques and clustering-engine adapting techniques to address the CC problem. Weighted constraints can be used to guide the clustering process towards high quality solutions more effectively than unweighted constraints, although very little work has been done on automatically generating constraint weights and on integrating them into distance-based CC methods. This study tackles these open problems. To do so the Three Stages Hybrid Agglomerative Hierarchical Constrained Clustering (3SHACC) method is proposed. It is a CC method capable of, given a dataset and a constraint set, obtaining a full dendrogram capturing the fine-grain manifold structures present in the dataset and integrating constraints into the process. To do so, it implements three well-defined stages:

1. In the first stage, the relevance of every constraint is determined and a new metric is built on the basis of the newly weighted constraint set by using a new Distance Metric Learning (DML) algorithm, which we call Weighted-Learning from Side Information (WLSI). The result of this stage is a metric matrix that allows to measure distances within the dataset having into account the information contained in the constraint set.

2. The second stage computes similarities among instances in the dataset attending to the newly computed distance metric and the pairwise reconstruction coefficient. The advantages of using both of these measures to determine pairwise similarities include robustness to noise and less sensitivity to outliers [28]. Overall, this stage results in a symmetric matrix containing pairwise similarities, which is later used to determine affinities between clusters and to build a dendrogram. This is the only stage in 3SHACC provided with tools to handle noise and outliers, which are inherited from the method proposed in [28], where these capabilities are experimentally proved.

3. Finally, a dendrogram is obtained by running a classic Agglomerative Hierarchical Clustering (AHC) method with a constraint-biased stepped affinity function integrating the computed similarities and the information contained in the constraint set. Once again constraints are used to guide the clustering process by using them to influence the clusters merge selection procedure, which is considered to be the clustering-engine of any AHC method and therefore resulting in a directly constraint-influenced dendrogram.

3SHACC mixes two constraint integration models: it computes a new metric on the basis of constraints which is used to compute pairwise similarities, and it adapts a classic clustering engine to consider constraints when assigning particular instances to clusters. For this reason, it should be considered a hybrid method. Our overall proposal extends the work by [29], where a basic AHC approach to the CC problem was proposed. Even if the basic drawlines presented in [29] are maintained, 3SHACC offers a mainly new approach to the CC problem. Automatic constraint weighting and weighted DML are exclusive to the proposal presented in this study, as in [29] the only step previous to the clustering process is a simple constraint propagation procedure using the Floyd-Warshall algorithm. Another difference with respect to [29] can be found in the agglomerative hierarchical CC procedure. In this study, a constraint-influenced affinity function is always applied to select the two clusters to merge, whereas in [29] a constraint-influenced affinity function can only be applied under very specific circumstances. Overall, 3SHACC presents several advantages over previous approaches:

- A completely unsupervised constraint weighting method based on classic clustering techniques is used to generate constraint weights.

- A new semi-supervised DML algorithm that is able to leverage both constraints and their weights in order to learn a distance metric, in contrast to classic DML algorithms which are incapable of handling weights.

- A reward-style cost function that accounts for satisfied constraints is used to apply AHC to the dataset and the constraint set, based on the similarities computed on the basis of the transformed dataset. This cost function allows for a strong influence of constraints on the dendrogram building process and while avoiding dead-ends, as it allows constraint violations when needed.

To prove the suitability and adaptability of 3SHACC for diverse instances of the CC problem, 3SHACC is tested over 25 datasets and compared with five classic approaches to the CC problem and the previous version of 3SHACC presented in [29], which we refer to as 2SHACC. Three constraint sets are generated for every dataset, with incremental level of constraints-based information, allowing for a scalability comparison in the quality of the results. Statistical Bayesian testing is later used to validate comparisons and draw our final conclusions.

The rest of this paper is organized as follows: background related to CC and multiobjective optimization is introduced in Section 2 and our proposal 3SHACC is described in Section 3. The experimental setup is explained in Section 4; results and their analysis are discussed in Sections 5 and 6, respectively; finally, conclusions are presented in Section 7.

2 Background

In this section, we present the background knowledge concerning AHC (Section 2.1) and CC (Section 2.2), as well as computational complexity results arising from approaching the CC problems via AHC (Section 2.3).

2.1 Agglomerative Clustering

Clustering can be defined as the task of grouping the n instances of a dataset X into k clusters. Each instance is described by u features. More formally, $X = \{x_1, \cdots, x_n\}$, with the ith instance noted as $x_i = (x_i^1, \cdots, x_i^u)$. A partition $C = \{c_1, \cdots, c_K\}$ assigns each instance to a cluster such that $c_i = \{x_i^1, \cdots, x_i^{|c_i|} | x_j \in X, j = 1, \cdots, |c_i|\}$. All instances need to be assigned to a cluster ($\bigcup_{i=0}^{K} c_i = X$) and no instance can be assigned to more than one cluster ($c_i \cap c_j = \varnothing \ \forall i \neq j$). Every partition C assigns a label y_i^C to every x_i such that $y_i^C = l \iff x_i \in c_l$. As a result, the list of labels for a given partition $Y_C = \{y_1^C, \cdots, y_n^C\}$, with $y_i^C \in \{1, \cdots, k\}$, is obtained. Assigning an instance to a cluster depends on the similarity to the rest of elements in that cluster, and the dissimilarity to the rest of instances of the dataset. The similarity between two instances can be obtained with some kind of distance measurement [30].

AHC methods produce an informative hierarchical structure of clusters called dendrogram. Partitions as describe above, with a number of clusters ranging from 1 to n, can always be obtained from a dendrogram by just selecting a level from its hierarchy and partitioning the dataset according to its structure. Typically, AHC methods start with a large number of clusters and iteratively merge them according to some affinity criteria until a stopping condition is reached. Every merge produces a new level in the hierarchy of the dendrogram. Formally, given an initial partition with n_c clusters $C = \{c_1, \cdots, c_{n_c}\}$ (usually $n_c = n$), a traditional agglomerative CC method selects two clusters to merge by applying Equation 1.

$$\{c_i, c_j\} = \underset{c_i, c_j \in C, i \neq j}{\operatorname{argmax}} A(c_i, c_j), \tag{1}$$

with $A(\cdot, \cdot)$ being a function used to determine the affinity between the two clusters given as arguments. This function needs to be carefully chosen for every problem, as it greatly affects the result of the clustering process. Some conventional methods to measure affinity between clusters are worth mentioning, such as single linkage, average linkage and complete linkage [30]. However, different measures are used in out-of-lab applications, as the manifold structures usually present in real-world datasets can be hardly captured by the aforementioned classic affinity measures.

To tackle this problem, new AHC algorithms operate on the basis of a structure known as similarity graph, where nodes represent instances in dataset X and weighted edges represent similarities between those instances. Methods such as Chameleon [31], GDL [32] or PIC [33] use similarity graphs to perform AHC. The K-nearest-neighbors (K-NN) graph is commonly

used by AHC methods as the similarity graph, it is typically built as in Equation 2, where $\mathcal{N}_i^K$ is the set of K-nearest neighbors of x_i and ρ is a tradeoff parameter.

$$S_{[i,j]} = \begin{cases} e^{(-\frac{\|x_i - x_j\|_2^2}{\rho})} & x_j \in \mathcal{N}_i^K \\ 0 & \text{otherwise} \end{cases} . \tag{2}$$

2.2 Constrained Clustering

In instance-level Constrained Clustering (CC), a set of constraints is given to guide the clustering process. Constraints consist of pairs of instances that state whether the two instances must be placed in the same cluster or in different clusters. Better results can be obtained by integrating these constraints into the clustering process. Constraints can be formalized as follows:

- Must-link (ML) constraints $(x_i, x_j) \in C_=$: instances x_i and x_j from X must be placed in the same cluster.
- Cannot-link (CL) constraints $(x_i, x_j) \in C_{\neq}$: instances x_i and x_j from X cannot be assigned to the same cluster.

In CC, a partition of k clusters $C = \{c_1, \cdots, c_k\}$ must be found given a dataset X and two constraint sets $C_=$ and $C_{\neq}$.

ML constraints are reflexive, transitive and symmetric, and therefore they constitute an equivalence relation. Given constraints $(x_a, x_b) \in C_=$ and $(x_b, x_c) \in C_=$ then $(x_a, x_c) \in C_=$ is verified. In addition, if $x_a \in c_i$ and $x_b \in c_j$ are related by $(x_a, x_b) \in C_=$, then $(x_c, x_d) \in C_=$ is verified for any $x_c \in c_i$ and $x_d \in c_j$ [11].

CL constraints do not constitute an equivalence relation. However, analogously, given $x_a \in c_i$ and $x_b \in c_j$, and the constraint $(x_a, x_b) \in C_{\neq}$, then it is also true that $(x_c, x_d) \in C_{\neq}$ for any $x_c \in c_i$ and $x_d \in c_j$ [11].

2.3 The Feasibility Problem

A relevant aspect to consider is how constraints affect the complexity of the classic clustering problem. We can formulate the feasibility problem for hierarchical instance-level CC as in Definition 1 [19].

Definition 1 ***Feasibility Problem****: given a dataset X, the constraint sets $C_{\neq}$ and $C_=$, and the symmetric distance measure $D(x_i, x_j) \geq 0$ for each pair of instances: Can X be partitioned into clusters so that all constraints in $C_{\neq} \cup C_=$ are satisfied? [19]*

Table 1 shows the algorithmic complexity for the feasibility problem for instance-level constraints. Please note that Definition 1 does not impose any limits on the value of K (the

number of clusters), which leads to more relaxed complexities than those found for the non-hierarchical case. However, the dead-ends problem arises: a hierarchical CC algorithm may find scenarios where no merge can be carried out without violating a constraint. Previous solutions based on the transitive closure of the constraint sets have been proposed to this problem, although they imply not generating a full dendrogram [19].

Constraints	Complexity	Dead Ends?
Must-Link	**P**	No
Cannot-Link	**NP**-complete	Yes
ML and CL	**NP**-complete	Yes

Table 1: Feasibility problem complexity [19].

These complexity results show that the feasibility problem with CL constraints is intractable and hence CC is intractable too. This can be proven by using a reduction from the One-in-three 3SAT with positive literals problem, which is **NP**-complete [34]. For more details on the complexity of CC see [19].

Regarding the dead-ends problem, a full dendrogram considering constraints can be obtained by switching from a hard interpretation of constraints to a soft one. This means that every level in the dendrogram tries to satisfy as many constraints as possible, but constraint violations are allowed in order for the algorithm to never reach a dead-end. This is the main reason why we focus on soft constraints in this study.

2.4 Distance Metric Learning

The vast majority of methods making up the overall data science algorithms and techniques corpus use distance measures. They are used to determine similarities between instances in the dataset from which we want to extract information. Clustering can be found among these techniques, with the assignation rule from the k-means algorithm being the foundation of the automatic clustering concept [35]. However, there is an infinite number of distance measures that can be used for this task, and finding the one that better adapts our dataset is crucial to obtain high quality results in any application. Distance Metric Learning (DML) arises to meet this need, with algorithms capable of finding distance metrics that capture hidden features or relations in our datasets that standard measures like the Euclidean distance could miss. Combining DML algorithms and distance-based learning algorithms results in more complete and adaptive approaches to a wide variety of problems [36].

One of the techniques that has helped developing DML is known as Learning from Side Information (LSI), sometimes also referred to as Mahalanobis Metric for Clustering [13]. It directly connects with the SSL paradigm, particularly with the constraint-based SSL area, as it incorporates side information referring to similar and dissimilar pairs of instances in the dataset, which can be easily compared with the must-link and cannot-link constraint sets. Given a pair of examples x_i and x_j, LSI can be viewed as a method to bring these instances

closer if they are similar ($(x_i, x_j) \in C_=$) or set them apart if they are dissimilar ($(x_i, x_j) \in C_{\neq}$). Formally, LSI searchs for a positive semidefinite matrix $M \in S_d(\mathbb{R})_0^+$ optimizing Equation 3.

$$\begin{array}{ll} \min_M & \sum_{(x_i,x_j)\in C_=} \|x_i - x_j\|_M^2 \\ s.t.: & \sum_{(x_i,x_j)\in C_{\neq}} \|x_i - x_j\|_M \geq 1 \end{array} . \tag{3}$$

where $\|x_i - x_j\|_M = \sqrt{(x_i - x_j)^T M (x_i - x_j)}$. However, Equation 3 is hard to optimize with traditional methods, so its authors propose an Equivalent form of it in Equation 4, which can be optimized using the projected gradient ascent method.

$$\begin{array}{ll} \max_M & \sum_{(x_i,x_j)\in C_{\neq}} \|x_i - x_j\|_M \\ s.t.: & \sum_{(x_i,x_j)\in C_=} \|x_i - x_j\|_M^2 \leq 1 \end{array} . \tag{4}$$

3 The Proposed Method: 3SHACC

In this section, we describe the Three Stages Hybrid Agglomerative Constrained Clustering (3SHACC) method. It uses the two constraint integration methods mentioned in the introduction (distance-based methods and clustering engine-adapting methods) to produce a partition of a dataset X taking into account the constraint sets $C_=$ and $C_{\neq}$. Three well-defined phases shape 3SHACC: the first phase is fully devoted to producing a new metric based on the information contained in the constraint sets; the second phase computes a similarity matrix based on the results of the first phase, which is later used in the third phase to finally produce a partition of the dataset through agglomerative clustering. These three phases are extensively discussed below.

3.1 Informativity and Metric Matrix Computation

In order for the new metric to take into account the most relevant information contained in the constraint sets $C_=$ and $C_{\neq}$, the informativity matrix $W \in \mathbb{R}^{n\times n}$ can be computed with respect to any classic clustering procedure $\mathcal{A}$. To do so, the set of partitions $P = [p_1, \cdots, p_\gamma]$ has to be obtained by running $\mathcal{A}$ over X γ times. With this, every partition in P is defined as in Section 2.1 with l_i being the label set associated with partition p_i and given in the form of any finite set. The list of label sets is defined as $L = [l_1, \cdots, l_\gamma]$ with $|l_i| \leq |l_j|$ $\forall i < j$. Then, every element $w_{[i,j]}$ from W is computed as in Equation 5.

$$w_{[i,j]} = \begin{cases} \frac{1}{|P|} \sum_{q=1}^{\gamma} \mathbb{1}[\![y_i^{p_q} \neq y_j^{p_q}]\!] r_{[\gamma-k+1]} & (x_i, x_j) \in C_= \\ \frac{1}{|P|} \sum_{q=1}^{\gamma} \mathbb{1}[\![y_i^{p_q} = y_j^{p_q}]\!] r_k & (x_i, x_j) \in C_{\neq} \\ 0 & \text{otherwise} \end{cases} , \tag{5}$$

where $\mathbb{1}[\![\cdot]\!]$ is the indicator function (returns 1 if the predicate given as argument holds and 0 otherwise), and $r = [r_1, \cdots, r_q |\ 0 < r_i \leq 1, i = 1, \cdots, q]$ is a weighting array with $r_i \leq r_j$ $\forall i < j$. Intuitively, W is used to quantify the relevance of every individual constraint under the

premise that the higher (lower) the number of clusters available to partition a dataset, the harder it is to violate CL (ML) constraints. This trend can be easily verified by observing the trivial cases: when $k = 1$ all ML constraints must be satisfied and all CL constraints are violated, whereas when $k = n$ all CL constraints are satisfied and all ML constraints are violated. Bearing this in mind, Equation 5 can be interpreted as a way to emphasize ML constraints that are violated when $|l_i|$ is low (first partitions in P), and CL constraints that are violated when $|l_i|$ is high (last partitions in P), as they clearly contain valuable information that must have an effect on the clustering process. On the other hand, constraints that are never violated are likely to not contain overall valuable information and their weight would tend to 0. It should be noted that the procedure described above is completely unsupervised, since not even the number of clusters k in the final partition need be known.

Note that W is always computed with respect to a specific classic clustering algorithm $\mathcal{A}$, and hence the selection of $\mathcal{A}$ is crucial to the results of this procedure. The intuition on this lies on identifying the strengths and weaknesses of the algorithm that will be used to obtain the final partition. For example, if a procedure uses the Euclidean distance to determine cluster memberships, clusters will tend to be hyper-spherical, and constraints must be used to guide the procedure away from this trend. To this end, any clustering algorithm using Euclidean distance (such as classic K-means) can be used as $\mathcal{A}$ to obtain W, as constraints that keep clusters away from the hyper-spherical shape are very likely to have a high weight.

Once the weights matrix W has been obtained, a new metric $M \in \mathbb{R}^{u \times u}$ is computed. It is computed by combining the information contained in the dataset X and the constraint sets $C_=$ and $C_{\neq}$, taking into account constraint weights stored in W. In order to do so, a new distance metric learning method based on the classic LSI algorithm [13] is proposed. As it has been mentioned in Section 2.4, the LSI algorithm is capable of integrating similarity information into classic DML procedures, producing as result a new metric that brings similar instances from X (related by ML) closer and separates dissimilar instances (related by CL). However, no information on the relevance of these similarities can be handled by LSI, so it cannot take advantage of the information stored in W. To tackle this issue we propose the Weighted LSI (WLSI) method, which is able to integrate constraint weights into the DML process by optimizing Equation 6.

$$\begin{array}{ll} \max_M & \sum_{(x_i,x_j)\in C_{\neq}} (1 + \varepsilon - w_{[i,j]}) \|x_i - x_j\|_M \\ s.t.: & \sum_{(x_i,x_j)\in C_=} (w_{[i,j]} + \varepsilon) \|x_i - x_j\|_M^2 \leq 1 \end{array} , \tag{6}$$

where M is the resulting metric and $\varepsilon > 0$ is the minimum weight a constraint can have. Intuitively, Equation 6 scales down distances separating instances related by relevant (high-weighted) CL constraints, so that the transformation to be applied needs to emphasize on these constraints to optimize Equation 6. At the same time, it separates instances related by relevant ML constraints so the space transformation needed to bring them together is larger. Please note that ε must be greater than zero in order for instances (x_i, x_j) related by highly relevant CL constraint ($w_{[i,j]} = 1$) to not collapse in the same space point (scaled distance between them $(1 + \varepsilon - w_{[i,j]})\|x_i - x_j\|_M$ becomes 0), in which case Equation 6 can

not be optimized. The iterated projections method [36] can be used to optimize Equation 6. Algorithm 1 summarizes the overall weights and metric matrix computation process.

Algorithm 1: Weights and Metric Matrix Computation (First Stage)

Input: Dataset X, constraint sets $C_{=}$ and $C_{\neq}$, weighting array r, the list of label sets L, number of partitions γ to be obtained with $\mathcal{A}$, minimum weight a constraint can have ε.

[1] Initialize $W \in \mathbb{R}^{n \times n}$ as $W = 0$
[2] Initialize P as an empty list with length γ
`// Build the set of partitions`
[3] **for** $i \in \{i, \cdots, \gamma\}$ **do**
[4] $\quad p_i \leftarrow \mathcal{A}(X, l_i)$
[5] **end**
`/* Iterate over` P `to build the weights matrix` W `following Equation 5 */`
[6] **for** $q \in \{1, \cdots, \gamma\}$ **do**
[7] $\quad$ **for** $i \in \{1, \cdots, n\}$ **do**
[8] $\quad\quad$ **for** $j \in \{1, \cdots, n\}$ **do**
[9] $\quad\quad\quad$ **if** $C_{=}(x_i, x_j) \in C_{=}$ ***and*** $y_i^{p_q} \neq y_j^{p_q}$ **then**
[10] $\quad\quad\quad\quad w_{[i,j]} \leftarrow w_{[i,j]} + r_{[\gamma-k+1]}$
[11] $\quad\quad\quad$ **end**
[12] $\quad\quad\quad$ **if** $C_{\neq}(x_i, x_j) \in C_{\neq}$ ***and*** $y_i^{p_q} = y_j^{p_q}$ **then**
[13] $\quad\quad\quad\quad w_{[i,j]} \leftarrow w_{[i,j]} + r_k$
[14] $\quad\quad\quad$ **end**
[15] $\quad\quad$ **end**
[16] $\quad$ **end**
[17] **end**
`/* Compute distance matrix` M `by optimizing Equation 6 with the iterated projections method. */`
[18] $M \leftarrow \texttt{WLSI}(X, C_{=}, C_{\neq}, W)$
[19] **return** M

3.2 Similarity Matrix Computation

The second stage of our proposal computes the similarity $S \in \mathbb{R}^{n \times n}$ matrix that will be used later to perform agglomerative clustering. Similarity measures typically used in agglomerative clustering are based on mere pairwise distances. However, this is sensitive to noise and outliers, even if they are able to reasonably capture the local structure of the data [37]. To avoid these drawbacks we follow the work in [28], where a combination of the pairwise distances and the reconstruction coefficient is taken as the optimization problem to solve in order to obtain the similarity matrix S, which is computed following Equation 7.

$$\begin{aligned} &\min_S \frac{\|X-XS\|_F^2}{2} + \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|x_i - x_j\|_M^2 z_{[i,j]} + \frac{\beta}{2}\|S\|_2^2 \\ &\textbf{s.t. } S^T\mathbf{1} = \mathbf{1}, s_{[i,i]} = 0, Z = \frac{|S|+|S^T|}{2} \end{aligned}, \tag{7}$$

where $X = [x_1, \cdots, x_n] \in \mathbb{R}^{u\times n}$ is a matrix containing the instances of the dataset in each column, $\|S\|_2^2$ is a regularization term, λ and β are tradeoff parameters, $\mathbf{1}$ is a column vector of ones with the needed dimension, and the constraint $Z = (|S| + |S^T|)/2$ ensures compatibility between the two similarities combined in the expression. Note that the operator $|\cdot|$ gives the absolute values corresponding to all elements given as arguments and operator $\|\cdot\|_F$ refers to the Frobenius norm. Thanks to the reconstruction coefficient, the above similarity measure is robust to noise and outliers [28]. We can rewrite this problem as in Equation 8 by defining D as a matrix containing pairwise distances within the dataset, computed according to the new distance metric M.

$$\begin{aligned} &\min_S \frac{\|X-XS\|_F^2}{2} + \frac{\lambda}{2}\mathrm{Tr}(ZD^{\circ 2}) + \frac{\beta}{2}\|S\|_2^2 \\ &\textbf{s.t. } S^T\mathbf{1} = \mathbf{1}, s_{[i,i]} = 0 \end{aligned}, \tag{8}$$

where $\mathrm{Tr}(\cdot)$ refers to the trace of the matrix given as argument and $D^{\circ 2}$ is the squared Hadamard exponentiation of D. The mathematical derivation of the optimization scheme for S based on the problem in Equation 7 is analyzed in detail in [28]. It iteratively updates every row s_i of matrix S separately by applying Equation 9.

$$w_{[i,j]} = \mathrm{sign}(v_j)\left(|v_j| - \frac{\lambda d_{[j,i]}}{2}\right)_+, \tag{9}$$

where $s_{[i,j]}$ is the j-th element in row s_i from S, $d_{[j,i]}$ is the j-th of the i-th column from matrix D, and v_j is the j-th element from vector v, which is computed as in Equation 10.

$$v = \frac{X_1^T x}{x^T x + \beta}. \tag{10}$$

In Equation 10 the term X_1 is computed as $X_1 = X - (XZ - x_i s_i)$, with x_i being the i-th column (instance) of X and s_i being the i-th row of matrix S. Note that when $i = j$ then $s_{[i,j]} = 0$.

Once S is obtained, a hard thresholding operator $H_\mu(\cdot)$ is applied to each column. This operator keeps the μ largest entries from the column given as argument and sets the rest to 0. The value of μ influences the resiliency of the overall proposal to noise and outliers, as it sets to 0 the lower similarities in S, effectively removing spurious correlations. Although a high value of μ could also remove valuable information, thus a specific tuning step is required to set its value [28].

Lastly, the resulting similarity measure is obtained by computing $S \leftarrow (|S| + |S^T|)/2$ and applying the unit l_2-norm to each column of S. These steps effectively build a K-NN graph (with μ defining the size of the K-NN neighborhood) as described in Section 2.1, except that in this case the similarity measure is not based only on pairwise distances. Algorithm 2 summarizes the process to obtain the similarity matrix. The loop in line 5 is shown to always converge in [28] and its stopping condition is controlled by the Frobenius norm (noted by $\|\cdot\|_F$) computed over the differences in S between two consecutive iterations.

Algorithm 2: Similarity Matrix Computation (Second Stage)

Input: Dataset X, distance metric matrix M, tradeoff parameter λ and β, thresholding parameter μ, stopping criteria threshold ϵ.

[1] Initialize S with values in $[0, 1]$ such that $S^T\mathbf{1} = \mathbf{1}$ and $s_{[i,i]} = 0$
[2] Initialize $D \in \mathbb{R}^{n\times n}$ as an empty matrix.
`// Compute pairwise distances to obtain` D
[3] **for** $i \in \{i, \cdots, n\}$ **do**
[4] | **for** $j \in \{j, \cdots, n\}$ **do**
[5] | | $d_{[i,j]} \leftarrow \|x_i - x_j\|_M$
[6] | **end**
[7] **end**
[8] $D \leftarrow D^{\circ 2}$
`// Iteratively update` S `until convergence`
[9] **do**
[10] | **for** $i \in \{i, \cdots, n\}$ **do**
[11] | | Obtain row $s_i^{(t+1)}$ following Equation 9 and store it to build $S^{(t+1)}$
[12] | **end**
[13] **while** $\|S - S^{(t+1)}\|_F < \epsilon$;
[14] Apply the hard thresholding operator $H_\mu(\cdot)$ to each column of S
[15] Obtain final similarity matrix by computing $S \leftarrow (|S| + |S^T|)/2$
[16] Normalize each column of S applying the l_2-norm.
[17] **return** S

3.3 Agglomerative Constrained Clustering

AHC methods produce an informative hierarchical structure of clusters. They start with a large (usually n) initial number of clusters and iteratively merge them by computing the affinity $A(\cdot, \cdot)$ between every pair of clusters with respect to some criteria. Formally, for an initial partition $C = \{c_1, \cdots, c_n\}$ an agglomerative clustering method tries to find the two clusters to merge c_i and c_j by applying Equation 1, introduced in Section 2.1.

In CC the constraint sets $C_=$ and $C_\neq$ can be used to bias the affinity towards merges violating the overall smallest possible number of constraints. In order to do so, the infeasibility can be

formally defined as in Equation 11, which can be interpreted as the number of constraints violated by a partition.

$$\text{Infs}(C, C_{=}, C_{\neq}) = \sum_{(x_i,x_j)\in C_{=}} \mathbb{1}[\![y_i^C \neq y_j^C]\!] + \sum_{(x_i,x_j)\in C_{\neq}} \mathbb{1}[\![y_i^C = y_j^C]\!] \tag{11}$$

A reward-style affinity criterion incorporating constraints can be defined as in Equation 12, where S_{c_i,c_j} is a submatrix of S containing row indices from c_i and column indices from c_j, $|c_i|$ is the number of elements in cluster c_i, and α is a scaling parameter for the reward term.

$$\begin{aligned} A(c_i, c_j) = \quad & \tfrac{1}{|c_i|^2}\mathbf{1}_{|c_i|}{}^T S_{c_i,c_j} S_{c_j,c_i} \mathbf{1}_{|c_i|} + \tfrac{1}{|c_j|^2}\mathbf{1}_{|c_j|}{}^T S_{c_j,c_i} S_{c_i,c_j} \mathbf{1}_{|c_j|} + \\ & \underbrace{(|C_{=}| + |C_{\neq}| - \text{Infs}(\{C - \{c_j, c_j\}\} \cup \{c_i \cup c_j\}, C_{=}, C_{\neq})) * \alpha}_{Reward} \end{aligned} . \tag{12}$$

The first two terms from the affinity criterion in Equation 12 can be interpreted as the addition of the input and output degree between the two clusters to merge when S is viewed as a K-NN graph. The rest of the terms define the reward, which is used to incorporate constraints into the clustering process. Parameter α is used to make $A(c_i, c_j)$ a stepped function, so that merges violating a lower number of constraints will always have a higher affinity value. To achieve this, α has to be set to an arbitrarily high value, always higher that the diameter of the dataset. The first argument for the Infs function $\{C - \{c_j, c_j\}\} \cup \{c_i \cup c_j\}$ refers to the partition resulting from the merge of clusters c_i and c_j.

Overall, the affinity criterion in Equation 12 can be interpreted as: among all merges violating the lowest possible number of constraints, choose the one merging the two most similar clusters. Please note that this affinity criterion directly affects the assignment of instances to clusters, while previous stages of 3SHACC aim to capture the overall constraint-based information.

3.4 3SHACC Summary

Algorithm 3 summarizes the overall constrained AHC process. Any kind of reasonable criteria can be used to stop the loop in line 2 from Algorithm 3; although, in cases where k is known, the number of clusters in C can be used to stop the clustering process by testing $|C| > k$. Note that Algorithm 3 returns a partition, although it can be trivially modified to return a dendrogram instead by storing the history of all merges carried out during the clustering process and returning it alongside with the partition C.

For the sake of efficiency, pairwise affinities can be stored in a matrix G so they do not have to be computed in every iteration of the clustering process. Only affinities involving the newly created cluster need to be computed after every merge and stored in G.

Algorithm 3: Constrained Agglomerative Clustering (Third Stage)

Input: Dataset X, constraint sets $C_{=}$ and $C_{\neq}$, similarity matrix S, scaling parameter α.

[1] Assign each instance to a singleton cluster to get initial partition $C = \{c_1, c_2, \cdots, c_n\}$
[2] Initialize affinity matrix G computing pairwise affinities
[3] **while** *stopping criteria not met* **do**
    `/* Select the best two clusters to merge using` A `in Equation 12 */`
[4] $c_a, c_b \leftarrow \arg\max_{i,j} G$
    `/* Merge clusters ↔ Generate new hierarchy level in the dendrogram */`
[5] $C \leftarrow \{C - \{c_a, c_b\}\} \cup \{c_a \cup c_b\}$
    `/* Update affinity matrix` G `computing affinities involving the newly created cluster */`
[6] **for** $i \in \{i, \cdots, |C|\}$ **do**
[7] $G[a, i] \leftarrow A(c_i, \{c_a \cup c_b\})$
[8] **end**
[9] **end**
[10] **return** C

In order for the reader to have a general view of our proposal, Figure 1 shows a pictorial representation of the overall 3SHACC algorithm.

3.5 On the Computational Complexity of 3SHACC

The computational complexity of each 3SHACC stage can be addressed separately. We assume that the constraint sets are given in the form of a matrix containing 1 in positions associated with indices relating instances linked by ML constraints in the datasets, -1 for the case of CL constraints and 0 in case of no constraint. This way the constraint sets can be randomly accessed in constant time. In the following, symbol T is always used to refer to the number of iterations any iterative process is carried out.

The first stage involves running a given algorithm $\mathcal{A}$ over the dataset X a number of times equal to γ, and therefore the overall complexity of this stage depends on $\mathcal{O}(\mathcal{A})$. For the sake of simplicity, we assume that $\mathcal{A}$ is an algorithm with complexity equivalent to the K-means algorithm, whose classic implementation runs in $\mathcal{O}(n \times u \times k \times T)$. As $\mathcal{A}$ has to be run γ times, computing matrix P requires $\mathcal{O}(n \times u \times k \times T \times \gamma)$ operations (assuming convergence on $\mathcal{A}$). The informativity matrix W is also computed in the first stage, which involves iterating over all possible pairs of instances in X to check their labels and contrast them with the constraint sets. This has to be done for every partition in P, so this procedure takes $\mathcal{O}(n^2 \times \gamma)$ operations. In WLSI, target functions and gradient computation require $\mathcal{O}(n^2 \times u^2 \times T)$ operations, and eigenvalue decompositions used to ensure M (computed metric) to be positive semidefinite require $\mathcal{O}(u^3 \times T)$ operations, so WLSI runs in $\mathcal{O}((n^2 \times u^2 + u^3) \times T)$.

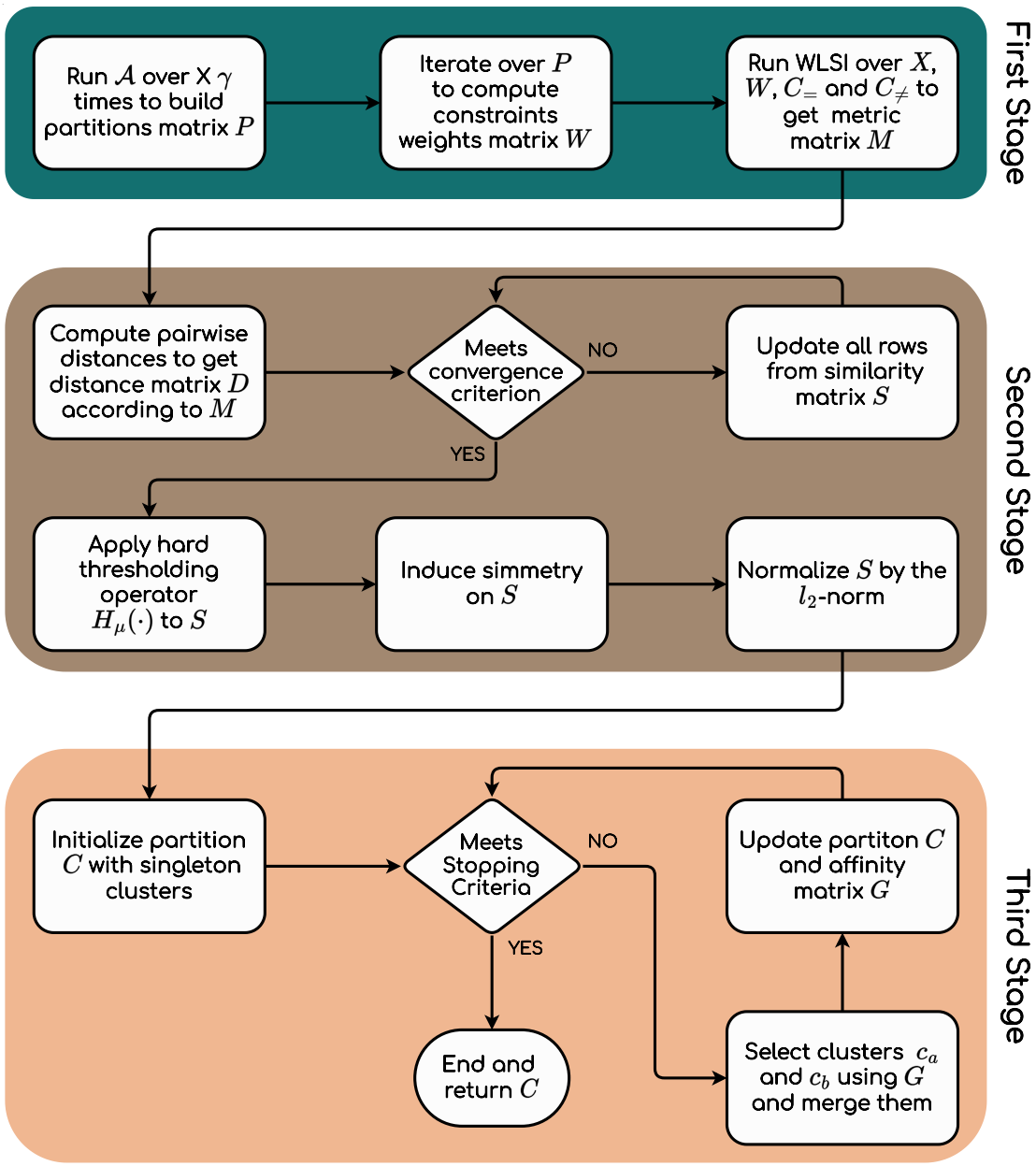


Figure 1: Diagram summarizing the overall 3SHACC clustering process.

Regarding the second stage, it involves computing pairwise distances according to the new metric matrix M, which can be done in $\mathcal{O}(d^2 \times u)$ operations. In [28] an in-depth analysis of the algorithmic complexity for the iterative process used to obtain S can be found, concluding that this procedure can be performed in $\mathcal{O}(d^2 \times u \times T)$ operations.

Finally, computational complexity of the third stage needs to be addressed. For the sake of simplicity we refer to the number of clusters in C as c in this analysis. Computing the affin-

ity between two clusters involves computing the product of two matrices, whose worst case scenario is found when they are square and is known to be of cubic order. This case correspond to partitions where the number of instances belonging to every cluster is as balanced as possible (n/c), so this is the case considered in this analysis. The last term in Equation 12 computes the affinity of a given partition, which can be done in $\mathcal{O}(n)$ operations by using auxiliary matrices. Equation 13 shows puts together all these concepts to present the overall computational complexity of A.

$$\mathcal{O}\left(2\left(\frac{n}{c}\right)^3 + n\right). \tag{13}$$

Firstly, affinity A is first used to initialize G with pairwise distances. At this point the number of clusters is known to be n, as all clusters are singleton ($c = n$). With this, the affinity criterion has to be computed $(n(n-1))/2$ times, as $A(c_i, c_j) = A(c_j, c_i)$. Equation 14 shows the final algorithmic complexity of this procedure, which is $\mathcal{O}(n^3)$.

$$\begin{gathered}\frac{n(n-1)}{2}\left(2\left(\frac{n}{n}\right)^3 + n\right) = \frac{n^2-n}{2}(2+n) = \\ \frac{2n^2+n^3-2n-n^2}{2} \in \mathcal{O}(n^3)\end{gathered} . \tag{14}$$

Secondly, A is used in the clustering iterative process to update matrix G after every merge. This procedure involves carrying out $(c-1)$ affinity computations. In the worst case scenario, a complete dendrogram would be obtained, meaning that G has to be updated for all possible values of c. Equation 15 shows the expression describing the algorithmic complexity of the iterative procedure in charge of obtaining the dendrogram (`while` loop in line 3 from Algorithm 3). Please note that the summation marked with $*$ is bounded by $[0, \pi^2/6)$, which can be taken as constant ($\mathcal{O}(1)$) and therefore does not affect the final complexity.

$$\begin{gathered}\sum_{c=1}^{n}(c-1)\left(\frac{n^3}{c^3}+n\right) = n^3 \overbrace{\sum_{c=1}^{n}\left(\frac{1}{c^2}-\frac{1}{c^3}\right)}^{*} + n\sum_{c=1}^{n}(c-1) = \\ n^3 + n\sum_{c=0}^{n-1} c = n^3 + \frac{n^2(n-1)}{2} = \\ = n^3 + \frac{n^3-n^2}{2} = n^3 + n^3 \in \mathcal{O}(n^3)\end{gathered} . \tag{15}$$

Assuming all iterative procedures involved in 3SHACC converge and assuming that T can always be fixed to a maximum number, then the overall algorithmic complexity of 3SHACC is $\mathcal{O}(n^3)$. Please note that the size of the constraint sets $C_=$ and $C_{\neq}$ is never involved in the complexity analysis, which makes its runtime completely independent of the amount of constraint-based information available.

4 Experimental Setup

A summary of the datasets used in our experiments can be found in Table 2. 25 real-world datasets and 3 constraint sets for each one of them are used to compare 3SHACC to a classic set of state-of-the-art representative methods. These datasets can be found at the Keel-dataset repository[1] [38], and at the `scikit-learn` Python package[2] [39].[3]

Table 2: Summary of datasets used for the experiments.

Name	No. Instances	No. Classes	No. Features
Appendicitis	106	2	7
Balance	625	3	4
Breast Cancer	569	2	30
Bupa	345	16	5
Ecoli	336	8	7
Glass	214	6	9
Haberman	306	2	3
Hayes Roth	160	3	4
Heart	270	2	13
Ionosphere	351	2	33
Iris	150	3	4
Monk2	432	2	6
Movement Libras	360	15	90
Newthyroid	215	3	5
Pima	768	2	8
Saheart	462	2	9
Segment*	693	7	19
Sonar	208	2	60
Soybean	47	4	35
Spectfheart	267	2	44
Tae	151	3	5
Vehicle	846	4	18
Wdbc	569	2	30
Wine	178	3	13
Zoo	101	7	16

Constraint sets can be easily built by having access to the labels of a given dataset. Ground-truth labels are used as an oracle which is queried with two instances. An ML or CL constraint is set between the two instances; depending on whether or not they belong to the

[1] https://sci2s.ugr.es/keel/category.php?cat=clas
[2] https://scikit-learn.org/stable/datasets/index.html
[3] Dataset *Segment*, marked with $*$, has been undersampled to 30%

same class (see Section 4.1). Another reason why classification datasets are commonly used in the CC literature is that the true labels can be used to assess the quality of the results (see Section 4.2) and therefore an objective ground-truth based comparison can be carried out.

The ARI index is widely used in CC-related literature as the main external validation index and quality indicator to evaluate the performance of new CC proposals. Please note that the original ground truth labels always reflect a valid partitioning of the data, as constraints are randomly allocated but not randomly generated. This means that the pairs of instances on which constraints are imposed are randomly selected, but the constraint itself is generated by querying an oracle (the true labels) and placing an ML constraint if the labels of the instances are equal, or a CL constraint if they are different.

4.1 Constraint Generation

Following the method proposed in [7], three constraint sets are generated for every dataset, with incremental level of constraint-based information. This method randomly selects two instances from the dataset to set an ML or CL constraint between the instances attending to their class membership, in this way the constraint set is built. Table 2 summarizes the constraint sets generated for every dataset.

The three constraint sets CS_{10}, CS_{15} and CS_{20} are associated with a small percentage of the size of the dataset: 10%, 15% and 20%, respectively. The number of constraints generated for every constraint set associated with every dataset is obtained by computing the number of edges in a complete graph with n_f vertices with the formula $(n_f(n_f-1))/2$.

The bias introduced by using subsets of labeled data can be avoided by randomly allocating the constraints as described above, as the risk of biasing the constraint set towards classes with poor representation is lower. Table 3 displays the number of constraints of each type obtained for every dataset and constraint set. All these constraint sets can be found here [4].

4.2 Evaluation Method

The Adjusted Rand Index (ARI) can be used to evaluate the quality of the partition obtained by any given CC method [40]. By using the ARI measure, the degree of similarity of two partitions for the same dataset can be computed; therefore, if one of these partitions corresponds to the true labels—which we have available—, the degree of similarity with the ground truth can be obtained.

ARI is the corrected-for-chance version of the basic Rand Index, which computes the degree of agreement between two partitions C_1 and C_2 of a given dataset X. C_1 and C_2 can be viewed as collections of $n(n-1)/2$ pairwise decisions [41]. For each x_i and x_j in X, they are assigned to the same cluster or to different clusters by a partition. The number of pairings where x_i is in the same cluster as x_j in both C_1 and C_2 is taken as a; conversely, b represents the number of pairings where x_i and x_j are in different clusters. The degree of similarity between C_1 and C_2 is computed as in Equation 16.

[4] https://drive.google.com/drive/u/1/folders/1sjnPYitey8q9zrPKa_YpFS7iNKTT15QH

Table 3: Number of constraints used in experiments.

Dataset	CS_{10}		CS_{15}		CS_{20}	
	ML	CL	ML	CL	ML	CL
Appendicitis	37	18	76	44	154	77
Balance	841	1112	1846	2525	3324	4426
Breast Cancer	846	750	1954	1701	3292	3149
Bupa	91	504	217	1109	374	1972
Ecoli	147	414	352	923	644	1634
Glass	58	173	138	390	233	670
Haberman	273	192	631	404	1173	718
Hayes Roth	47	73	86	190	177	319
Heart	173	178	436	384	747	684
Ionosphere	338	292	738	640	1357	1128
Iris	28	77	92	161	132	303
Monk2	484	462	1064	1016	1835	1906
Movement Libras	46	584	83	1348	139	2417
Newthyroid	125	106	273	255	488	415
Pima	1572	1354	3601	3069	6443	5338
Saheart	613	468	1281	1134	2368	1910
Segment*	377	2038	738	4618	1353	8238
Sonar	106	104	241	255	416	445
Soybean	1	9	11	17	5	40
Spectfheart	230	121	563	257	952	479
Tae	31	89	88	165	164	301
Vehicle	916	2654	1993	6008	3576	10789
Wdbc	864	732	1975	1680	3448	2993
Wine	57	96	105	246	210	420
Zoo	13	42	27	93	53	157

$$\text{Rand}(C_1, C_2) = \frac{a+b}{n(n-1)/2} \tag{16}$$

Equation 16 can be corrected for chance by taking into account the expected similarity of all comparisons between partitions specified by the random model to establish a baseline. Equation (17) shows the expression used to compute ARI.

$$\text{ARI}(C_1, C_2) = \frac{\text{Rand}(C_1, C_2) - \text{Expected Index}}{\text{Maximum Index} - \text{Expected Index}}, \tag{17}$$

where Expected Index is the degree of similarity with a random model and Maximum Index is assumed to be 1. With this, it is clear that $\text{ARI}(C_1, C_2) \in [-1, 1]$, with a value close to 1 meaning a high degree of agreement between C_1 and C_2, positive values close to 0 meaning no agreement, and a value below 0 meaning that the $\text{Rand}(C_1, C_2)$ is less than expected when compared with the results of a random model. To summarize: the higher the ARI, the greater the degree of similarity between C_1 and C_2. For more details on ARI, see [40].

The ability of each method to integrate constraints into the clustering process also needs to be evaluated. To this end, we define the Unsat measure, which, given a partition C and the constraint sets $C_=$ and $C_\neq$, computes the percentage of unsatisfied constraints as in Equation 18.

$$\text{Unsat}(C, C_=, C_\neq) = \frac{\text{Infs}(C, C_=, C_\neq)}{|C_=| + |C_\neq|}, \tag{18}$$

4.3 Validation of results

An in-depth analysis of the disadvantages of the classic Null Hypothesis Statistical Tests (NHST) can be found in [42], where a new model is proposed to carry out statistical comparisons between two sets of results. In [42], the authors summarize the drawbacks of NHST as *"In a nutshell: NHST do not answer the question we ask"*. These drawbacks stem from the trap of black-and-white thinking, that is: to reject, or not to reject?

Bayesian tests can be used to overcome the drawbacks found in NHST. The Bayesian sign test is the Bayesian version of the NHST non-parametric sign test. It can be easily used to compare two sets of results by employing the `rNPBST` R package in [43]. A general use guide to Bayesian statistical testing can be found in [44].

The Bayesian sign test obtains the statistical Dirichlet distribution of a parameter ρ according to the differences between two sets of results. The number of cases where $A - B < 0$, the number of cases where no significant differences are found, and the number of cases where $A - B > 0$ are counted to get the distribution of ρ. The region of practical equivalence (rope) $[r_{\min}, r_{\max}]$ needs to be defined in order to identify cases where there are no significant differences, so that $P(A \approx B) = P(\rho \in \text{rope})$. Finally, the weights of the Dirichlet distribution are computed so it can be sampled to get triplets with the following form:

$$[P(\rho < r_{\min}) = P(A - B < 0), \;\; P(\rho \in \text{rope}), \;\; P(\rho > r_{\max}) = P(A - B > 0)]$$

4.4 Competing Methods

Our proposal, 3SHACC, is compared with other six previous CC algorithms, including 2SHACC. These methods can be briefly summarized as follows:

- **2SHACC**: Two Stages Hybrid Agglomerative Constrained Clustering is presented in [29]. It uses the Floyd-Warshall algorithm to propagate the information contained in the constraints set over the distance matrix. The updated distance matrix is used later to determine similarities between clusters. These clusters are generated through an agglomerative clustering process, in which an alternative constraint-influenced criterion is used in cases where the information contained in the similarity matrix is not enough to select two clusters to merge.

- **COPKM**: COnstrained Partitional K-means modifies the assignment rule of instances to clusters of the classic K-means algorithm in such a way that an instance can be assigned to a cluster only if no constraints are violated by the assignment [7].

- **LCVQE**: The Linear Constrained Vector Quantization Error algorithm introduces a modification of the cost function of CVQE to make it less computationally complex [45].

- **TVClust**: Two Views Clustering applies clustering methods on the dataset and the constraint set separately, and tries to find a consensus between the results [46]. It makes a soft interpretation of the constraints.

- **RDPM**: Relation Dirichlet Process - Means can be viewed as an extension of K-means that includes side information (constraints). It is a deterministic derivation of the TVClust model. The number of clusters (k) does not need to be specified [46].

  PCSKM: The Pairwise Constrained Sparse K-Means algorithm is an extension of the classic Sparse K-Means algorithm that integrates constraints by means of a weighted penalty term [47].

4.5 Calibration

This section discusses 3SHACC parameters, analyzing their impact on the quality of the results. Only three of the parameters of 3SHACC have an impact on the results in a way that is not easy to predict: λ and β (which are the tradeoff parameters for the pairwise distances and the reconstruction coefficients in the computation of similarity matrix and μ (which is the parameter for the hard thresholding operator). The rest of the parameters have very little impact on the results or are not suitable for optimization. A basic intuition on the influence of these parameter is given as follows:

- Parameter ε represents the minimum weight a constraint can have when computing the metric matrix M with the WLSI procedure. Its value has to be grater than 0 to prevent instances from collapsing in a single point. A high value for ε would mask constraints considered to be truly relevant by WLSI.

- Parameter ϵ gives the threshold for the iterative procedure computing S to stop. It is clear that the lower the value, the more stable the values of S would be and the more time-consuming the iterative process would be. The rule to set this parameter is the closer to 0, the better, taking computational time into account.

- Parameter α is used as the scaling parameter for the reward term in the affinity computation. It is left as a parameter so the user can set it to 0 in cases where constraints must not be taking into account; however, any value grater than 1 would provide the same results when working with normalized datasets. Datasets must always be normalized when working with the Euclidean distance. Values for α between 0 and 1 are nor recommended, as they would interfere with the balance between the reward term and the affinity term in Equation 12, making it a non-stepped function.

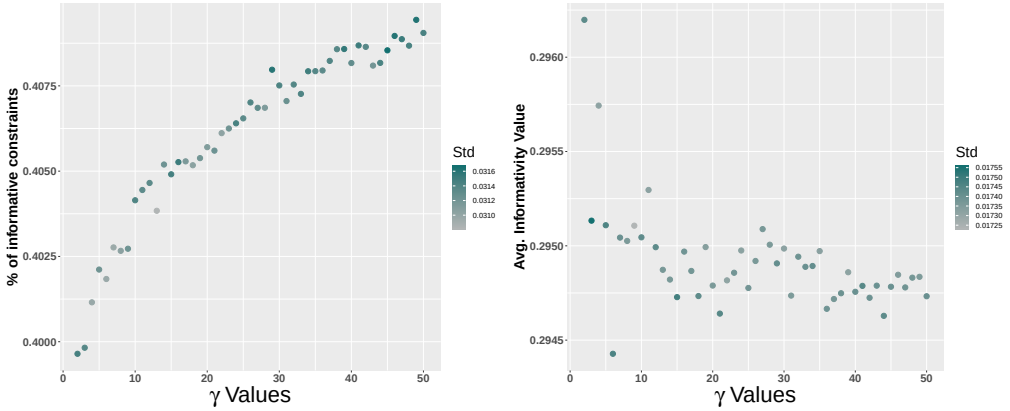
- Parameter $\mathcal{A}$ refers to the algorithm to be used to compute the informativity matrix W. It can be any classic clustering algorithm and it has to be selected specifically for every problem to be solved. We used the K-means algorithm because of the familiarity with it, its efficiency and its simplicity. It is clear that this parameter can not be optimized via classic optimization methods.

- Parameter r gives the array used to compute the informativity matrix. It can be set on the basis of the confidence in the constraint set. The lower part of the range should have a value close to 0 when the constraint set is expected to present noise or inconsistencies.

The influence of γ in the computation of the informativity matrix W can be studied separately from the second and the third stage of the 3SHACC method (see Figure 1). It can be observed in Figures 2a and 2b, which show average results over the 25 datasets used in our experiments. The highest number of constraints (found in CS_{20}) is used, so that the effects of γ are more notorious. Figure 2a depicts the percentage of constraints that are considered to be informative as a function of the number of times algorithm $\mathcal{A}$ is run over the dataset, given by the value of parameter γ in the range $[1, 50]$. The rest of the parameters are set to fixed values (see Table 4). It can be observed how the percentage of informative constraints grows as the value of γ increases, with a tendency towards stabilization (as there is a limited number of constraints).

The opposite tendency can be observed in Figure 2b, which represents the average informativity value as function of γ. Contrary to what can be observed in Figure 2a, the average value of the informativity matrix decreases and stabilizes as the value of γ increases. This happens because constraints found to be informative for a given value of k may not be so for different values of k, which is not fixed by the user but takes values from the range $[\lceil n/3 \rceil, \lceil 2n/3 \rceil]$. Let

us remember that the higher the number of cluster available, the easier it is to satisfy CL constraints; whereas the lower it is, the easier to satisfy ML constraints. Parameter γ has te be fine-tuned for every particular case in out-of-lab applications, therefore, a balance between the percentage of informative constraints and the average informativity value is found.



(a) (b)

Figure 2: Influence of parameter γ over the informativity matrix.

Given the strong interdependence between the λ and β tradeoff parameters, an independent analysis is not suitable for studying their influence over 3SHACC results. A classic optimization grid can be used to address this task. The aim of this study is to get the general intuition on the influence of λ and β, rather than carrying out a particularized and in-depth analysis of these parameters. The 10 smaller datasets (in terms of number of instances) presented in Table 2 have been selected in order to average results: iris, appendicitis, glass, hayes roth, heart, newthyroid, sonar, tae, wine and zoo. Figure 3 shows the optimization grid featuring averaged ARI results over the selected datasets for values of λ and β ranging from 0 to 11. The rest of the parameters are assigned fixed values (see Table 4). A clear preference towards low λ values and high β values can be observed. Let us remember that λ determines the weight for the pairwise distances and β does so for the similarity matrix S. These results can be interpreted as a preference towards high constraint-influenced results, as S is computed on the basis of the metric computed with WLSI, which uses constraints to find this metric. Please note that, in cases where constraints are not generated from the ground truth, lower values of β would be preferred. Such scenario is not presented in this study, since there is no noise in the constraints and they are generated from the true labels.

The μ parameter determines the number of instances to be removed from every column of the similarity matrix in the final stage of its computation. It is used to eliminate spurious correlations between pairs of instances, which allows the control over the resiliency of 3SHACC to noise and outliers. As a summary of ideas: for every instance, the μ less similar instances are considered not to be similar. To examine the influence of this parameter over 3SHACC results, average ARI results have been obtained for the same 10 datasets used in the study of λ and β. Values for μ range from $n/2$ to $n/50$. Figure 4 shows average ARI values obtained

Figure 3: ARI results for the λ and β optimization grid.

for the 10 datasets and the values for μ. A tendency towards stabilization around $\mu = n/10$ can be observed. The datasets used in our experiments are free of noise and outliers, hence low values of μ are expected to be preferred, since there is no spurious correlations in our data. Such case may not be shown in out-of-lab applications.

These results will be taken into account to fix the parameter of 3SHACC in the experimental section. Although an in-depth analysis of these parameters have not been performed yet, these results can be used to guide our intuition to set their value. Please note that the parameter settings highly depend on the datasets and the problem to be solved, so that a parameter tuning step needs to be carried out for every out-of-lab specific application.

4.6 Parameters setting

In order for our experiments to be accurately reproducible, the parameter setups used for every competing method are described in this section. Table 4 shows a summary of every parameter needed for 3SHACC, and shows the value used in our experiments for all of them.

A stopping criterion needs to be specified in order for the loop in line 2 from Algorithm 3 to reach an end. For our experiments, the criterion would be the number of different clusters remaining in partition C. This number decreases in each iteration, as two clusters are always merged into one cluster in every iteration, eventually reaching a single cluster that encompasses all instances in dataset X. This ensures convergence and allows us to generate partitions with the optimal number of clusters for comparison purposes, as k is known for all classification datasets used in our experiments.

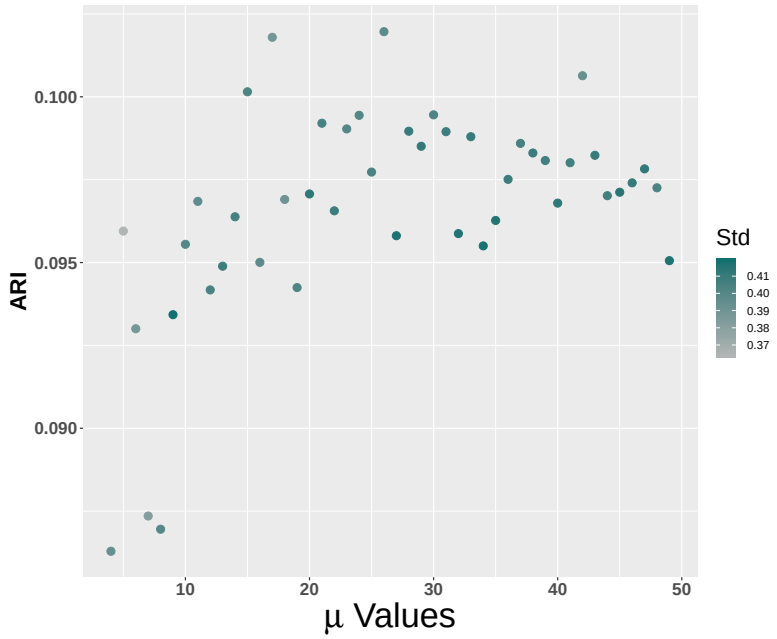


Figure 4: Influence of μ over average ARI results.

3SHACC makes use of auxiliary procedures such as the iterated projections methods (mentioned in Section 3) and some kind of classic clustering method to be used as algorithm $\mathcal{A}$, which in this case is the K-means algorithm. Parameters for these procedures are described in Table 5. The scikit-learn [39] K-means implementation has been used, whose documentation can be found here[5]. The Weighted LSI method has been included in the `pyDML`[6] package, taking the implementation of classic LSI as basis. A tutorial on the use of the `pyDML` package can be found in [48].

Note that parameter r from Table 4 and `n_clusters` from Table 5 are given in the form of intervals. This is because γ values need to be specified for these parameters, as algorithm $\mathcal{A}$ is run γ times over dataset X. For our experiments, values in array r are set by dividing the interval $[0.5, 1]$ into γ regular subintervals. As k is not known by the 3SHACC method, the number of clusters given to K-means to perform clustering is not fixed, again, these values correspond to the interval $[\lceil n/3 \rceil, \lceil 2n/3 \rceil]$ divided into γ regular subintervals, rounding up in order for this parameter to be an integer number. This may lead to repeated values, which does not pose any problem whatsoever. In order to induce variability in the results of the K-means algorithm, centroids are randomly initialized, and the maximum number of iterations is set to a random value from the interval $[10, 50]$.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
[6] https://pydml.readthedocs.io/en/latest/

Parameter	Meaning	Value
γ	Number of times algorithm $\mathcal{A}$ is run to compute informativity matrix W.	20
ε	Minimum weight a constraint can have to compute metric matrix M.	0.1
λ	Tradeoff parameter on the influence of pairwise distances in the computation of similarity matrix S.	8.5
β	Tradeoff parameter on the influence of reconstruction coefficients in the computation of similarity matrix S.	2.5
μ	Parameter for the hard thresholding operator $H_\mu(\cdot)$.	$n/10$
ϵ	Stopping threshold for the iterative procedure computing similarity matrix S.	1×10^{-3}
α	Scaling parameter for the reward term in pairwise clusters affinity computation function A.	10000
r	Weighting array used to compute informativity matrix W.	$[0.5, 1]$
$\mathcal{A}$	Algorithm to be used as reference to compute informativity matrix W.	K-means

Table 4: 3SHACC parameters setup.

Method name	Parameters name and values
K-means	`n_clusters` $= [\lceil n/3 \rceil, \lceil 2n/3 \rceil]$; `max_iter` $=$ rand($[10, 50]$); `init` $=$ ``random''
Weighted LSI	`eta0` $= 0.1$; `max_iter` $= 1000$; `max_proj_iter` $= 5000$; `itproj_err` $= 1 \times 10^{-5}$; `err` $= 1 \times 10^{-5}$; `weights_thresh` $= 0.1$;

Table 5: Auxiliary procedures parameters setup.

Finally, Table 6 describes parameter setup for the previous approaches mentioned in Section 4.4. A Python implementation for these methods can be found at GitHub[7]. In all cases the value for k is equal to the number of classes displayed in Table 2.

[7]https://github.com/GermangUgr/TFG/tree/master/Software

Method name	Parameters name and values
2SHACC	$\lambda = 2$; $\beta = 0.5$; $\epsilon = 10^{-3}$; `max_it` $= 200$
COPKM	`max_iter` $= 300$; `tolerance` $= 1 * 10^{-4}$; `init_mode` $=$ ``rand''
LCVQE	`max_iter` $= 300$; `initial_centroids` $= \emptyset$
RDPM	`max_iter` $= 300$; $\xi_0 = 0.1$; $\xi_{\text{rate}} = 1$; λ is calculated on the basis of the mean distances in the dataset.
TVClust	`max_iter` $= 300$; $\alpha_0 = 1.2$ `stop_threshold` $= 5 * 10^{-4}$
PCSKM	`max_iter` $= 300$; `tol` $= 1 * 10^{-4}$; `sparsity` $= 1.1$; `init_centroids` $=$ []

Table 6: Classic methods parameters setup.

Since our goal is not to optimize parameters in a case-by-case basis, but to get the fairest and most general comparison possible, we have not included a parameter fine-tuning step for any competing method, including our proposal 3SHACC. Also, the high number of datasets used in our experiments makes tuning each parameter specifically for each dataset unfeasible in a reasonable time. Parameters for the state-of-the-art methods have been set following the guidelines of the authors, and 3SHACC parameters are set by taking into account the preliminary experimentation carried out in Section 4.5. The final purpose of this work is to provide a fair comparison between algorithms, assessing their robustness in a common environment with multiple datasets.

All our experiments have been carried out in the Hercules computing server of the University of Granada, which features 19 computing nodes with two 2.2 GHz Intel®Xeon Silver 4214 processor, 256 GB of RAM, a 6 TB SATA2 HDD and a 1 TB SSD in each node. Two Gigabit Ethernet internal nets are used to interconnect nodes. Ubuntu 20 LTS is installed in each node.

5 Experimental Results

In this section experimental results are reported. For the sake of readability, all the tables have been moved to Appendix A. Tables 7 to 12 display results obtained by our proposal 3SHACC and the six previous approaches for all 25 datasets and the three constraint sets CS_{10}, CS_{15}, CS_{20} generated for every one of them. Given that all methods compared in this study rely on non-deterministic procedures to different extents, variations in the results can be found between different runs. Every method is run 30 times for every dataset and

constraint set to mitigate the effects these non-deterministic procedures could have on the results. This makes a total of 15750 experiments. In all tables, the "Avg." column shows average results for every compared method in every dataset, while the "SD" column shows the Standard Deviation. Please note that the only non-deterministic procedure involved in our proposal 3SHACC is the K-means algorithm used for the computation of the informativity matrix W, which affects final partition results indirectly, so it is not surprising that standard deviations for our proposal 3SHACC are found to be consistently 0 for all our experiments.

Please note that some of the COPKM results are missing. This is due to the inconsistency of this method to produce results, as it is highly dependent on the order in which constraints and instances are explored and may reach dead ends. COPKM may not be able to find a solution for a given constraint set, even if this solution always exists, as constraints are generated on the basis of the true labels and no noise is introduced in the constraint set. This is a major drawback of the COPKM method, and cases where COPKM does not output a partition are considered to produce the worst benchmark values for plotting and averaging purposes.

Figures 5, 6 and 7 are used to compare average results for all methods, and we refer to them as average plots. They allow for a quick view of the distribution of ARI results achieved by each method within the ARI output range $[-1, 1]$, with black marks representing average results for each method.

The results obtained by every method for the CS_{10} constraint set generated for every dataset are presented in Tables 7 and 8 (in Appendix A). These results display strong evidence of 3SHACC representing a consistent improvement in average results over all other methods compared, as it is able to achieve optimal results in up to four cases. Let us remember that an ARI value of 1 indicates a perfect match with the true labels of a given dataset. It can also be observed how our proposal presents one of the lower standard deviation (computed over average results) among all methods compared in this table, suggesting evidence of improvements on reliability over the six classic approaches to CC. Figure 5 offers a very clear visual representation of the above: the black bar, which represents average results, is significantly higher for our proposal.

The results obtained for the CS_{15} constraint set are presented in Tables 9 and 10 (in Appendix A). 3SHACC continues to outperform all other methods, even increasing the advantage over them and achieving optimal results in 13 datasets. This is indicative of a proper constraint integration scheme for the base agglomerative clustering process. It is also worth noting how all methods are able to scale the quality of the results with the amount of constraint-based information available, with 3SHACC being the one capable of leveraging this information the best, as it scales the quality of the results at a higher rate than its six competitors, followed by the most recent of them —PCSKM. This can be observed in Figure 6, where the upward movement of all black bars is quite noticeable. Regarding the standard deviation, the conclusions obtained from Tables 7 and 8 are reinforced by results displayed in Tables 9 and 10.

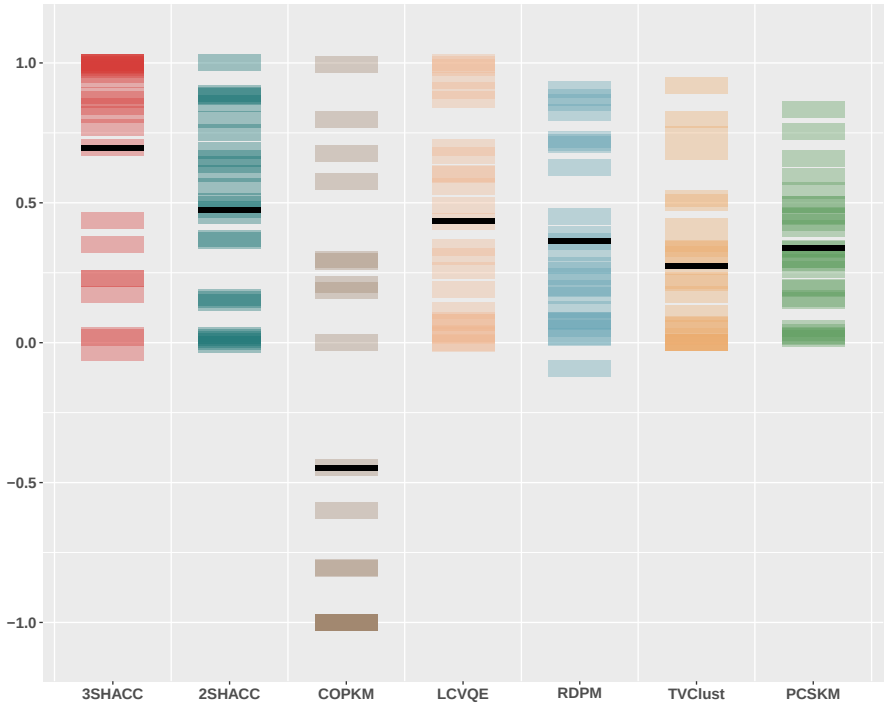


Figure 5: CS_{10} comparative average plot.

In Tables 11 and 12 (in Appendix A), featuring results obtained for the CS_{20} constraint set, the scaling tendency observed in previous tables remains. Once again, our proposal outperforms classic methods, achieving near-optimal average results, as it can be observed in Figure 7, where all red bars congregate near the top (optimal ARI value). In general, tendencies observed from Tables 7 and 8 to Table 9 and 10 are also validated by Tables 11 and 12, including the scaling capabilities of PCSKM.

Figure 8a visually summarizes overall results obtained by the seven methods compared. We can observe how 3SHACC consistently achieves the best average results, representing a reliable improvement over all other methods compared. 3SHACC is capable of scaling the quality of the results with the amount of constraint-based information available, as well as keeping standard deviation in its results under reasonable values. This is strong evidence of 3SHACC being able to incorporate constraints into the clustering process regularly and with remarkable effectiveness, as the quality of the results obtained for individual datasets never decreases when the size of the constraint sets increases.

Figure 8b shows a clear comparison for the overall number of optimal results obtained by each method compared. Again, it is 3SHACC the one obtaining the best results in terms of number of optimal solutions. Please note how the number of optimal results obtained for the Unsat measure is consistently higher than the one obtained for the ARI measure for every method, meaning that satisfying all constraints does not always lead to the optimal solution.

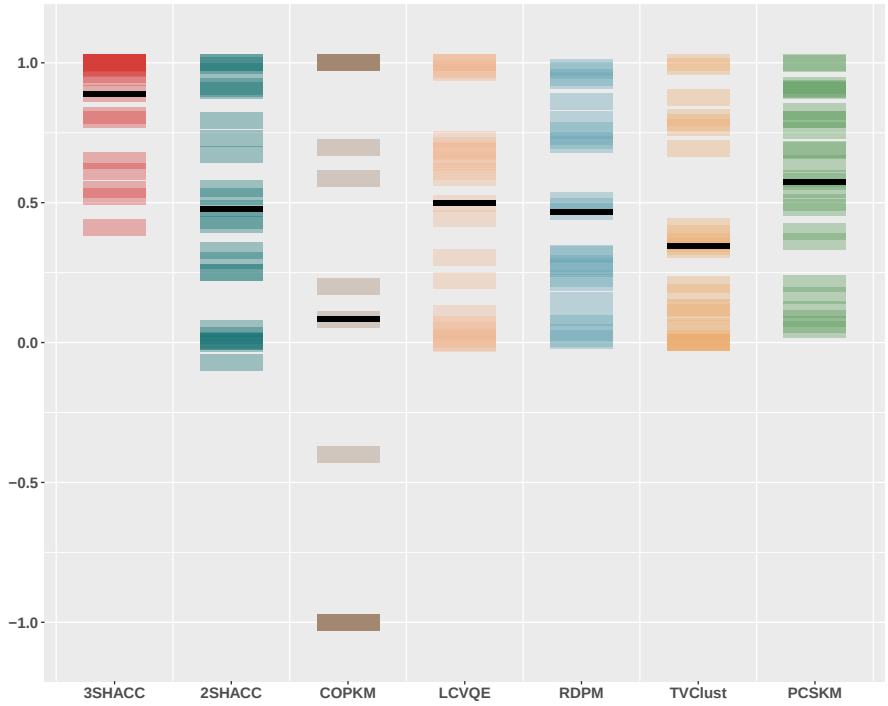


Figure 6: CS_{15} comparative average plot.

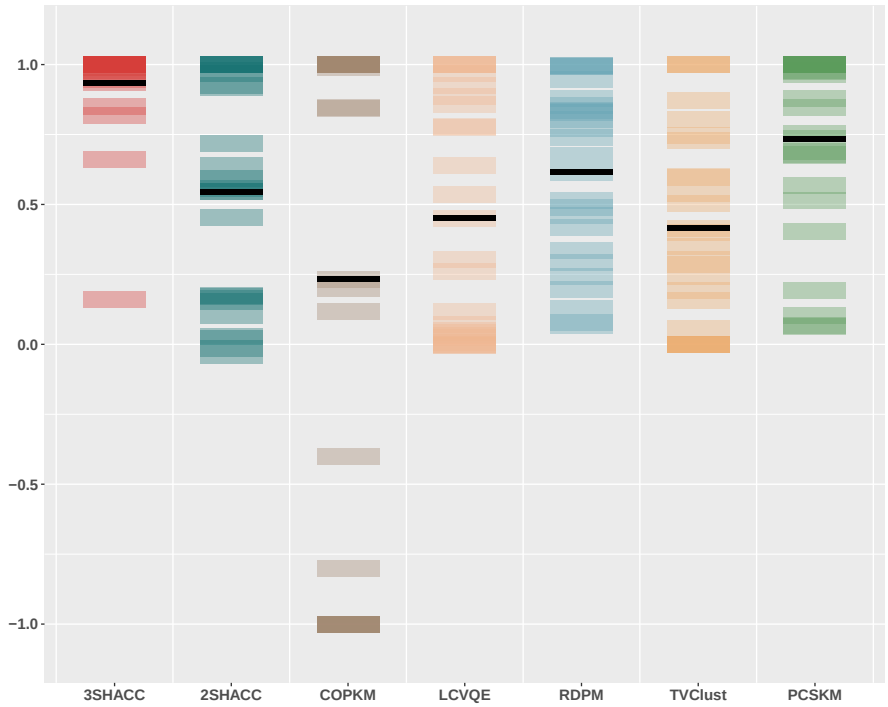
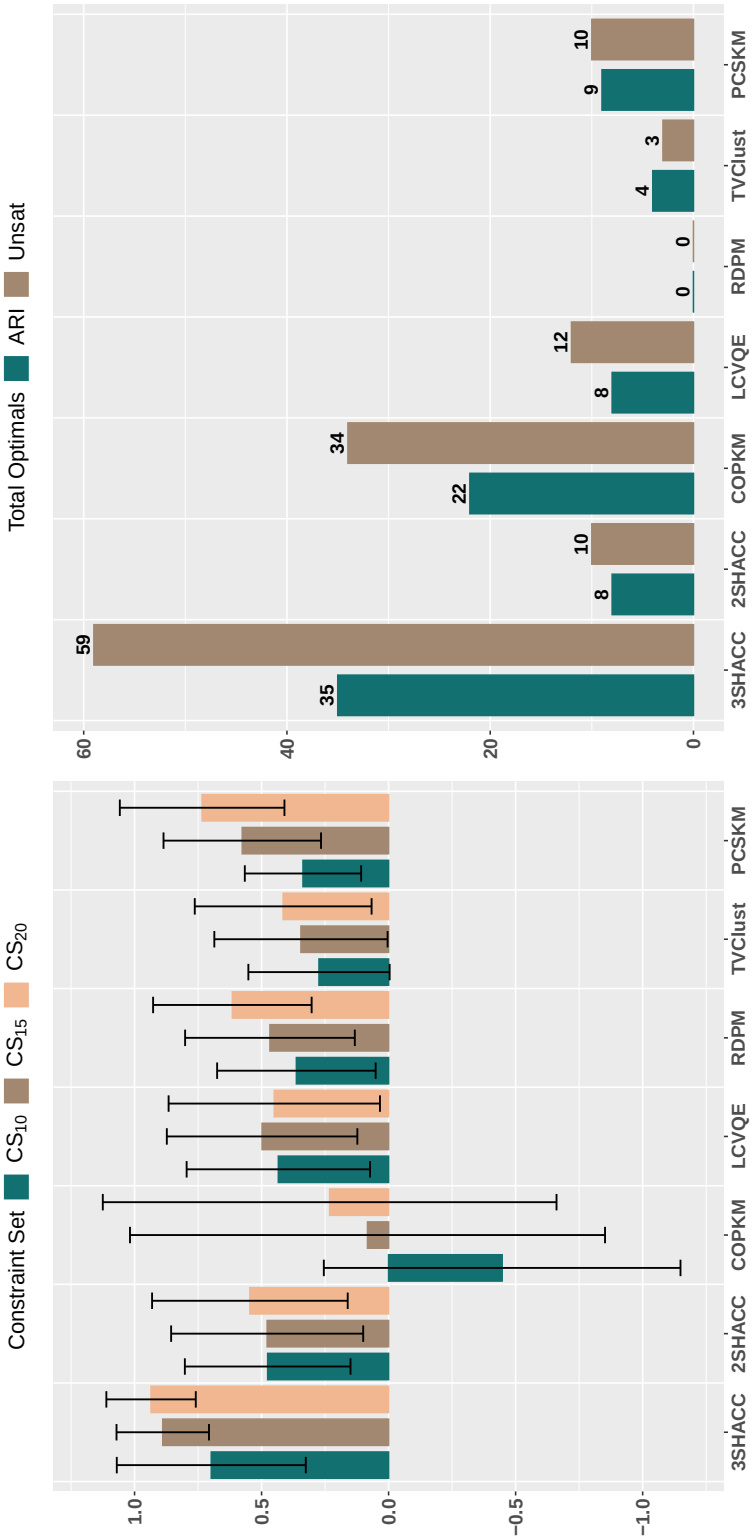


Figure 7: CS_{20} comparative average plot.



(a) Diagram summarizing average ARI results.

(b) Optimal results obtained by every method.

Figure 8: General statistics about the overall results

6 Statistical Analysis of Results

An empirical analysis comparing all ARI results obtained by every method—25 datasets in combination with 3 constraint sets for a total of 75 results—can be performed by using the Bayesian signed-rank test. This test is similar to the Bayesian sign test described in Section 4.3, with the only difference being that it sorts (ranks) differences by absolute values when obtaining the Dirichlet distribution of the parameter ρ. Keeping in mind the notation introduced in Section 4.3, we take the results obtained by a given classic method as sample A, and the results obtained by 3SHACC as sample B. Since ARI is a measure to maximize, a higher value for $P(\rho < r_{\text{min}}) = P(A - B < 0)$ would give the advantage to 3SHACC, whereas a high value for $P(\rho > r_{\text{max}}) = P(A - B > 0)$ would mean the opposite. The rope area has been set to rope $= [-0.02, 0.02]$, following the guidelines in [44].

A very illustrative visual representation of the results obtained by the Bayesian signed-rank test can be created. When the result distribution has been sampled, it can be represented in the form of a heatmap by plotting every triplet in it as a point in barycentric coordinates in an equilateral triangle. To do so, each triplet value is associated with each of the three vertices of the triangle; the higher the value, the closer it is to its associated vertex. Values of every triplet must add up to one, as they describe a probability distribution, so all triplets lie within the triangle.

Figure 9 shows heatmaps comparing our proposal with all other six competing methods in a 1vs1 style, taking into account all 75 results obtained for each comparison and with color indicating the density of points in a given region: yellow represents high density and red is used for low density. We can observe how there is not a single triplet represented in the top region of any heatmap. This means that the Bayesian signed-rank test assigns a very low probability to our proposal being equivalent to any other method. In general terms, all heatmaps point to the same conclusion: there is statistical evidence of our proposal 3SHACC being different to all other methods compared, with these difference being in favor of 3SHACC.

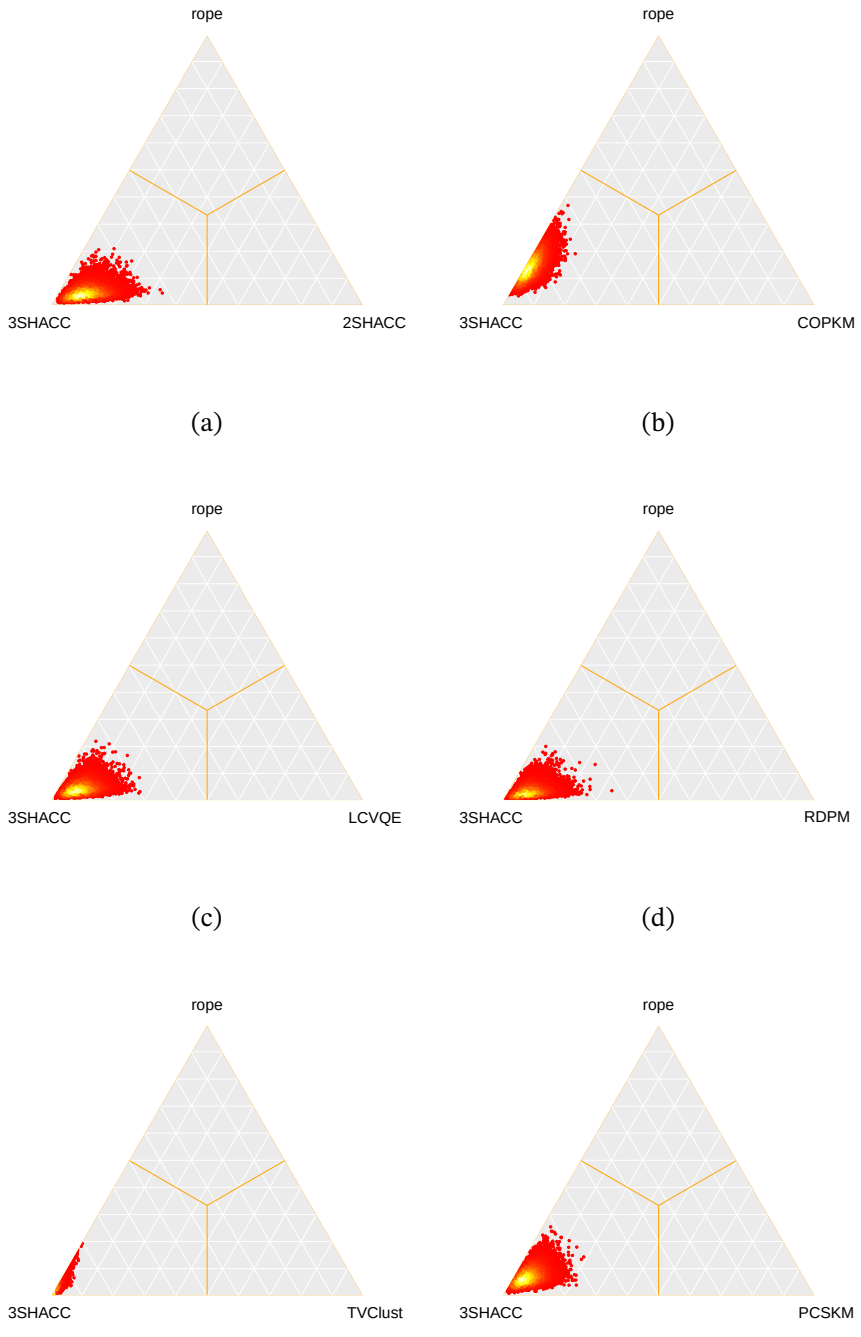


Figure 9: Heatmaps comparing 3SHACC with all other compared methods.

7 Conclusions

In this paper, the Three Stages Hierarchical Agglomerative Constrained Clustering (3SHACC) method is proposed. It approaches the CC problem by combining distance-based methods and clustering engine-adapting methods. 3SHACC implements procedures to determine the relevance of individual constraints in a given set of constraints. This relevance is quantified and later used in the distance-based phase by the newly proposed distance metric learning method WLSI (Weighted Learning from Side Information), which is also proposed in this paper. Finally, 3SHACC performs agglomerative clustering by taking into account both similarities on a pairwise basis and the overall number of violated constraints.

3SHACC is compared with a total of six previous approaches to the CC problem over a total of 25 classification datasets and 3 constraint sets generated for each one of them, including 2SHACC, which is the basis of 3SHACC. These constraint sets are generated with incremental levels of constraint-based information, which allows for a study on the scalability of solution quality with respect to this parameter. The experimental results show 3SHACC represents a consistent improvement in average performance over classic approaches to CC and over its predecessor 2SHACC in both measures analyzed—ARI and Unsat. They also constitute strong evidence of 3SHACC being able to scale the quality of the results with the amount of constraint-based information available to a higher extent than classic approaches. These conclusions are supported by the Bayesian signed-rank test, which assigns a significantly higher probability to our proposal being better on average than any other method presented in this study.

Acknowledgements

This study is has been funded by the research projects `TIN2017-89517-P`, `PID2020-119478GB-I00` and `A-TIC-434-UGR20`.

A Experimental Results - Tables

This appendix presents result tables for the clustering quality comparison. Tables 7, 9 and 11 show results for ARI measure. Tables 8, 10 and 12 present results for Unsat measure. All these tables are referenced and analyzed in Section 5.

ARI Results for CS_{10}

Dataset	3SHACC		2SHACC		COPKM		LCVQE		RDPM		TVClust		PCSKM	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	.018	.000	**.636**	**.000**	-	-	.257	.000	.322	.087	.106	.227	.152	.181
Balance	**1.00**	**.000**	.004	.000	-	-	.606	.036	.235	.053	.921	.080	.544	.126
Breast Cancer	**1.00**	**.000**	.657	.000	.000	1.00	.903	.000	.859	.048	.023	.049	.594	.308
Bupa	**.228**	**.000**	.020	.000	.188	.018	.031	.005	.074	.039	.065	.024	.049	.016
Ecoli	**.768**	**.000**	.495	.000	.577	.043	.492	.082	.451	.145	.502	.105	.494	.113
Glass	.228	.000	.160	.000	.209	.028	.191	.010	.257	.021	.221	.070	**.262**	**.085**
Haberman	**.959**	**.000**	.143	.000	-	-	-.002	.000	.169	.223	.416	.031	.298	.211
Hayes Roth	.024	.000	-.007	.000	-	-	.073	.044	**.081**	**.037**	.063	.028	.014	.017
Heart	**.843**	**.000**	.750	.000	-	-	.603	.000	.197	.061	.216	.254	.192	.151
Ionosphere	**.977**	**.000**	.856	.000	-.807	.580	.342	.000	.195	.061	.000	.000	.157	.149
Iris	**.941**	**.000**	.558	.000	.297	.849	.868	.000	.626	.088	.516	.051	.756	.147
Monk2	**.991**	**.000**	.686	.000	-	-	.699	.004	.023	.007	.336	.439	.409	.278
Mov. Libras	.352	.000	**.374**	**.000**	.289	.020	.308	.015	.276	.028	.000	.000	.196	.020
Newthyroid	**1.00**	**.000**	.153	.000	-.802	.595	.937	.000	.715	.099	.743	.134	.832	.206
Pima	**1.00**	**.000**	.892	.000	-.600	.800	.027	.000	.878	.062	.799	.391	.298	.333
Saheart	**.983**	**.000**	.454	.000	-	-	.072	.000	.078	.043	.685	.334	.285	.229
Segment*	**.868**	**.000**	.798	.000	-	-	.558	.051	.385	.094	.173	.123	.332	.046
Sonar	**.816**	**.000**	.364	.000	-	-	.032	.002	.018	.014	.000	.000	.429	.267
Soybean	-.036	.000	**1.00**	**.000**	.798	.251	**1.00**	**.000**	.822	.086	.000	.000	.658	.187
Spectfheart	**.885**	**.000**	.024	.000	-	-	.002	.000	-.093	.035	.000	.000	.037	.030
Tae	**.171**	**.000**	.010	.000	-	-	.116	.007	.052	.022	.050	.010	.022	.026
Vehicle	**.994**	**.000**	.606	.000	-	-	.080	.002	.120	.025	.317	.120	.025	.014
Wdbc	.993	.000	.883	.000	**.993**	**.000**	**.993**	**.000**	.906	.055	.000	.000	.495	.245
Wine	**1.00**	**.000**	.505	.000	-	-	.982	.000	.725	.086	.356	.104	.451	.136
Zoo	.437	.000	**.880**	**.000**	.677	.085	.669	.106	.708	.076	.332	.121	.454	.113
Average	**.698**	**.372**	.476	.326	-.447	.702	.434	.361	.363	.312	.274	.278	.337	.229

Table 7: Experimental results for the ARI measure obtained by 3SHACC and 6 previous approaches in the CS_{10} constraint set.

Unsat Results for CS_{10}														
Dataset	3SHACC		2SHACC		COPKM		LCVQE		RDPM		TVClust		PCSKM	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	10.90	0.00	16.36	0.00	-	-	21.81	0.00	32.72	5.07	38.54	10.96	**8.54**	**2.70**
Balance	**0.00**	**0.00**	56.47	0.00	-	-	14.71	1.86	27.91	2.35	1.55	1.77	9.97	3.03
Breast Cancer	**0.00**	**0.00**	16.60	0.00	50.00	50.00	4.07	0.00	4.74	1.82	45.99	2.30	11.96	9.15
Bupa	**0.00**	**0.00**	17.64	0.00	**0.00**	**0.00**	7.96	0.75	13.76	3.10	21.71	8.16	4.60	0.48
Ecoli	0.17	0.00	3.03	0.00	**0.00**	**0.00**	6.59	1.26	15.84	6.46	17.04	5.85	2.74	0.98
Glass	**0.00**	**0.00**	7.79	0.00	**0.00**	**0.00**	13.76	1.23	30.73	4.14	28.52	2.92	5.54	1.25
Haberman	**0.21**	**0.00**	38.92	0.00	-	-	37.20	0.00	30.90	11.92	19.22	1.82	13.97	5.31
Hayes Roth	**5.83**	**0.00**	26.66	0.00	-	-	14.50	4.93	27.83	3.12	29.58	8.03	7.25	1.23
Heart	**2.56**	**0.00**	8.26	0.00	-	-	14.81	0.00	36.09	4.10	36.92	14.75	15.98	3.28
Ionosphere	**0.31**	**0.00**	3.96	0.00	90.00	30.00	27.46	0.00	34.52	3.68	46.34	0.00	18.69	3.75
Iris	**0.00**	**0.00**	21.90	0.00	30.00	45.82	2.85	0.00	14.00	5.65	18.66	3.16	3.61	2.16
Monk2	**0.00**	**0.00**	10.35	0.00	-	-	11.20	0.00	46.14	0.81	32.56	22.17	13.03	7.14
Mov. Libras	**0.00**	**0.00**	0.47	0.00	**0.00**	**0.00**	2.09	0.51	11.63	1.36	92.69	0.00	0.60	0.23
Newthyroid	**0.00**	**0.00**	40.26	0.00	90.00	30.00	0.86	0.00	8.31	2.47	8.26	5.03	2.12	2.92
Pima	**0.00**	**0.00**	4.95	0.00	80.00	40.00	46.03	0.00	3.30	1.75	9.64	19.02	23.24	11.11
Saheart	**0.00**	**0.00**	19.14	0.00	-	-	39.31	0.00	42.14	2.73	11.69	16.30	20.16	7.26
Segment*	**0.87**	**0.00**	3.18	0.00	-	-	6.93	1.09	19.11	4.70	39.40	15.77	5.20	0.51
Sonar	**1.90**	**0.00**	12.38	0.00	-	-	32.38	0.00	40.14	2.33	49.52	0.00	6.09	4.92
Soybean	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	2.00	6.00	90.00	0.00	**0.00**	**0.00**
Spectfheart	**0.28**	**0.00**	28.20	0.00	-	-	33.90	0.00	41.28	2.52	34.47	0.00	18.49	0.92
Tae	7.50	0.00	61.66	0.00	-	-	14.08	0.25	48.75	8.55	47.91	8.69	**6.83**	**2.13**
Vehicle	**0.00**	**0.00**	16.86	0.00	-	-	30.38	0.08	29.66	4.21	26.76	4.99	16.18	0.43
Wdbc	**0.00**	**0.00**	5.95	0.00	**0.00**	**0.00**	**0.00**	**0.00**	3.46	2.63	45.86	0.00	14.88	7.34
Wine	**0.00**	**0.00**	9.15	0.00	-	-	**0.00**	**0.00**	8.62	3.15	23.79	8.33	5.42	1.37
Zoo	**0.00**	**0.00**	1.81	0.00	**0.00**	**0.00**	**0.00**	**0.00**	5.81	2.27	21.09	4.46	0.18	0.54
Average	**1.22**	**2.70**	17.28	16.40	65.60	44.00	15.31	13.96	23.18	14.73	33.51	21.52	9.41	6.67

Table 8: Experimental results for the Unsat measure obtained by 3SHACC and 6 previous approaches in the CS_{10} constraint set.

ARI Results for CS_{15}														
Dataset	3SHACC		2SHACC		COPKM		LCVQE		RDPM		TVClust		PCSKM	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**.957**	**.000**	-.070	.000	-	-	.443	.000	.265	.144	.050	.166	.587	.304
Balance	**1.00**	**.000**	.004	.000	**1.00**	**.000**	.612	.025	.509	.136	**1.00**	**.000**	.907	.121
Breast Cancer	**1.00**	**.000**	.435	.000	**1.00**	**.000**	.965	.000	.971	.021	.015	.040	**1.00**	**.000**
Bupa	**.611**	**.000**	.008	.000	.585	.008	.042	.008	.150	.046	.120	.071	.169	.096
Ecoli	**.947**	**.000**	.673	.000	-	-	.643	.051	.757	.034	.692	.155	.686	.112
Glass	**.797**	**.000**	.423	.000	-	-	.221	.018	.284	.029	.209	.072	.483	.117
Haberman	**1.00**	**.000**	-.004	.000	**1.00**	**.000**	-.002	.000	.708	.267	.986	.000	.547	.297
Hayes Roth	**.650**	**.000**	.024	.000	-	-	.042	.022	.088	.035	.124	.067	.063	.045
Heart	**1.00**	**.000**	.970	.000	**1.00**	**.000**	.687	.000	.213	.072	.333	.389	.689	.440
Ionosphere	**1.00**	**.000**	**1.00**	**.000**	**1.00**	**.000**	.656	.000	.319	.067	.000	.000	.827	.350
Iris	**1.00**	**.000**	.549	.000	-	-	.980	.000	.797	.137	.785	.211	.759	.169
Monk2	**1.00**	**.000**	.991	.000	**1.00**	**.000**	.725	.000	.037	.009	.416	.479	.901	.298
Mov. Libras	**.411**	**.000**	.328	.000	-	-	.304	.018	.271	.032	.000	.000	.210	.022
Newthyroid	**1.00**	**.000**	.250	.000	-.400	.917	**1.00**	**.000**	.982	.019	.876	.186	.918	.155
Pima	**1.00**	**.000**	.521	.000	**1.00**	**.000**	.017	.000	.937	.057	.804	.391	.904	.288
Saheart	**1.00**	**.000**	.729	.000	**1.00**	**.000**	-.001	.000	.230	.108	.767	.372	.799	.399
Segment*	**.997**	**.000**	.917	.000	-	-	.706	.041	.488	.044	.178	.036	.117	.034
Sonar	**.981**	**.000**	.003	.000	-	-	.590	.000	.016	.020	.000	.000	.796	.308
Soybean	.523	.000	**1.00**	**.000**	.200	.980	**1.00**	**.000**	.862	.110	.000	.000	.639	.165
Spectfheart	**1.00**	**.000**	.901	.000	**1.00**	**.000**	.010	.013	.008	.246	.000	.000	.362	.188
Tae	**.548**	**.000**	.051	.000	-	-	.103	.000	.068	.032	.058	.014	.046	.024
Vehicle	**1.00**	**.000**	.250	.000	**1.00**	**.000**	.066	.001	.314	.096	.354	.154	.086	.038
Wdbc	**1.00**	**.000**	.792	.000	**1.00**	**.000**	.972	.000	.947	.036	.108	.288	.999	.003
Wine	.983	.000	.295	.000	-	-	**1.00**	**.000**	.723	.094	.391	.132	.397	.206
Zoo	.813	.000	**.909**	**.000**	.697	.126	.683	.111	.735	.085	.361	.193	.499	.133
Average	**.889**	**.182**	.478	.378	.083	.935	.498	.375	.467	.334	.345	.341	.576	.310

Table 9: Experimental results for the ARI measure obtained by 3SHACC and 6 previous approaches in the CS_{15} constraint set.

Unsat Results for CS_{15}														
Dataset	3SHACC		2SHACC		COPKM		LCVQE		RDPM		TVClust		PCSKM	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**0.00**	**0.00**	53.33	0.00	-	-	19.16	0.00	29.25	7.09	40.75	11.14	6.08	5.97
Balance	**0.00**	**0.00**	57.35	0.00	**0.00**	**0.00**	16.56	1.22	17.61	5.24	**0.00**	**0.00**	2.78	3.70
Breast Cancer	**0.00**	**0.00**	27.57	0.00	**0.00**	**0.00**	1.91	0.00	1.06	0.72	46.17	1.78	**0.00**	**0.00**
Bupa	**0.00**	**0.00**	33.40	0.00	**0.00**	**0.00**	12.11	0.56	12.31	1.73	15.46	3.64	5.93	1.00
Ecoli	**0.00**	**0.00**	4.62	0.00	-	-	8.60	1.14	6.76	1.43	10.99	8.08	3.60	1.46
Glass	**0.00**	**0.00**	12.68	0.00	-	-	18.92	1.96	21.26	6.10	31.53	4.92	5.05	1.21
Haberman	**0.00**	**0.00**	39.03	0.00	**0.00**	**0.00**	46.28	0.00	9.94	10.47	0.19	0.00	13.85	9.46
Hayes Roth	**4.34**	**0.00**	50.36	0.00	-	-	30.25	2.24	29.85	3.62	29.20	7.79	10.76	1.22
Heart	**0.00**	**0.00**	0.97	0.00	**0.00**	**0.00**	11.82	0.00	35.64	4.89	31.14	18.84	9.37	13.51
Ionosphere	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	15.53	0.00	32.38	3.60	46.44	0.00	5.52	11.18
Iris	**0.00**	**0.00**	23.32	0.00	-	-	**0.00**	**0.00**	7.74	8.17	8.18	9.93	4.98	3.46
Monk2	**0.00**	**0.00**	0.28	0.00	**0.00**	**0.00**	12.78	0.00	45.48	1.01	28.56	23.49	3.22	9.66
Mov. Libras	**0.62**	**0.00**	7.89	0.00	-	-	3.94	0.34	9.71	1.33	94.20	0.00	1.23	0.25
Newthyroid	**0.00**	**0.00**	38.25	0.00	70.00	45.82	**0.00**	**0.00**	0.39	0.47	5.34	9.46	1.70	2.91
Pima	**0.00**	**0.00**	23.31	0.00	**0.00**	**0.00**	48.84	0.00	1.78	1.73	9.65	19.31	3.77	11.33
Saheart	**0.00**	**0.00**	12.29	0.00	**0.00**	**0.00**	50.51	0.00	33.61	5.15	11.39	18.44	6.99	13.91
Segment*	**0.00**	**0.00**	1.60	0.00	-	-	5.98	1.44	12.96	1.60	37.70	6.12	9.18	0.43
Sonar	**0.00**	**0.00**	49.59	0.00	-	-	20.56	0.00	41.69	2.26	51.41	0.00	4.61	8.35
Soybean	**0.00**	**0.00**	**0.00**	**0.00**	40.00	48.99	**0.00**	**0.00**	6.07	4.24	60.71	0.00	2.14	1.75
Spectfheart	**0.00**	**0.00**	3.41	0.00	**0.00**	**0.00**	41.26	0.41	37.22	8.61	31.34	0.00	18.02	5.27
Tae	**6.71**	**0.00**	48.61	0.00	-	-	33.59	0.00	39.92	7.82	44.66	3.18	13.00	1.12
Vehicle	**0.00**	**0.00**	41.77	0.00	**0.00**	**0.00**	34.84	0.07	19.98	2.86	27.25	6.35	20.46	1.02
Wdbc	**0.00**	**0.00**	10.17	0.00	**0.00**	**0.00**	1.01	0.00	2.06	1.35	41.22	13.09	0.06	0.13
Wine	**0.00**	**0.00**	34.47	0.00	-	-	**0.00**	**0.00**	7.37	1.86	28.43	9.54	8.49	3.18
Zoo	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	4.00	1.04	7.00	3.61	20.91	8.64	1.66	0.91
Average	**0.46**	**1.53**	22.97	19.69	44.40	47.84	17.54	16.25	18.76	14.29	30.11	20.85	6.50	5.24

Table 10: Experimental results for the Unsat measure obtained by 3SHACC and 6 previous approaches in the CS_{15} constraint set.

ARI Results for CS_{20}														
Dataset	3SHACC		2SHACC		COPKM		LCVQE		RDPM		TVClust		PCSKM	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**1.00**	**.000**	-.040	.000	-	-	.117	.000	.459	.097	.158	.282	.690	.381
Balance	**1.00**	**.000**	.174	.000	**1.00**	**.000**	.774	.056	.947	.093	**1.00**	**.000**	.974	.079
Breast Cancer	**1.00**	**.000**	.455	.000	**1.00**	**.000**	**1.00**	**.000**	.992	.007	.194	.389	**1.00**	**.000**
Bupa	.817	.000	.020	.000	**.844**	**.007**	.032	.006	.334	.110	.400	.083	.402	.151
Ecoli	**.988**	**.000**	.980	.000	-	-	.536	.060	.832	.034	.746	.134	.848	.057
Glass	**.945**	**.000**	.172	.000	-	-	.259	.035	.417	.123	.241	.133	.679	.140
Haberman	**1.00**	**.000**	-.015	.000	**1.00**	**.000**	-.002	.000	.879	.129	**1.00**	**.000**	.999	.004
Hayes Roth	.978	.000	.153	.000	**.990**	**.008**	.026	.020	.128	.045	.303	.360	.103	.090
Heart	**1.00**	**.000**	**1.00**	**.000**	**1.00**	**.000**	.857	.000	.515	.176	.502	.454	**1.00**	**.000**
Ionosphere	**1.00**	**.000**	**1.00**	**.000**	**1.00**	**.000**	.007	.000	.490	.086	.000	.000	**1.00**	**.000**
Iris	**1.00**	**.000**	.558	.000	-	-	.886	.000	.837	.203	.597	.141	.752	.167
Monk2	**1.00**	**.000**	**1.00**	**.000**	**1.00**	**.000**	.925	.009	.077	.032	.727	.423	**1.00**	**.000**
Mov. Libras	**.662**	**.000**	.165	.000	-	-	.304	.020	.291	.030	.000	.000	.191	.027
Newthyroid	**1.00**	**.000**	.104	.000	-.400	.917	.968	.000	.992	.014	.871	.169	.980	.026
Pima	**1.00**	**.000**	.917	.000	**1.00**	**.000**	.043	.000	.997	.002	**1.00**	**.000**	**1.00**	**.000**
Saheart	**1.00**	**.000**	.639	.000	**1.00**	**.000**	-.001	.000	.853	.020	.803	.394	**1.00**	**.000**
Segment*	**1.00**	**.000**	.925	.000	.200	.980	.641	.051	.675	.073	.286	.122	.064	.036
Sonar	**1.00**	**.000**	**1.00**	**.000**	-	-	-.003	.001	.244	.210	.000	.000	**1.00**	**.000**
Soybean	.159	.000	**1.00**	**.000**	.118	.925	**1.00**	**.000**	.829	.114	.000	.000	.569	.245
Spectfheart	**1.00**	**.000**	.592	.000	**1.00**	**.000**	-.004	.001	.197	.438	.000	.000	.879	.282
Tae	**.849**	**.000**	.027	.000	-	-	.048	.007	.069	.042	.055	.034	.066	.068
Vehicle	**1.00**	**.000**	.547	.000	**1.00**	**.000**	.072	.003	.802	.146	.539	.132	.963	.111
Wdbc	**1.00**	**.000**	.717	.000	**1.00**	**.000**	**1.00**	**.000**	.995	.005	.000	.000	**1.00**	**.000**
Wine	**1.00**	**.000**	.577	.000	-.800	.600	**1.00**	**.000**	.772	.060	.600	.232	.674	.320
Zoo	**.977**	**.000**	.968	.000	.846	.090	.777	.140	.740	.073	.350	.113	.515	.128
Average	**.935**	**.176**	.546	.385	.232	.893	.450	.416	.615	.312	.415	.348	.734	.324

Table 11: Experimental results for the ARI measure obtained by 3SHACC and 6 previous approaches in the CS_{20} constraint set.

Unsat Results for CS_{20}														
Dataset	3SHACC		2SHACC		COPKM		LCVQE		RDPM		TVClust		PCSKM	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
Appendicitis	**0.00**	**0.00**	40.69	0.00	-	-	41.12	0.00	19.91	3.16	32.38	11.22	8.09	10.23
Balance	**0.00**	**0.00**	45.53	0.00	**0.00**	**0.00**	10.65	2.63	1.88	3.62	**0.00**	**0.00**	0.93	2.79
Breast Cancer	**0.00**	**0.00**	26.92	0.00	**0.00**	**0.00**	**0.00**	**0.00**	0.30	0.26	39.37	19.02	**0.00**	**0.00**
Bupa	**0.00**	**0.00**	30.47	0.00	**0.00**	**0.00**	15.35	0.36	8.52	1.69	11.87	4.03	4.86	1.55
Ecoli	**0.00**	**0.00**	0.13	0.00	-	-	12.85	1.70	4.17	0.89	8.72	7.39	2.31	0.97
Glass	**0.00**	**0.00**	29.45	0.00	-	-	20.47	2.46	16.58	8.62	31.62	8.52	3.89	2.48
Haberman	**0.00**	**0.00**	48.59	0.00	**0.00**	**0.00**	47.96	0.00	4.32	5.06	**0.00**	**0.00**	0.05	0.17
Hayes Roth	0.20	0.00	38.10	0.00	**0.00**	**0.00**	35.32	1.67	29.37	2.11	29.45	17.81	18.14	2.47
Heart	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	6.70	0.00	20.47	8.61	24.83	22.58	**0.00**	**0.00**
Ionosphere	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	48.65	0.00	24.42	4.44	45.39	0.00	**0.00**	**0.00**
Iris	**0.00**	**0.00**	24.13	0.00	-	-	3.67	0.00	5.21	7.85	19.49	7.81	5.72	4.06
Monk2	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	3.37	0.44	42.20	1.81	13.49	20.90	**0.00**	**0.00**
Mov. Libras	**0.70**	**0.00**	27.30	0.00	-	-	4.89	0.38	9.89	0.88	94.56	0.00	1.67	0.13
Newthyroid	**0.00**	**0.00**	44.07	0.00	70.00	45.82	2.10	0.00	0.29	0.49	5.62	7.69	0.56	0.80
Pima	**0.00**	**0.00**	4.08	0.00	**0.00**	**0.00**	47.78	0.00	0.05	0.05	**0.00**	**0.00**	**0.00**	**0.00**
Saheart	**0.00**	**0.00**	17.48	0.00	**0.00**	**0.00**	49.11	0.00	6.28	0.58	9.28	18.56	**0.00**	**0.00**
Segment*	**0.00**	**0.00**	1.70	0.00	40.00	48.99	8.18	1.55	7.61	2.42	31.59	11.41	12.13	0.57
Sonar	**0.00**	**0.00**	**0.00**	**0.00**	-	-	47.54	0.61	31.26	9.17	51.68	0.00	**0.00**	**0.00**
Soybean	**0.00**	**0.00**	**0.00**	**0.00**	40.00	48.99	**0.00**	**0.00**	2.00	2.09	88.88	0.00	0.44	1.33
Spectfheart	**0.00**	**0.00**	19.49	0.00	**0.00**	**0.00**	51.82	0.67	30.07	16.49	33.47	0.00	4.47	10.49
Tae	**1.72**	**0.00**	53.33	0.00	-	-	40.71	0.45	39.37	7.69	46.96	5.74	19.31	1.48
Vehicle	*Do***0.00**	**0.00**	20.52	0.00	**0.00**	**0.00**	35.01	0.24	5.92	4.38	21.17	7.06	0.98	2.95
Wdbc	**0.00**	**0.00**	13.25	0.00	**0.00**	**0.00**	**0.00**	**0.00**	0.19	0.15	46.46	0.00	**0.00**	**0.00**
Wine	**0.00**	**0.00**	22.22	0.00	90.00	30.00	**0.00**	**0.00**	6.15	1.61	18.07	10.92	7.25	6.74
Zoo	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	3.09	2.20	7.66	2.05	24.28	5.14	3.66	1.12
Average	**0.10**	**0.35**	20.30	17.51	37.60	45.10	21.45	19.72	12.96	12.69	29.15	23.80	3.78	5.38

Table 12: Experimental results for the Unsat measure obtained by 3SHACC and 6 previous approaches in the CS_{20} constraint set.

References

[1] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[2] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[3] Baptiste Lafabregue, Jonathan Weber, Pierre Gançarski, and Germain Forestier. Deep constrained clustering applied to satellite image time series. In *ECML/PKDD Workshop on Machine Learning for Earth Observation Data (MACLEAN)*, Würzburg, Germany, 2019.

[4] Chao-Lung Yang and Thi Phuong Quyen Nguyen. Constrained clustering method for class-based storage location assignment in warehouse. *Industrial Management & Data Systems*, 116(4):667–689, 2016.

[5] Son T Mai, Sihem Amer-Yahia, Sébastien Bailly, Jean-Louis Pépin, Ahlame Douzal Chouakria, Ky T Nguyen, and Anh-Duong Nguyen. Evolutionary active constrained clustering for obstructive sleep apnea analysis. *Data Science and Engineering*, 3(4):359–378, 2018.

[6] R Hazratgholizadeh, MRF Derakhshi, MA Balafar, et al. Active learning for constrained document clustering with uncertainty region. *Complexity*, 2020:1–16, 2020.

[7] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 577–584. Morgan Kaufmann Publishers Inc., 2001.

[8] Xiaohua Xu, Lin Lu, Ping He, Zhoujin Pan, and Ling Chen. Improving constrained clustering via swarm intelligence. *Neurocomputing*, 116:317–325, 2013.

[9] Mahdieh Soleymani Baghshah and Saeed Bagheri Shouraki. Learning low-rank kernel matrices for constrained clustering. *Neurocomputing*, 74(12-13):2201–2211, 2011.

[10] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, MSR-TR-2000-65, Microsoft Research, 2000.

[11] Ian Davidson and Sugato Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from data*, 1:1–41, 2007.

[12] Jana Schmidt, Elisabeth Maria Brandle, and Stefan Kramer. Clustering with attribute-level constraints. In *2011 IEEE 11th International Conference on Data Mining*, pages 1206–1211. IEEE, 2011.

[13] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 521–528, 2003.

[14] Xuesong Yin, Songcan Chen, Enliang Hu, and Daoqiang Zhang. Semi-supervised clustering with metric learning: An adaptive kernel method. *Pattern Recognition*, 43(4):1320–1333, 2010.

[15] Ian Davidson and SS Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 59–70. Springer, 2005.

[16] Qiyue Yin, Shu Wu, Ran He, and Liang Wang. Multi-view clustering via pairwise sparse subspace representation. *Neurocomputing*, 156:12–21, 2015.

[17] Chuan Chen, Hui Qian, Wuhui Chen, Zibin Zheng, and Hong Zhu. Auto-weighted multi-view constrained spectral clustering. *Neurocomputing*, 366:1–11, 2019.

[18] Martin HC Law, Alexander Topchy, and Anil K Jain. Clustering with soft and group constraints. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 662–670. Springer, 2004.

[19] Ian Davidson and SS Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 International Conference on Data Mining*, pages 138–149. SIAM, 2005.

[20] Absalom E Ezugwu, Amit K Shukla, Moyinoluwa B Agbaje, Olaide N Oyelade, Adán José-García, and Jeffery O Agushaka. Automatic clustering algorithms: a systematic review and bibliometric analysis of relevant literature. *Neural Computing and Applications*, pages 1–60, 2020.

[21] Eric Bae and James Bailey. Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 53–62. IEEE, 2006.

[22] Dan Klein, Sepandar D Kamvar, and Christopher D Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. Technical report, Stanford, 2002.

[23] Li Zheng and Tao Li. Semi-supervised hierarchical clustering. In *2011 IEEE 11th International Conference on Data Mining*, pages 982–991. IEEE, 2011.

[24] Rudinei Martins de Oliveira, Antonio Augusto Chaves, and Luiz Antonio Nogueira Lorena. A comparison of two hybrid methods for constrained clustering problems. *Applied Soft Computing*, 54:256–266, 2017.

[25] Germán González-Almagro, Julián Luengo, José-Ramón Cano, and Salvador García. Dils: constrained clustering through dual iterative local search. *Computers & Operations Research*, page 104979, 2020.

[26] Germán González-Almagro, Alejandro Rosales-Pérez, Julián Luengo, José-Ramón Cano, and Salvador García. Improving constrained clustering via decomposition-based multiobjective optimization with memetic elitism. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 333–341, 2020.

[27] Julia Handl and Joshua Knowles. On semi-supervised clustering via multiobjective optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1465–1472, 2006.

[28] Zhiling Cai, Xiaofei Yang, Tianyi Huang, and William Zhu. A new similarity combining reconstruction coefficient with pairwise distance for agglomerative clustering. *Information Sciences*, 508:173–182, 2020.

[29] Germán González-Almagro, Juan Luis Suarez, Julián Luengo, José-Ramón Cano, and Salvador García. Agglomerative constrained clustering through similarity and distance recalculation. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 424–436. Springer, 2020.

[30] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[31] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[32] Wei Zhang, Xiaogang Wang, Deli Zhao, and Xiaoou Tang. Graph degree linkage: Agglomerative clustering on a directed graph. In *European Conference on Computer Vision*, pages 428–441. Springer, 2012.

[33] Wei Zhang, Deli Zhao, and Xiaogang Wang. Agglomerative clustering via maximum incremental path integral. *Pattern Recognition*, 46(11):3056–3065, 2013.

[34] Ian Davidson and S. S. Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. pages 59–70, 2005.

[35] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[36] Juan Luis Suárez, Salvador García, and Francisco Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing,*, 425:300–322, 2021.

[37] Xi Peng, Zhiding Yu, Zhang Yi, and Huajin Tang. Constructing the l2-graph for robust subspace learning and subspace clustering. *IEEE transactions on cybernetics*, 47(4):1053–1066, 2016.

[38] Isaac Triguero, Sergio González, Jose M Moyano, Salvador García, Jesús Alcalá-Fdez, Julián Luengo, Alberto Fernández, Maria José del Jesús, Luciano Sánchez, and Francisco Herrera. KEEL 3.0: an open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems*, 10(1):1238–1249, 2017.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[41] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[42] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[43] Jacinto Carrasco, Salvador García, María del Mar Rueda, and Francisco Herrera. rnpbst: An r package covering non-parametric and bayesian statistical tests. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 281–292. Springer, 2017.

[44] Jacinto Carrasco, Salvador García, MM Rueda, S Das, and Francisco Herrera. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, 54:100665, 2020.

[45] Dan Pelleg and Dorit Baras. K-means with large and noisy constraint sets. In *European Conference on Machine Learning*, pages 674–682. Springer, 2007.

[46] Daniel Khashabi, John Wieting, Jeffrey Yufei Liu, and Feng Liang. Clustering with side information: From a probabilistic model to a deterministic algorithm. *arXiv preprint arXiv:1508.06235*, 2015.

[47] Avgoustinos Vouros and Eleni Vasilaki. A semi-supervised sparse k-means algorithm. *Pattern Recognition Letters*, 142:65–71, 2021.

[48] Juan Luis Suárez, Salvador García, and Francisco Herrera. pydml: a python library for distance metric learning. *Journal of Machine Learning Research*, 21(96):1–7, 2020.

5 Semi-supervised Clustering with Two Types of Background Knowledge: Fusing Pairwise Constraints and Monotonicity Constraints

- **Journal:** Information Fusion (INFFUS)
- **JCR Impact Factor:** 17.564
- **Rank:** 4/145
- **Quartile:** Q1
- **Category:** Computer Science, Artificial Intelligence
- **Status:** Submitted

Ref.: González-Almagro, G., Suárez, J. L., Sánchez-Bermejo, P., Cano, J. R., & García, S. (2023). Semi-supervised Clustering with Two Types of Background Knowledge: Fusing Pairwise Constraints and Monotonicity Constraints. *arXiv preprint arXiv:2302.14060*.

A preliminary version of this paper was published at the 2022 International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems with title: Monotonic Constrained Clustering: A First Approach (DOI: https://doi.org/10.1007/978-3-031-08530-7_61). This conference was held in Kitakyushu, Japan.

Semi-supervised Clustering with Two Types of Background Knowledge: Fusing Pairwise Constraints and Monotonicity Constraints

Germán González Almagro [*,a,b] **Juan Luis Suárez** [,a,b]

Pablo Sánchez-Bermejo [a] **José-Ramón Cano** [c,b] **Salvador García** [a,b]

[a] *Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada, Granada, Spain*
[b] *Andalusian Institute of Data Science and Computational Intelligence (DaSCI)*
[c] *Department of Computer Science, University of Jaén, Jaén, Spain*

Abstract

This study addresses the problem of performing clustering in the presence of two types of background knowledge: pairwise constraints and monotonicity constraints. To achieve this, the formal framework to perform clustering under monotonicity constraints is, firstly, defined, resulting in a specific distance measure. Pairwise constraints are integrated afterwards by designing an objective function which combines the proposed distance measure and a pairwise constraint-based penalty term, in order to fuse both types of information. This objective function can be optimized with an EM optimization scheme. The proposed method serves as the first approach to the problem it addresses, as it is the first method designed to work with the two types of background knowledge mentioned above. Our proposal is tested in a variety of benchmark datasets and in a real-world case of study.

Keywords Pairwise constraints · Monotonicity constraints · Expectation-minimization · Semi-supervised learning · Machine learning

* Corresponding Author (germangalmagro@ugr.es)
Email addresses: jlsuarezdiaz@ugr.es (Juan Luis Suárez), pablo.snz.brm@gmail.com (Pablo Sánchez-Bermejo), julianlm@decsai.ugr.es (Julián Luengo), jrcano@ujaen.es (José Ramón Cano), salvagl@decsai.ugr.es (Salvador García)

1 Introduction

Clustering constitutes a key research area in the unsupervised learning paradigm, where no information on how data should be handled is available. It can be viewed as the task of grouping instances from a dataset into groups (or clusters), with the aim to extract new information from them [1]. From the classic K-means algorithm to the newer proposals [2], clustering has been applied to many problems, such as time series monitoring [3], COVID-19 medical image segmentation [4] and regular image segmentation [5], noisy speech processing [6] or band selection in hyperspectral images [7]. Background knowledge can be integrated into the classic clustering framework, thus reframing it into the semi-supervised learning paradigm [8, 9], where partial or incomplete information about the dataset is given to the perform clustering.

When additional information is given in the form of constraints, the constrained clustering problem arises. Constraints can be understood in three main ways: cluster-level [10], instance-level pairwise (or simply pairwise) [11] and feature-level constrained clustering [12]. This study focuses on pairwise constraints, which indicate whether two specific instances of a dataset must be placed in the same or in different clusters, resulting in Must-link (ML) and Cannot-link (CL) constraints, respectively. Constrained clustering has been applied in a variety of real world problems before, such as: satellite image time series [13], storage location assignment in warehouses [14], obstructive sleep apnea analysis [15] or electoral district design [16]. Recent studies and proposals, such as [17], prove the growing interest in the area of constrained clustering.

Recently, a new type of background knowledge coming from the supervised learning paradigm has been integrated into unsupervised learning. Monotonic classification is a particular case of supervised learning where classes are a set of ordered categories and classification models must respect monotonicity constraints among instances based on their descriptive features. This means that, if an instance x_i has greater feature values than those of instance x_j, its assigned class must also be higher (greater) in the ordering than that of x_j [18]. Considering the classic example of house pricing: for two houses in the same neighborhood, the bigger ones are constrained to have higher prices than smaller houses when the rest of the features of the houses are similar [19]. This defines an order relationship between houses (instances) based on the value of their features, therefore models predicting house prices must take this into account to produce accurate results. Monotonicity constraints are a type of background knowledge that can be used to produce more accurate predictive models [20], and has been successfully applied in real world problem such as fraudulent firm classification [21], real-time dynamic malware detection [22], or learning activities analysis based on students' opinion surveys [23]. Additionally, a recent study from The Alan Turing Institute states that considering underlying data monotonicity in data science/machine learning models leads to fairer applications [24].

In [18] a methodology to perform clustering in the presence of monotonicity information (ordered clustering) is proposed within the Multi Criteria Decision Aid (MCDA) framework. It is done by defining a distance measure based on the concept of preference, which is later

explained in detail (Section 2.3), and whose basic concept relies on comparing instances from the dataset, discriminating feature-level comparative relationships. This results in a distance measure that produces ordered labeling in terms of monotonicity, understanding it as in the monotonic classification models which have previously been described.

This study addresses the fusion of the two types of background knowledge mentioned above: pairwise constraints and monotonicity constraints. It extends a previous study by the same authors [25] in both the theoretical background of the proposed method and the testing of its capabilities. A real-world application is also presented in this study, addressing the Shanghai Ranking of World Universities (SRWU) dataset from a new perspective. A previous study which combines monotonicity constraints and cluster-size constraints (capacitated clustering) can be found in [26], where researchers are motivated by the existence of problems in which both types of background knowledge is available. This constitutes evidence in favor of the interest in the combination of different types of background knowledge, as there are real-world problems in which background knowledge is given in a heterogeneous fashion. Following this trend, our research is motivated by the existence of real-world problems in which monotonicity constraints and pairwise constraints are available, such as the SRWU partitioning problem. To the best of our knowledge, there is no previous research on this topic, as models to perform ordered clustering have emerged very recently. In this study, the logical relationship between monotonic classification and ordered clustering is tackled, producing the monotonic clustering paradigm, in which pairwise constraints are later included, resulting in Monotonic Constrained Clustering (MCC). An expectation-minimization (EM) scheme is proposed to optimize a hybrid objective function which fuses both monotonicity and pairwise constraints. The proposed hybrid objective function is composed of a monotonic distance metric and a penalty term for pairwise constraints violations. The overall proposed optimization method for MCC is coined as Pairwise Constrained K-Means - Monotonic (PCKM-Mono).

The rest of this study is organized as follows: background concerning classic clustering, pairwise constrained clustering, monotonic classification and ordered clustering, which is presented in Section 2 and whose content is later used in Section 3 to introduce the proposed MCC method. Once the experimental setup used to carry out our experiments is presented in Section 4, Sections 5 and 6 report and analyze the experimental results obtained by the proposed method. A real-world case of study is carried out in Section 7, where our proposal is used to perform clustering on the SRWU dataset and compare the results obtained by other methods in the same task. Lastly, our conclusions are discussed in Section 8.

2 Background

As stated before, partitional clustering is the action of grouping instances of a dataset into k clusters. A dataset $X = \{x_1, \cdots, x_n\}$ contains n instances, each one described by u features. The ith instance from X is noted as $x_i = (x_{[i,1]}, \cdots, x_{[i,u]})$. The goal of a clustering algorithm is to assign a class label l_i to each instance in X. The result is a list of labels $L = [l_1, \cdots, l_n]$, with $l_i \in \{1, \cdots, k\}\ \forall i \in \{1, \cdots, n\}$, that effectively splits X into k non-overlapping clusters c_i

to form a partition called $C = \{c_1, \cdots, c_K\}$. The label associated with a given cluster c_i can be accessed as $l(c_i)$. The cluster membership of every instance is determined by the similarity of the instance to the rest of instances in the same cluster, and the dissimilarity to instances in other clusters. Many types of distance measurements can be used to determine pairwise similarities [27].

2.1 Constrained Clustering

In real world applications, it is common to have some information about the analyzed datasets, even if this information is not given in the form of labels. In pairwise constrained clustering, a set of constraints is given to guide the clustering process. Constraints involve pairs of instances, indicating whether they must or must not belong to the same cluster; thus, two types of pairwise constraints can be formalized:

- Must-link (ML) constraints $C_{=}(x_i, x_j)$: instances x_i and x_j from X must be placed in the same cluster.
- Cannot-link (CL) constraints $C_{\neq}(x_i, x_j)$: instances x_i and x_j from X cannot be assigned to the same cluster.

It is known that ML constraints are transitive, reflexive and symmetrical, and therefore they constitute an equivalence relationship. This is not the case for CL constraints; however, they can be chained to deduce new ML constraints [28]. Pairwise constraints can be enforced in two ways: hard [28] and soft [29] constraints. The former must necessarily be satisfied in the output partition of any algorithm which makes use of them, while the latter are interpreted as strong suggestions by the algorithm but can be only partially satisfied in the output partition.

In CC (Constrained Clustering), the goal is to find a partition (clustering) of k clusters such that $C = \{c_1, \cdots, c_k\}$ of X, ideally satisfying all constraints (in hard CC) or as many constraints as possible (in soft CC). The classic clustering requirements also have to be observed: it must be fulfilled that the sum of instances in each cluster c_i is equal to the number of instances in X, which has been defined as $n = |X| = \sum_{i=1}^{k} |c_i|$.

2.2 Monotonicity Constraints in Classification

Monotonicity constraints were originally integrated into the supervised learning classification task, leading to monotonic classification. It can be viewed as a special case of standard classification where the classes constitute a set of ordered categories. Monotonic classification models must respect monotonicity constraints between the feature values of the instances and their class labels [20].

Formally, monotonic classification aims to predict the class label y_i from an instance x_i with $y \in \mathcal{Y} = \{l_1, \cdots, l_m\}$. The categories in $\mathcal{Y}$ are arranged in an order relation $\prec$ such as $l_1 \prec l_2 \prec \cdots \prec l_m$. In doing so, features and class labels are monotonically constrained by the problem

background knowledge i.e. $x_i \succeq x_j \rightarrow f(x_i) \geq f(x_j)$ where $x_i \succeq x_j$ implies that all features in x_i compare to features in x_j with operator $\geq$, this is: $x_{i,q} \geq x_{j,q}\ \forall q \in \{1, \cdots, u\}$ [30]. This given relationship between instances referred as dominance. In this case x_1 dominates x_2. The goal of monotonic classification is to build a classifier that does not violate monotonicity constraints (pairwise dominance relationships). The result is a monotonic classifier [20].

Much in the same way as it is done with constrained clustering methods, a distinction can be done in monotonic classifiers: soft monotonic models try to minimize the number of monotonic constrains violation, while hard monotonic models always produce monotonic predictions (never violate monotonic constraints) [19].

2.3 Partially Ordered Data Clustering in MCDA

In [18] the monotonicity constraints are integrated into unsupervised learning to produce the ordered clustering framework. Particularly, they are integrated into the MCDA paradigm, which is a subfield of operational research that concerns the structuring and solving decision problems including multiple criteria [31]. To do so, the classic symmetrical notion of distance in pattern recognition is replaced with the asymmetrical notion of preference from the MCDA paradigm. The preference of an instance over another evaluates the global advantages of the former over the latter with respect to some preference criteria. The notion of preference can be seen as a decomposition of a distance measure, taking into account the sign of the differences. To cluster instances in an MCDA context, the similarity between every pair of instances is evaluated in terms of preferences taking all the other alternatives into account. With this in mind, two instances are similar if they are preferred to or by the same set of instances. To formalize these concepts, let us consider the weighted L_1 distance (for the maximization case and without loss of generality) as in Equation 1, which can be simplified as in Equation 2, with $w_d \in [0, 1]$ being the weight assigned to the dth feature.

$$L_1(x_i, x_j) = \sum_{d=1}^{u} w_d |x_{[i,d]} - x_{[j,d]}|. \tag{1}$$

$$L_1(x_i, x_j) = \sum_{d:x_{[i,d]}>x_{[j,d]}}^{u} w_d x_{[i,d]} - w_d x_{[j,d]} + \sum_{d:x_{[j,d]}>x_{[i,d]}}^{u} w_d x_{[j,d]} - w_d x_{[i,d]}. \tag{2}$$

Subsequently, let us define the preference of x_i over x_j as in Equation 3. To put this into words, $r(x_i, x_j)$ quantifies the sum of differences between x_i and x_j limited to the features in which x_i has higher (lower) values than x_j for the maximization (minimization) case. Intuitively, the preference $r(x_i, x_j)$ indicates the cumulative quantified value of the advantage of x_i over x_j. Please note that, as it has already been mentioned, the preference is not symmetrical: $r(x_i, x_j) \neq r(x_j, x_i)$ in most cases.

$$r(x_i, x_j) = \sum_{d:x_{[i,d]}>x_{[j,d]}}^{u} w_d x_{[i,d]} - w_d x_{[j,d]}. \tag{3}$$

Finally, note that the weighted L_1 distance between two instances can always be expressed as in Equation 4. This decomposition can be done the same way for any L_p distance.

$$L_1(x_i, x_j) = r(x_i, x_j) + r(x_j, x_i). \tag{4}$$

3 The Proposal: Pairwise Constrained Monotonic Clustering

In this study, the combination of pairwise constraints and monotonicity constraints is investigated. Bearing in mind all formal concepts from monotonic classification and ordered clustering (from Section 2), establishing a logical relation between the concepts of dominance and preference is straightforward. This is: if an instance x_i dominates x_j, then it is also true that instance x_i is preferred over x_j (for uniform weights). More formally: $x_i \succeq x_j \rightarrow r(x_i, x_j) \geq r(x_j, x_i)$. This way, any distance L_p defined as in Equation 4 can be used to measure distances in clustering methods for them to produce output partition satisfying monotonicity constraints. This new clustering paradigm is coined as monotonic clustering.

To perform pairwise constrained monotonic clustering, an Expectation-Minimization (EM) optimization scheme is used, along with a hybrid objective function which takes into account both pairwise constraints and monotonicity constraints. To this end, a distance measure designed on the basis of the definition of preference (originally used in ordered clustering), and a pairwise constraint-based penalty term are combined to produce the already mentioned function. We named this approach Pairwise Constrained K-Means - Monotonic (PCKM-Mono).

The EM optimization scheme is widely used in the literature to approach clustering problems ranging from classic clustering problems to constrained clustering [32] and monotonic clustering [18]. Two steps build the EM optimization scheme: (1) in the Expectation step (E step), given a set of cluster representatives (centroids) $\{\mu_1, \cdots, \mu_K\}$, every instance x_i is assigned to the cluster c_j that minimizes its contribution to the objective function, computed with respect to the cluster representatives; (2) in the Minimization step (M step), the cluster representatives $\{\mu_1, \cdots, \mu_K\}$ are reestimated for the current cluster assignment $\{c_1, \cdots, c_K\}$ to minimize the objective function. The EM optimization scheme iterates between these two steps until some convergence criteria are met. With this in mind, two elements need to be defined in order to apply the EM scheme to the constrained monotonic problem: the objective function and the centroid computation criteria.

Cost function. The cost function of the proposed PCKM-Mono algorithm combines two main elements: a monotonic distance measure (proposed in [18]) and a pairwise constraint-based penalty term. Equation 5 defines the hybrid objective function optimized by PCKM-

Mono, where $\mathbb{1}[\![\cdot]\!]$ is the indicator function (returns 1 if the predicate given as argument holds, and 0 otherwise), and μ_k is the centroid associated with cluster k. The first term in Equation 5 is a preference-based distance metric, while the other two terms refer to the cost of violating CL and ML constraints (the penalty term), respectively. Please note that the first term of Equation 5 produces completely stratified clusters when applied alone, which would produce perfectly monotonic partitions. However, this is not a desirable result in most real-world problems, as will be proved in Section 5.

$$J_{PCKMM} = \begin{array}{c} \frac{1}{K}\sum_{k=1}^{K}\sum_{x_i\in c_k}|(r(x_i,\mu_k)-r(\mu_k,x_i))|+ \\ \sum_{(x_i,x_j)\in C_=}\mathbb{1}[\![l_i\neq l_j]\!]+\sum_{(x_i,x_j)\in C_{\neq}}\mathbb{1}[\![l_i=l_j]\!] \end{array}. \tag{5}$$

This cost function can be translated into an assignation rule as in Equation 6, which can be intuitively interpreted as: assign each instance to its closest (preferred) cluster among those where it produces the least violated constraints.

$$x_i \in c_{h^*} \ \textbf{if} \ h^* = \begin{array}{c} \texttt{argmin}_h\Big(|\sum_{j=1}^{u}(x_{[i,j]}-\mu_{[h,j]})|+ \\ \sum_{x_j:(x_i,x_j)\in C_=}\mathbb{1}[\![l(c_h)\neq l_j]\!]+\sum_{x_j:(x_i,x_j)\in C_{\neq}}\mathbb{1}[\![l(c_h)=l_j]\!]\Big) \end{array}. \tag{6}$$

Centroid update rule. Regarding the computation of the centroid for every cluster after the E step, it is done by following its traditional form: every centroid is computed as the average of all instances which belong to the cluster it represents. This can be formalized as in Equation 7.

$$\mu_i = \frac{1}{|c_i|}\sum_{x_i\in c_i} x_i \tag{7}$$

The overall PCKM-Mono optimization procedure is summarized in Algorithm 1. It is clear that the proposed method is soft constrained for both pairwise constraints and monotonicity constraints.

4 Experimental Setup and Calibration

In order to evaluate the capabilities of our proposal and compare its performance with previous methods, monotonic datasets need to be used. In [19] a list of 12 monotonic datasets is used to test the capabilities of monotonic methods. These are the datasets used in our experiments, which can be found in the Keel-dataset repository[1] [33] and are used in recent research concerning monotonic classification [34]. Three constraint sets with incremental levels of constraint-based information are generated for each dataset. Since the Euclidean

[1] https://sci2s.ugr.es/keel/category.php?cat=clas

Algorithm 1: Pairwise Constrained K-Means - Monotonic (PCKM-Mono)

Input: Dataset X, constraint sets $C_{=}$ and $C_{\neq}$, the number of clusters K.
Output: Partition C of K non-overlapping clusters.

[1] Initialize centroids $\{\mu_1, \cdots, \mu_K\}$ randomly
[2] **do**
    `// Expectation Step`
[3] **for** $i \in \{i, \cdots, n\}$ **do**
[4] Assign each instance x_i to cluster h^* following Equation 6.
[5] **end**
    `// Minimization Step`
[6] **for** $i \in \{i, \cdots, K\}$ **do**
[7] $\mu_i = \frac{1}{|c_i|} \sum_{x_i \in c_i} x_i$
[8] **end**
[9] **while** *not converged*;
[10] **return** C

distance is used to measure pairwise distances in all compared algorithms, a standardization procedure is applied to all datasets. No other preprocessing step is performed on the datasets.

Constraints are generated following the method in [28]. Three constraint sets are generated for every datasets, namely: CS_{10}, CS_{15} and CS_{20}. Each constraint set is associated with a small percentage of the size of the dataset: 10%, 15% and 20%, respectively. The formula $(n_f(n_f-1))/2$ tells us how many artificial constraints will be created for each constraint set, with n_f being the fraction of the size of the dataset associated with each of these percentages. Table 1 displays a summary of all datasets and constraint sets used in our experiments.

4.1 Evaluation Method and Validation of Results

Given the hybrid nature of our proposal, different features of the obtained partitions results have to be inspected to assess their quality in terms of different measures. The Adjusted Rand Index (ARI) will be used to measure the overall degree of agreement between the obtained partitions and the ground truth [35]. The Rand Index measures the degree of agreement of two partitions C_1 and C_2 for the same given dataset X, with C_1 and C_2 viewed as collections of $n(n-1)/2$ pairwise decisions. This measure is corrected for chance to obtain the ARI. For more details on ARI, see [35]. An ARI value of 1 indicates total agreement between C_1 and C_2, while -1 means total disagreement. The quality with respect to the monotonicity of the obtained partition can be measured with the Non-Monotonic Index (NMI), which measures the degree to which monotonicity constraints are violated. It is defined as the rate of violations of monotonicity divided by the total number of examples in a dataset [36]. Finally, the *Unsat* measure is used to evaluate the quality of the results from the point of view of constrained clustering. *Unsat* is computed as the rate of violated constraints in a given partition [37].

Dataset	Instances	Classes	Features	CS_{10}		CS_{15}		CS_{20}	
				ML	CL	ML	CL	ML	CL
Artiset	899	10	2	494	3422	1240	7671	2061	13870
Balance	625	3	4	832	1059	1799	2479	3332	4418
BostonHousing4CL	506	4	13	284	941	686	2089	1266	3784
Car	1728	4	6	7961	6745	18167	15244	32076	27264
ERA	1000	9	4	676	4274	1562	9613	2760	17140
ESL	488	9	4	216	912	521	2107	949	3707
LEV	1000	5	4	1381	3569	3174	8001	5692	14208
MachineCPU	209	4	6	41	149	99	366	205	615
Qualitative Bankruptcy	250	2	6	35	147	153	344	322	617
SWD	1000	4	10	1566	3384	3674	7501	6583	13317
Windsor Housing	546	2	11	915	516	2105	1135	3827	2059
Wisconsin	683	2	9	1273	1005	2834	2317	5146	4034

Table 1: Datasets and Constraint Sets Summary

Bayesian statistical tests are used in order to validate the results (which will be presented in Section 5), instead of using the classic Null Hypothesis Statistical Tests (NHST), whose disadvantages are analyzed in [38], where a new statistical comparative framework is also proposed. The Bayesian version of the frequentist non-parametric sign test is used in this study. In the Bayesian sign test, the statistical distribution of a given parameter ρ is obtained according to the differences between two sets of results, assuming it is a Dirichlet distribution. To do so, the Bayesian sign test proceeds as follows: the number of times that $A-B<0$, the number of times where there are no significant differences, and the number of times that $A-B>0$, then the weights of the Dirichlet distribution are iteratively updated and finally sampled to obtain a large sample of the distribution. In order to identify cases where there are no significant differences, the region of practical equivalence (rope) $[r_{\min}, r_{\max}]$ is defined, so that $P(A \approx B) = P(\rho \in \text{rope})$. The result of this process is a set of triplets with the form described in Equation 8. The `rNPBST` R package is employed to apply the test, whose documentation and guide can be found in [39].

$$[P(\rho < r_{\min}) = P(A-B<0),\ \ P(\rho \in \text{rope})P(\rho > r_{\max}) = P(A-B>0)]\ . \tag{8}$$

4.2 Calibration

To demonstrate the capabilities of the proposed PCKM-Mono algorithm, it is compared with four other previous EM-style clustering algorithms, including the only existing purely monotonic clustering algorithm, two purely constrained clustering algorithms (including the most recent one), and a classic clustering algorithm:

- **P2Clust**: The first approach to monotonic clustering. It modifies the distance measure used in the expectation step of the EM scheme to produce purely monotonic partitions. Monotonicity constraints are never violated in partitions produced by P2Clust [18], thus it is a hard constrained method for monotonicity constraints. It does not consider pairwise constraints, therefore it is purely monotonic.

- **COP-Kmeans**: COnstrained Partitional K-means constitutes the first approach to constrained clustering [28]. It is taken as the baseline comparison for any constrained clustering method. To integrate constraints into the clustering process, it modifies the assignment rule of instances to a cluster in such a way that no constraints can be violated. The algorithm halts when a dead-end is reached. It produces partitions which satisfy all constraints when it does not arrive at dead-ends, thus it is a hard constrained method for pairwise constraints. It a purely constrained clustering algorithm.

- **Kmeans**: The original Kmeans algorithm proposed in [40]. Neither pairwise constraints nor monotonicity constraints are considered in Kmeans.

- **PCSKMeans**: The Pairwise Constrained Sparse K-Means algorithm is an extension of the classic Sparse K-Means algorithm that integrates constraints by means of a weighted penalty term [32]. It constitutes the most recent EM-style approach to constrained clustering.

Regarding the parameter setup, all algorithms use an EM scheme to find a partition of the datasets, thus sharing many of their parameters. The k parameter, which indicates the number of clusters of the output partition is always set to the number of classes for every dataset (in Table 1). The maximum number of iterations allowed before convergence is set to 100 in all cases. The convergence criterion is centroid shifting: the EM optimization procedure is considered to have converged when average centroid shifting is less than 10^{-4}. Random centroid initialization is used for all algorithms. The P2Clust algorithm allows us to parameterize the computation of its internal α coefficient; this parameter is set to 1.1. The sparsity level of the PCSKMeans algorithm is set to 1.1. All parameters have been set by following the guidelines of the authors, and PCKM-Mono parameters have been decided upon preliminary experimentation. The final purpose of this work is to provide a fair comparison between algorithms, assessing their robustness in a common environment with multiple datasets.

5 Experimental Results

The experimental results obtained for all datasets and constraint sets are presented in this section. Since non-deterministic procedures are present in every compared method (such as the random initialization of centroids), the average results of 50 runs are presented in Tables 2, 3 and 4, aiming to mitigate the effects that stochastic procedures may cause. Please note that, in cases where the COP-Kmeans algorithm is not able to produce a partition, we

assign that particular run the worst possible benchmark values. Cases where no result is reported are cases in which COP-Kmeans was never able to produce an output partition. Let us remember that ARI is a maximization external quality index, while NMI and Unsat are both for minimization.

Two types of diagrams are used to visualize the numerical results contained in the tables. On the one hand, Figures 1, 3 and 5 are used to compare average results for constraint sets CS_{10}, CS_{15}, CS_{20}, respectively. They are referred to as violinplots. They allow for a quick view of the distribution of results achieved by each method, as they contain a boxplot in addition to the outer violinplot. On the other hand, Figures 2, 4 and 6 are used to show detailed results for every method and every dataset. They can be used to understand how every methods behaves in every dataset, bringing to light their strengths and weaknesses.

By examining the results, it seems obvious that the proposed algorithm, PCKM-Mono, is able to find a balance between constraint satisfaction and the monotonicity of the output partition. Clearly P2Clust, which is a purely monotonic algorithm, always produces the best results with respect to NMI, as shown in Figures 1b, 3b, and 5b. Similarly, Figures 1a, 3a, and 5a show how purely constrained clustering algorithms (COP-Kmeans and PCSKMeans) produce the best results with respect to Unsat. However, PCKM-Mono is able to produce the best average ARI results (see Figures 1c, 3c, and 5c), while also achieving better NMI results than purely constrained clustering algorithms, and better Unsat results than purely monotonic clustering algorithms. This is indicative of the viability of the combination of pairwise and monotonic constraints to solve benchmark problems in both areas; moreover, it provides evidence in favor of the proposed EM optimization scheme, which is simple but can be, nonetheless, suitable for this task.

Some of the particular numerical results are worth noting, for example: the COP-Kmeans algorithm achieves near-optimum results for the CS_{10} constraint set. The reason for this being that, the lower the number of constraints, the easier it is for the algorithm to find a feasible partition, which is usually a very accurate partition in the case of COP-Kmeans. With regard to the results obtained by PCKM-Mono for Unsat and NMI, both are observed to be stable with the increasing amount of constraint based information, while the ARI is observed to scale with it (although not in a consistent manner). Please note that, the results obtained by Kmeans and P2Clust are practically identical, independently of the constraint set, which is a virtually average result, as they are not affected at all by constraints.

Dataset	ARI (↑)					NMI (↓)					Unsat (↓)				
	PCKM-Mono	P2Clust	COP-KMeans	KMeans	PCSKMeans	PCKM-Mono	P2Clust	COP-KMeans	KMeans	PCSKMeans	PCKM-Mono	P2Clust	COP-KMeans	KMeans	PCSKMeans
Artiset	**1.00**	.366	-	.241	.995	.020	**.000**	-	.810	.189	**.000**	.136	-	.160	**.000**
Balance	.016	.005	**1.00**	.146	**1.00**	.610	**.000**	.767	.914	.631	.053	.477	**.000**	.406	**.000**
Bostonhousing4Cl	.123	.122	**1.00**	.124	.657	**.000**	**.000**	**.000**	**.000**	**.000**	.091	.332	**.000**	.356	.005
Car	.825	.036	**1.00**	.112	.993	.058	**.000**	.057	.128	.164	.008	.500	**.000**	.458	**.000**
ERA	.996	.013	-	-.045	**.997**	1.00	**.000**	-	1.00	1.00	**.000**	.256	-	.283	**.000**
ESL	**.979**	.281	.975	.241	.974	.584	**.000**	.653	1.00	.584	.001	.210	**.000**	.206	**.000**
LEV	**1.00**	-.224	**1.00**	.071	.999	.989	**.000**	.971	1.00	.971	**.000**	.558	**.000**	.345	**.000**
MachineCPU	.156	.159	**.987**	.224	.196	.143	**.000**	.258	.364	.258	.034	.385	**.000**	.339	.011
Qualitative Bankruptcy	**1.00**	.665	-.300	.934	**1.00**	**.000**	**.000**	.650	**.000**	**.000**	**.000**	.143	.650	.035	**.000**
SWD	.217	.111	**1.00**	.066	.955	.947	**.000**	.947	.973	.933	.093	.380	**.000**	.390	.001
Windsor Housing	.984	.073	.994	.064	**1.00**	**.000**	**.000**	**.000**	**.000**	**.000**	.001	.452	**.000**	.452	**.000**
Wisconsin	**1.00**	.857	**1.00**	.849	**1.00**	.009	**.000**	.009	.764	.009	**.000**	.070	**.000**	.074	**.000**
Average	.691	.205	.555	.252	**.897**	.363	**.000**	.526	.579	.395	.023	.325	.221	.292	**.001**

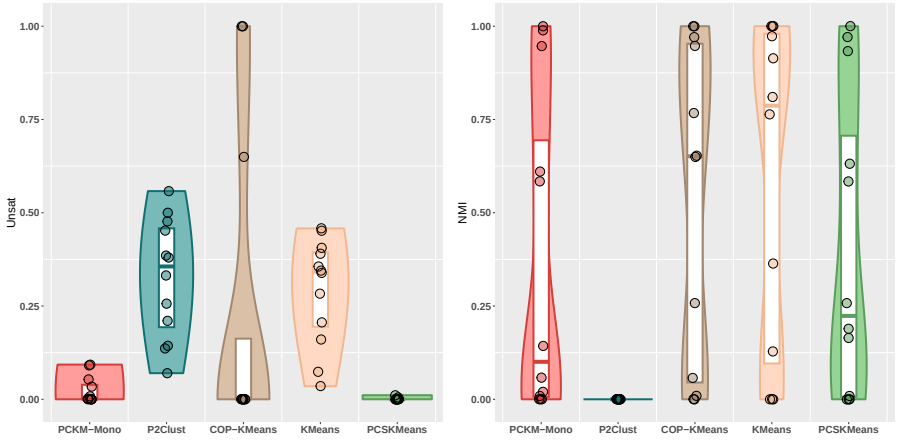
Table 2: Results obtained by the five compared methods for the CS_{10} constraint set.

Dataset	ARI (↑)					NMI (↓)					Unsat (↓)				
	PCKM-Mono	P2Clust	COP-KMeans	KMeans	PCSKMeans	PCKM-Mono	P2Clust	COP-KMeans	KMeans	PCSKMeans	PCKM-Mono	P2Clust	COP-KMeans	KMeans	PCSKMeans
Artiset	.591	.370	**1.00**	.244	.404	.182	**.000**	**.000**	.999	.957	.002	.134	**.000**	.157	.001
Balance	**1.00**	.005	**1.00**	.140	**1.00**	.566	**.000**	.790	.914	.778	**.000**	.482	**.000**	.392	**.000**
Bostonhousing4Cl	.999	.122	**1.00**	.127	.989	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	.342	**.000**	.349	**.000**
Car	.999	.029	**1.00**	.113	**1.00**	.057	**.000**	.057	.120	.131	**.000**	.501	**.000**	.461	**.000**
ERA	**.998**	.012	-	-.070	.573	1.00	**.000**	-	1.00	1.00	**.000**	.252	-	.307	**.000**
ESL	**.995**	.273	.993	.245	.366	.403	**.000**	.626	.998	.594	**.000**	.209	**.000**	.211	.001
LEV	**.934**	-.225	.250	.062	.896	.971	**.000**	.982	1.00	.985	.003	.555	.375	.340	**.000**
MachineCPU	**1.00**	.159	**1.00**	.216	.803	.212	**.000**	.258	.349	.258	**.000**	.344	**.000**	.377	**.000**
Qualitative Bankruptcy	**1.00**	.665	.689	.935	**1.00**	**.000**	**.000**	.132	**.000**	**.000**	**.000**	.173	.125	.038	**.000**
SWD	.997	.114	**1.00**	.068	**1.00**	.947	**.000**	.947	.963	.947	.001	.381	**.000**	.394	**.000**
Windsor Housing	**1.00**	.073	-	.058	.992	**.000**	**.000**	-	**.000**	**.000**	**.000**	.464	-	.451	**.000**
Wisconsin	**1.00**	.857	-	.848	**1.00**	.009	**.000**	-	.764	.009	**.000**	.069	-	.075	**.000**
Average	**.960**	.205	.411	.249	.835	.362	**.000**	.566	.592	.472	.001	.326	.292	.296	**.000**

Table 3: Results obtained by the five compared methods for the CS_{15} constraint set.

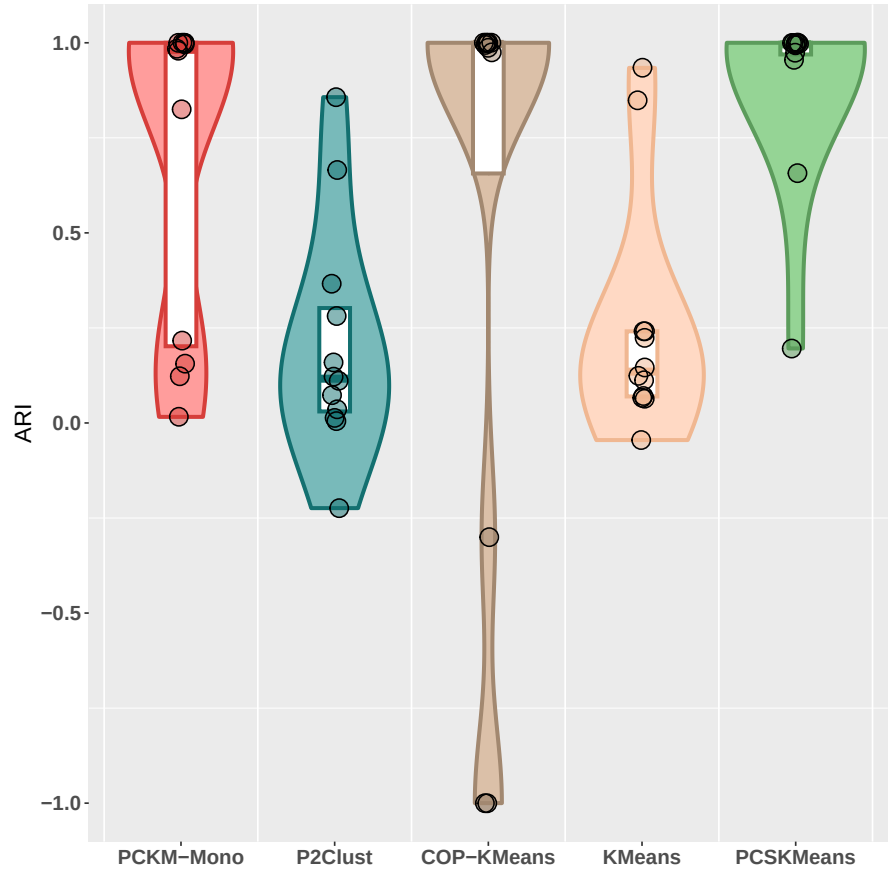
Dataset	ARI (↑)					NMI (↓)					Unsat (↓)				
	PCKM-Mono	P2Clust	COP-KMeans	KMeans	PCSKMeans	PCKM-Mono	P2Clust	COP-KMeans	KMeans	PCSKMeans	PCKM-Mono	P2Clust	COP-KMeans	KMeans	PCSKMeans
Artiset	.941	.364	-.950	.239	**1.00**	.194	**.000**	.976	.793	.855	.001	.133	.975	.166	**.000**
Balance	**1.00**	.004	**1.00**	.136	**1.00**	.509	**.000**	.831	.914	.825	**.000**	.479	**.000**	.407	**.000**
Bostonhousing4Cl	**1.00**	.122	-	.128	**1.00**	**.000**	**.000**	-	**.000**	**.000**	**.000**	.342	-	.340	**.000**
Car	.999	.030	-	.107	**1.00**	.057	**.000**	-	.122	.063	**.000**	.504	-	.464	**.000**
ERA	.040	.012	.034	-.017	**.999**	.986	**.000**	.999	1.00	.999	.019	.252	**.000**	.262	**.000**
ESL	.295	.303	.246	.244	**.994**	.405	**.000**	.482	1.00	.759	**.000**	.189	**.000**	.213	**.000**
LEV	**.997**	-.229	-	.078	.012	.971	**.000**	-	.996	.973	**.000**	.551	-	.343	.001
MachineCPU	**.987**	.159	.231	.216	.949	.258	**.000**	.241	.359	.258	**.000**	.365	**.000**	.406	**.000**
Qualitative Bankruptcy	.727	.665	**1.00**	.938	.937	**.000**	**.000**	**.000**	.116	.031	.106	.151	**.000**	.031	**.000**
SWD	.998	.113	-	.066	**1.00**	.947	**.000**	-	.963	1.00	.001	.385	-	.397	**.000**
Windsor Housing	.286	.073	**1.00**	.027	.133	**.000**	**.000**	**.000**	**.000**	.002	.190	.466	**.000**	.456	.006
Wisconsin	.945	.857	**1.00**	.848	.836	.008	**.000**	.009	.041	.009	.016	.068	**.000**	.071	.004
Average	.768	.206	-.037	.251	**.822**	.361	**.000**	.628	.525	.481	.028	.324	.415	.296	**.001**

Table 4: Results obtained by the five compared methods for the CS_{20} constraint set.



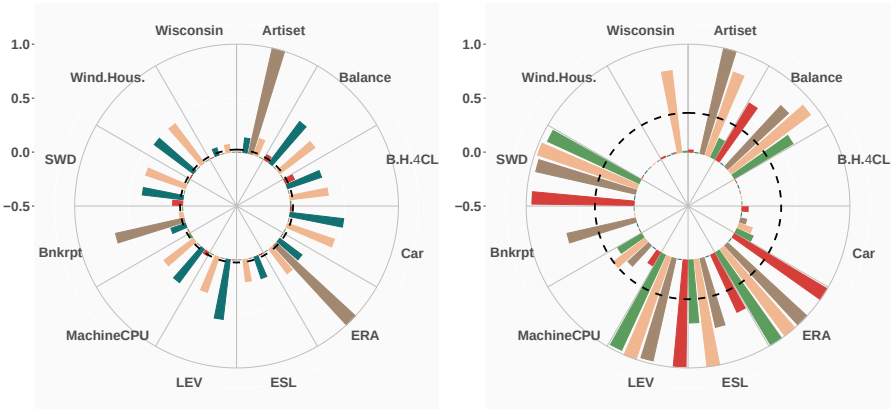
(a)

(b)

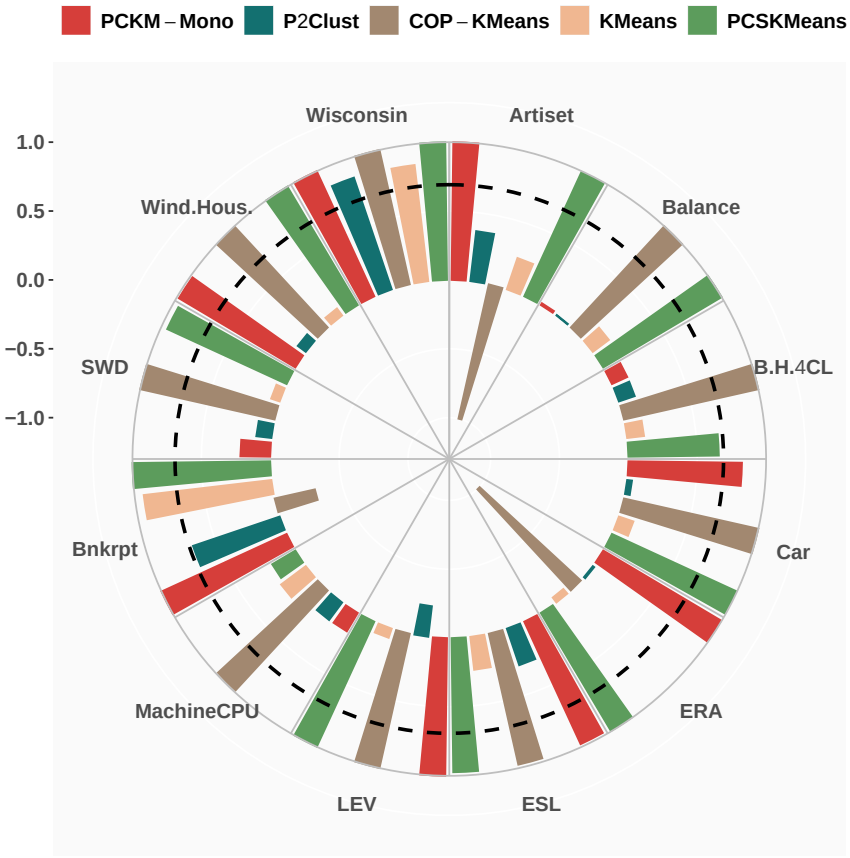


(c)

Figure 1: Comparative violinplots for the results obtained with CS_{10}. Figures 1a, 1b, and 1c show the results obtained for the Unsat, NMI, and ARI measures, respectively.

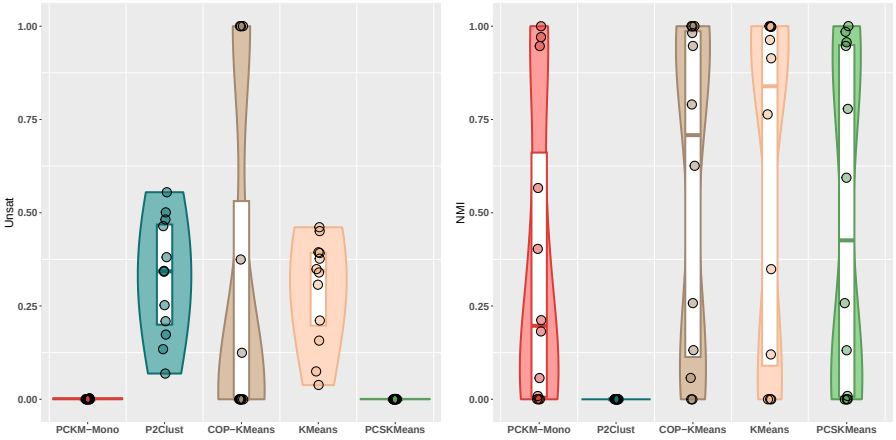


(a) (b)



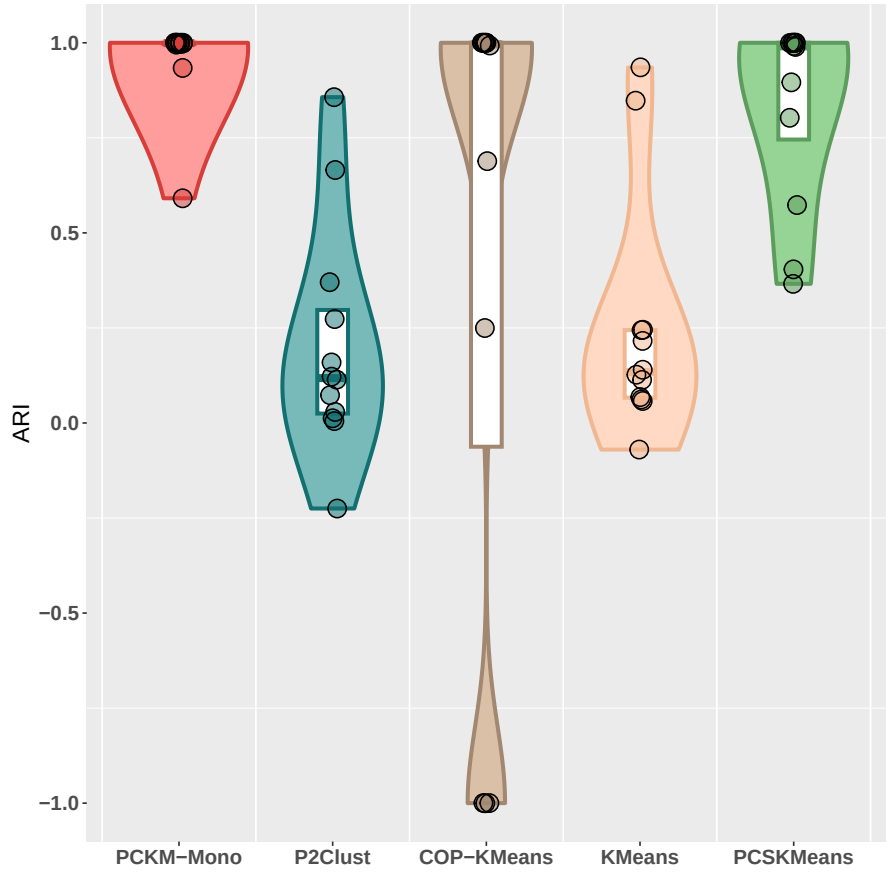
(c)

Figure 2: Comparative radarplots for the results obtained with CS_{10} in every dataset. Figures 2a, 2b, and 2c show the results obtained for the Unsat, NMI, and ARI measures, respectively. (Color legend in Figure 2c is valid for Figures 2a and 2b.)



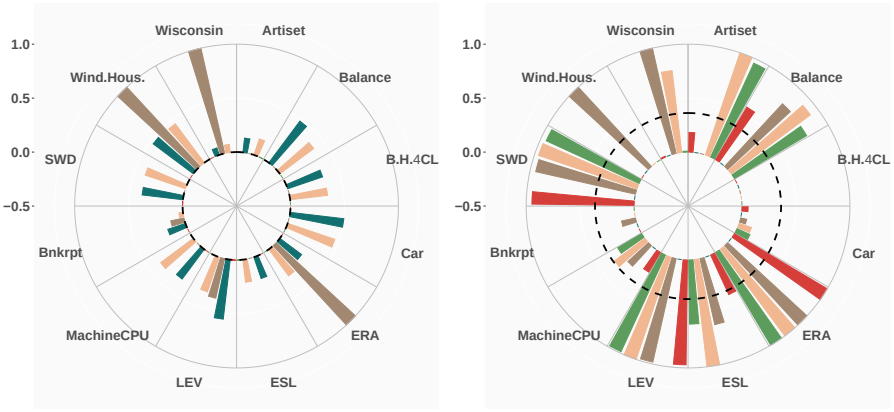
(a)

(b)

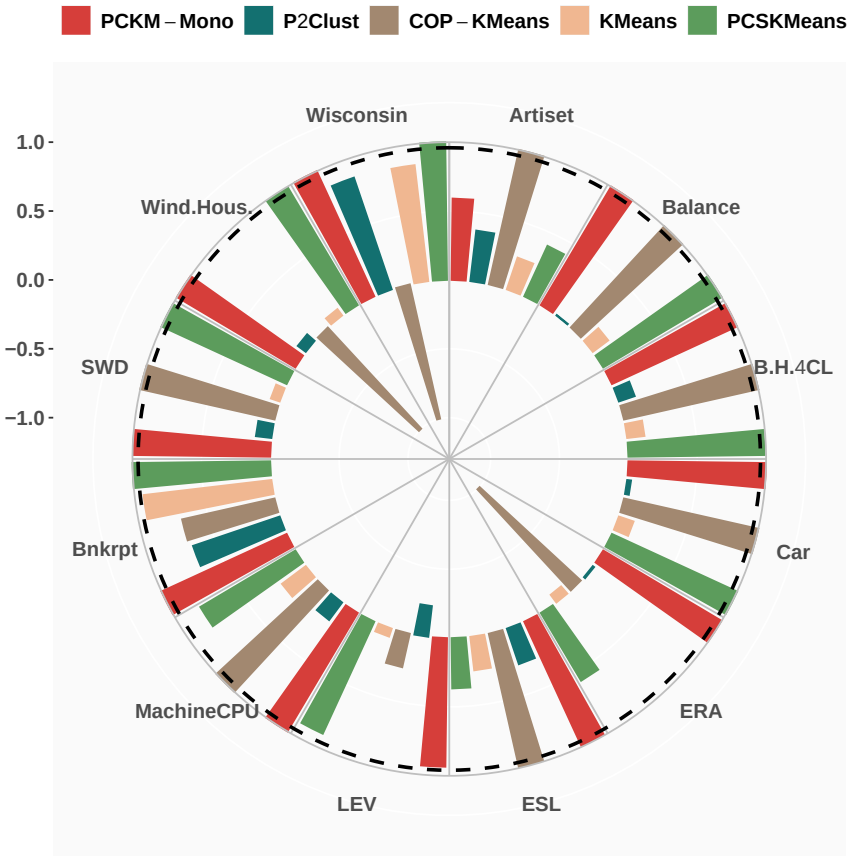


(c)

Figure 3: Comparative violinplots for the results obtained with CS_{15}. Figures 3a, 3b, and 3c show the results obtained for the Unsat, NMI, and ARI measures, respectively.

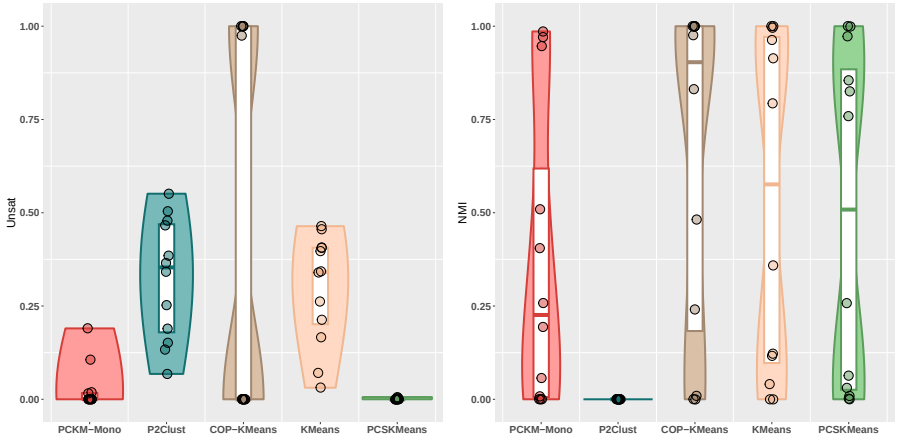


(a) (b)



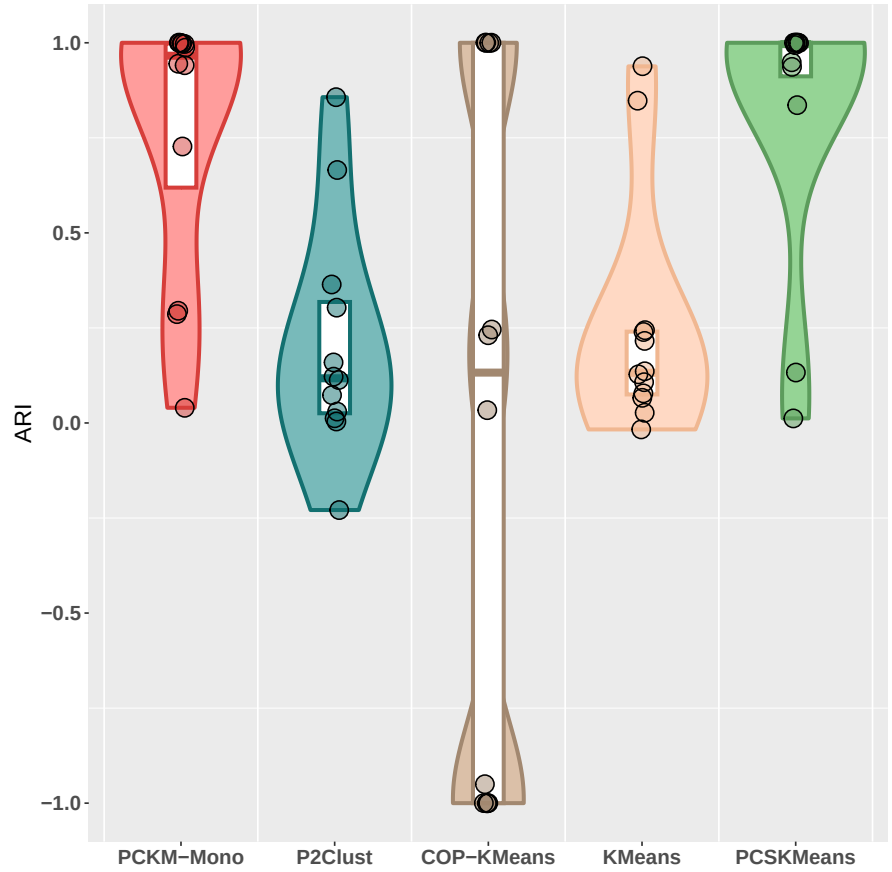
(c)

Figure 4: Comparative radarplots for the results obtained with CS_{15} in every dataset. Figures 4a, 4b, and 4c show the results obtained for the Unsat, NMI, and ARI measures, respectively. (Color legend in Figure 4c is valid for Figures 4a and 4b.)



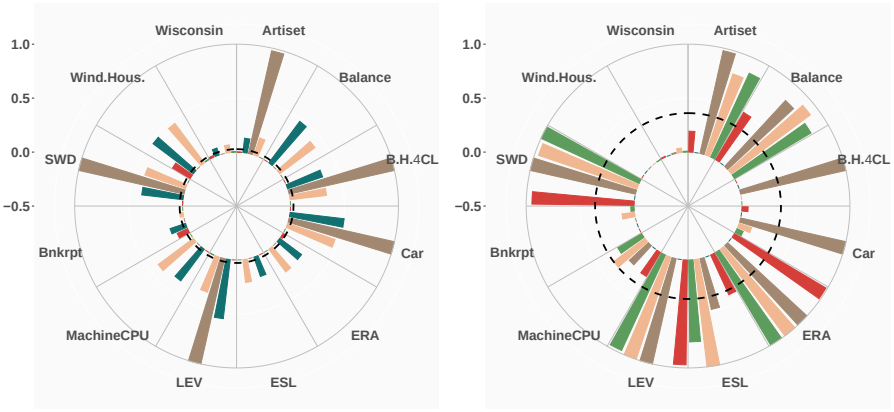
(a)

(b)

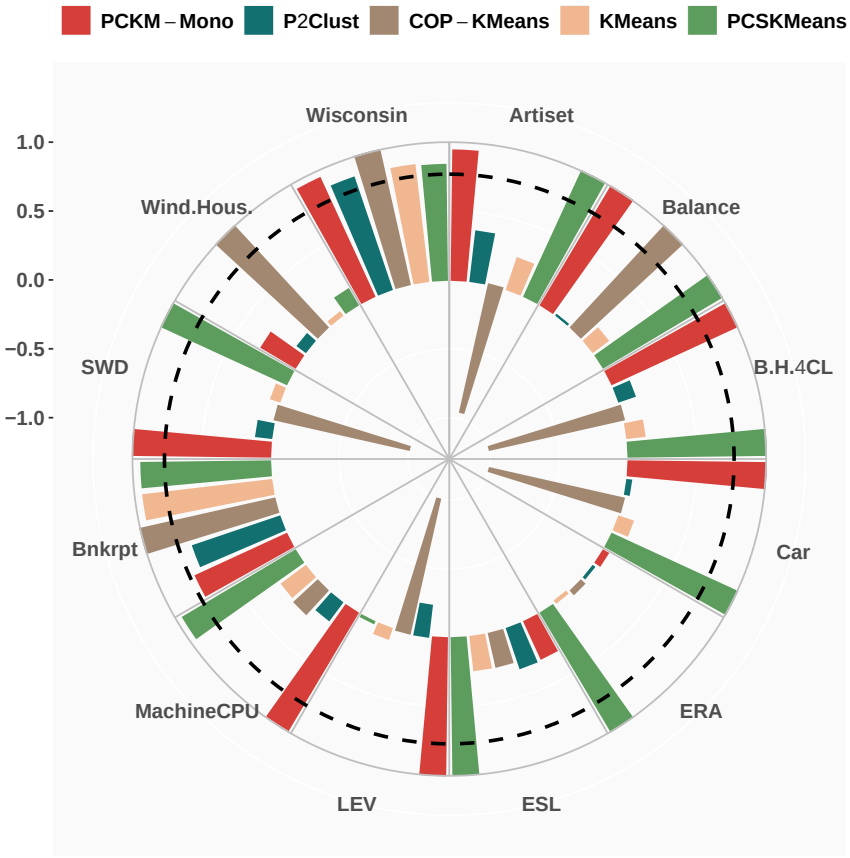


(c)

Figure 5: Comparative violinplots for the results obtained with CS_{20}. Figures 5a, 5b, and 5c show the results obtained for the Unsat, NMI, and ARI measures, respectively.



(a) (b)



(c)

Figure 6: Comparative radarplots for the results obtained with CS_{20} in every dataset. Figures 6a, 6b, and 6c show the results obtained for the Unsat, NMI, and ARI measures, respectively. (Color legend in Figure 6c is valid for Figures 6a and 6b.)

6 Statistical Analysis of Results

In contrast with NHST, it is possible to create illustrative graphical representations of the results of the Bayesian sign test. To do so, the obtained distribution is sampled to obtain a set of triplets, which are interpreted as barycentric coordinates in an equilateral triangle, thus producing a cloud of points with varying density. This is known as a heatmap. Figure 7 shows heatmaps which compare the proposed method PCKM-Mono with the rest of the benchmarked methods for the three measures obtained: ARI, NMI and Unsat. The region of practical equivalence is set to $rope = [-0.02, 0.02]$ for ARI, and to $rope = [-0.01, 0.01]$ for NMI and Unsat, following the guidelines in [41]. The results produced by PCKM-Mono are always taken as B in 8, and A represents the set of results obtained by the compared method. Please note that, as ARI is a measure to maximize, a cloud of points located in the region of the map corresponding to MPCK-Means would indicate statistically significant differences between the two methods in favor of MPCK-Means. The opposite situation is found for NMI and Unsat.

All heatmaps reinforce the conclusions obtained in the Experimental Results Section 5. It is clear that PCKM-Mono represents a statistically significant improvement over all compared method with respect to ARI, except for the comparison against PCSKMeans, which is the most debated one with a slight advantage for PCSKMeans. Heatmap 7d gives the general advantage to PCSKMeans for the ARI measure, but not by a wide margin, indicating no significant differences in some cases and advantage of PCKM-Mono in a significant portion of the experiments. When it comes to the comparison concerning NMI, and Unsat, conclusions remain unchanged. Heatmap 7e confirms the indisputable superiority of purely monotonic algorithms with respect to NMI. However, 7l reveals no statistically significant differences between PCKM-Mono and PCSKMeans with respect to Unsat, and 7j an advantage of PCKM-Mono over COP-Kmeans for the same measure. With this in mind, it is reasonable to assert that, for the experiments conducted in this study, the proposed PCKM-Mono algorithm has the same or better capabilities than previous CC algorithms to include constraints into the clustering process. Please note that, even if PCKM-Mono and PCSKMeans feature disputed results for the ARI measure, PCKM-Mono is indisputably superior to PCSKMeans for NMI and statistically similar to PCSKMeans regarding the Unsat measure, thus it is fair to claim an advantage of PCKM-Mono over PCSKMeans in the general case.

7 A case of study: The Shanghai Ranking dataset

In this section we assess the applicability of our proposal for a real-world problem. The Shanghai Ranking of World Universities (SRWU) dataset has been used before to test the capabilities of monotonic clustering methods, e.g. in [18] the top 100 institutions are used to test the P2Clust method. In our experiments we used the dataset available in this kaggle repository [2], which contains the SWRU results for years 2005 to 2015. Only the results from the year 2015 are used in our experiments. In our dataset, institutions are ranked from best

[2] https://www.kaggle.com/code/saurav9786/eda-for-university-ranking/data?select=shanghaiData.csv

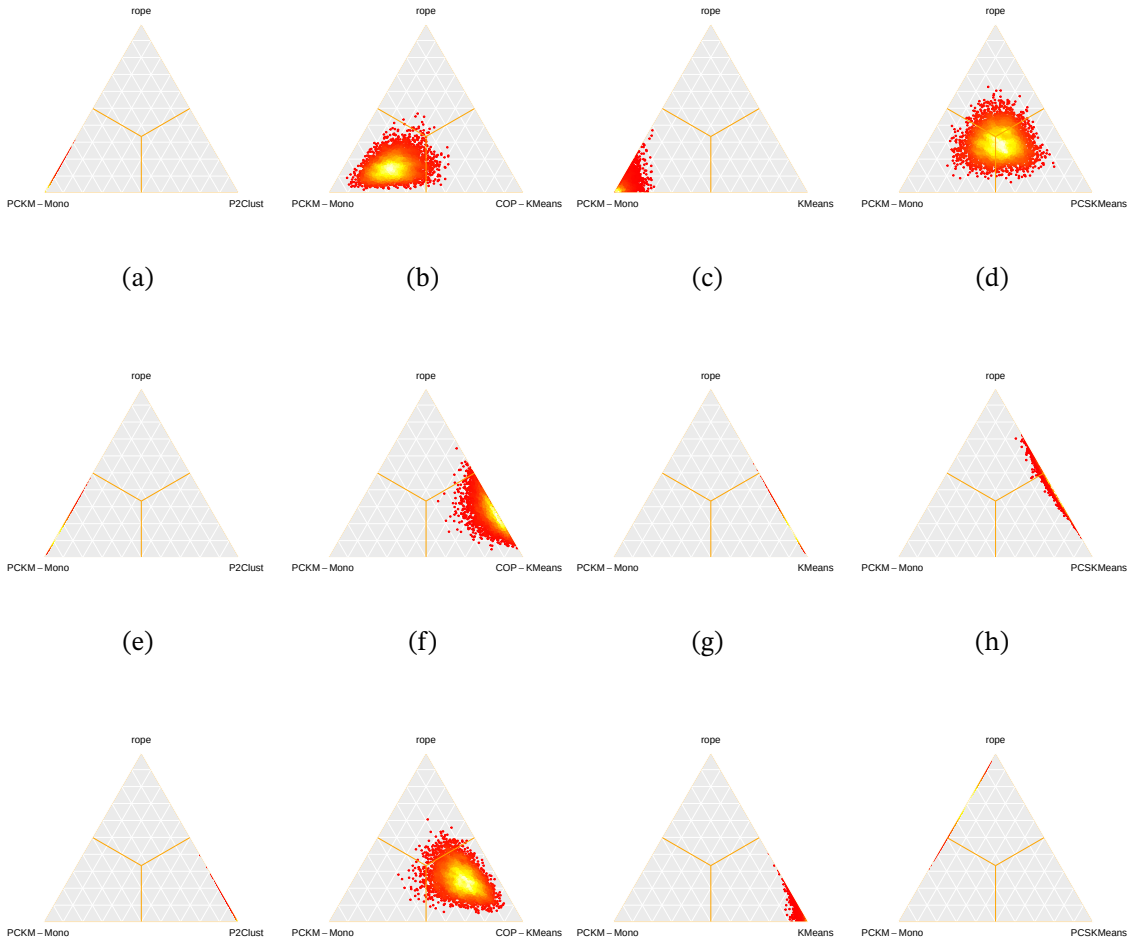


Figure 7: Heatmaps 7a to 7l compare the proposed method PCKM-Mono to the other four compared methods: P2Clust, COP-Kmeans, Kmeans and PCSKMeans, respectively, from left to right in every row. The first row compares the results for the ARI measure, the second row does so for NMI and the third is for Unsat.

to worst in chunks of size 50 for the first 100 institutions and in chunks of size 100 for the rest of them, generating a total of 7 classes for the 500 institution in the dataset. Our goal is to cluster the dataset so that institutions ranked in the same chunk appear in the same cluster in the final partition.

Originally, the dataset has 9 features, although some of them do not provide any valuable information for clustering methods and thus they can be removed, such as the institution name or its national rank. The dataset has 6 features after removing the useless ones, and can be visualized in the pairplot in Figure 8. Observing Figure 8, it seems clear that the

SRWU is in fact a monotonic dataset, an therefore, it is has to be addressed with monotonic methods. However, there are some exceptions to this monotonicity. In fact, if we compute the NMI value for the true partition of the dataset, we obtain a value of 0.07 as a result, which indicates that the monotonicity is broken by some instances. This is the reason why constraints can help improve the results, as if the dataset was purely monotonic, a method like P2Clust, which is hard constrained for monotonicity constraints, could solve it more accurately.

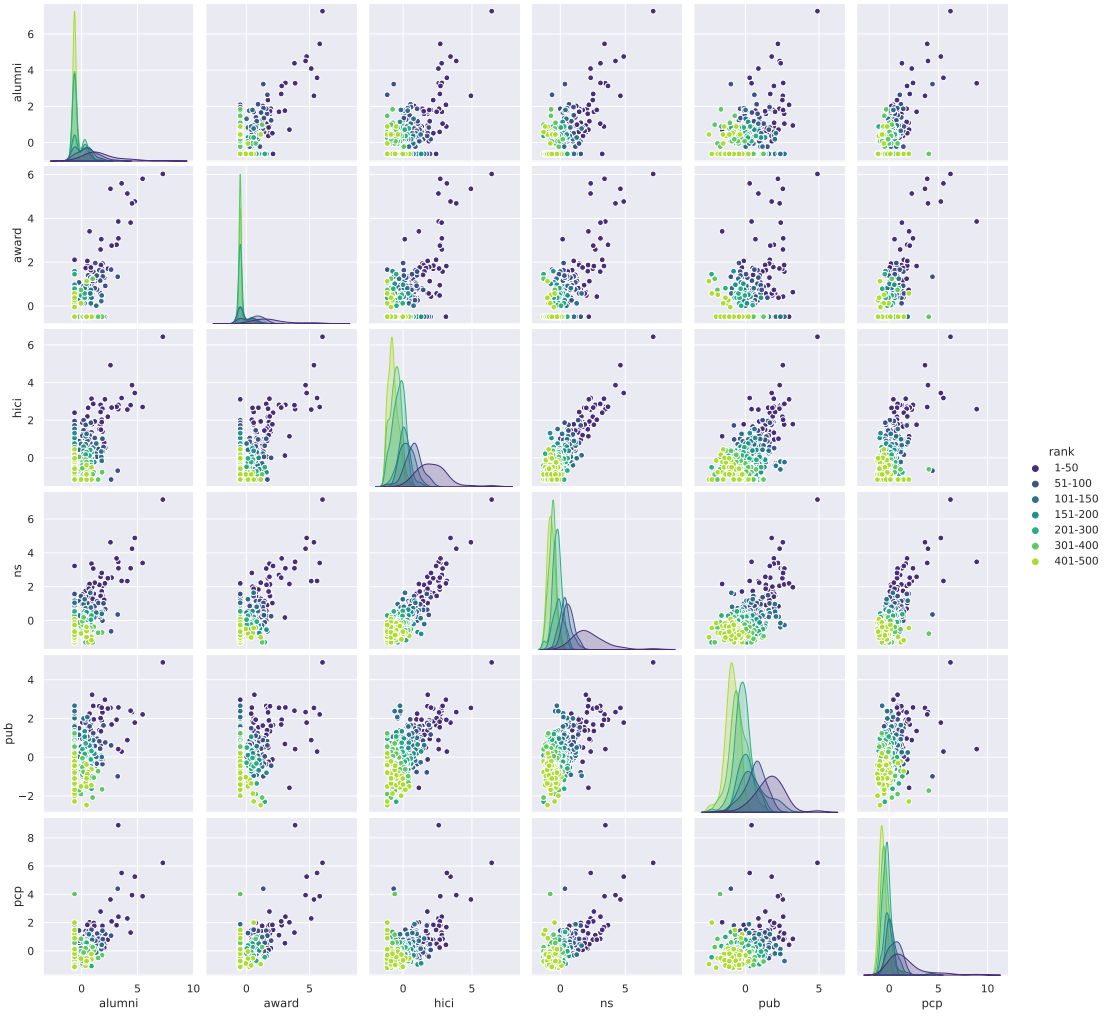
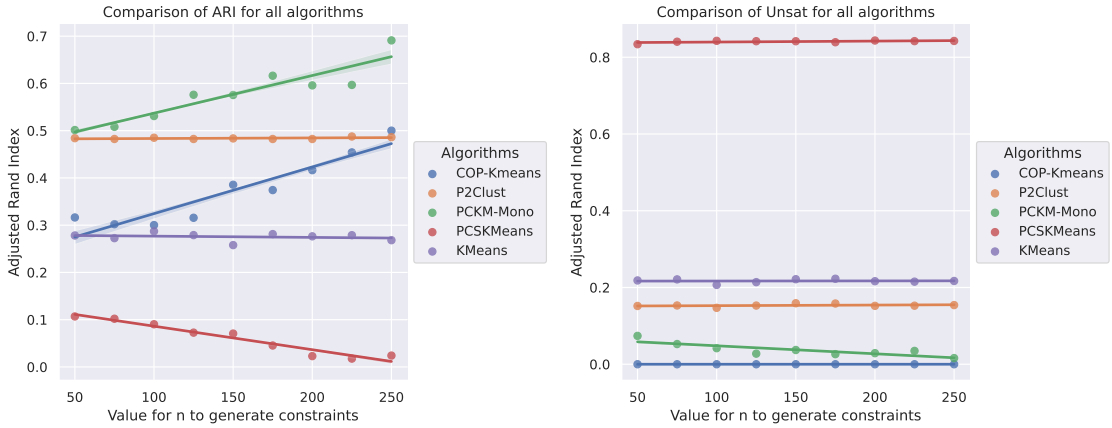


Figure 8: Pairplot for the Shanghai Ranking of world universities dataset

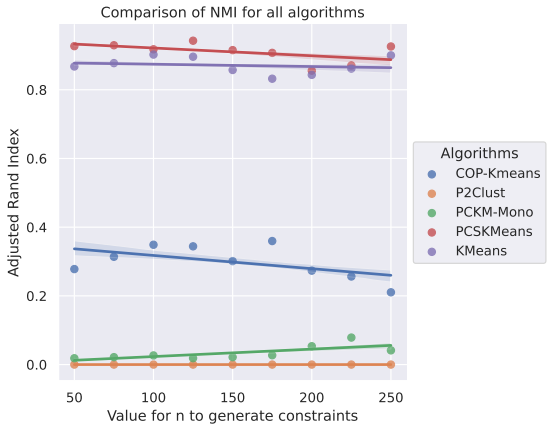
Scaling, standardization and missing values imputation (basic Knn imputer) steps are performed before applying all 5 clustering methods considered in this study to the dataset. Figure 9 shows the results obtained by all method for the three quality measures and with increasing values for n in the formula $(n_f(n_f - 1))/2$, and thus, generating increasing levels of constraint-based information. This is conducted to observe how the results scale with

the number of available constraints. Constraints are generated as it is done for benchmark datasets (see Section 4).



(a)

(b)



(c)

Figure 9: Scatterplot 9a compares ARI results obtained by all five method compared in this study. Figures 9b and 9c do so for the Unsat and NMI measures respectively.

In Figure 9a it can be clearly observed that PCKM-Mono represents the best option to generate scaling quality results for the SRWU partitioning problem. It is followed by the purely monotonic P2Clust method, which maintains stable results, as it does not consider constraints. It is also interesting to note how COP-Kmeans scale the results even by a greater factor than PCKM-Mono, although achieving worse results, as it cannot deal with the monotonicity of the data.

Regarding the results for Unsat, presented in Figure 9b, we can observe how Unsat values produced by PCKM-Mono scale inversely proportional with respect to the number of constraints. This is indicative of constraints helping the clustering process to find the true shape

of the cluster, therefore making it easier for the method to satisfy a higher number of them. The rest of the methods maintain a stable Unsat, with COP-Kmeans always producing a value of 0 for this measure (as it can never generate partitions which violate any constraint) and with PCSKMeans featuring the worse value for it. This is indicative of the method not being suitable at all for the problem, as even non-constrained non-monotonic methods such as Kmeans are able to obtain better Unsat results.

In Figure 9c we can observe one of the most interesting effects of constraints. Please note that NMI results for PCKM-Mono decline as the number of constraints increases. The interpretation of this result can be counterintuitive, as one could expect it to decrease. However, the NMI is actually shifting towards the NMI value produced by the true labels of SRWU $(0,07)$, thus being more accurate in practice. With regard to non-constrained methods, they maintain an stable NMI value (as expected), with P2Clust always producing an NMI of 0, as it can never generate partitions which violate monotonicity. For the non-monotonic constrained clustering methods, it can be observed that the influence of constraints in COP-Kmeans is enough to divert clusters from the hyperspherical shape produced by the Euclidean distance, and thus generating an acceptable NMI value, which is not the case for PCSKMeans.

All of these results are in favor of the hypothesis of pairwise constraints and monotonicity constraints benefiting from each other when combined. Please note that, PCKM-Mono would produce the same NMI values as P2Cust if it were not for pairwise constraints, which have proved to divert the method from this trend and towards more accurate NMI results.

8 Conclusion

In this study, the first method which addresses Monotonic Constrained Clustering (MCC) is proposed: Pairwise Constrained K-Means - Monotonic (PCKM-Mono). An expectation-minimization scheme is used to locally optimize a hybrid objective function, integrating a monotonic distance metric and a pairwise constraint-based penalty term. The experimental results obtained from a variety of datasets and their following statistical analysis confirm the viability of the proposed method when compared with purely monotonic and purely pairwise constrained clustering techniques. Even if PCKM-Mono obtains results similar to those obtained by previous approaches for specific monotonicity and pairwise constraint satisfaction, there is strong statistical evidence in favor of PCKM-Mono regarding general clustering quality measures.

Acknowledgements

Our work has been supported by the research projects PID2020-119478GB-I00, A-TIC-434-UGR20 and PREDOC_01648.

References

[1] Absalom E Ezugwu, Abiodun M Ikotun, Olaide O Oyelade, Laith Abualigah, Jeffery O Agushaka, Christopher I Eke, and Andronicus A Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022.

[2] Xiaosha Cai, Dong Huang, Guang-Yu Zhang, and Chang-Dong Wang. Seeking commonness and inconsistencies: A jointly smoothed approach to multi-view subspace clustering. *Information Fusion*, 91:364–375, 2023.

[3] Jonatan Enes, Roberto R Expósito, José Fuentes, Javier López Cacheiro, and Juan Touriño. A pipeline architecture for feature-based unsupervised clustering using multivariate time series from hpc jobs. *Information Fusion*, 93:1–20, 2023.

[4] Mohamed Abd Elaziz, Mohammed AA Al-Qaness, Esraa Osama Abo Zaid, Songfeng Lu, Rehab Ali Ibrahim, and Ahmed A. Ewees. Automatic clustering method to segment covid-19 ct images. *PLoS One*, 16(1):e0244416, 2021.

[5] Li Guo, Pengfei Shi, Long Chen, Chenglizhao Chen, and Weiping Ding. Pixel and region level information fusion in membership regularized fuzzy clustering for image segmentation. *Information Fusion*, 92:479–497, 2023.

[6] H Vani, MA Anusuya, and ML Chayadevi. Fuzzy clustering algorithms-comparative studies for noisy speech signals. *Ictact J. Soft Comput.*, 9(3):1920–1926, 2019.

[7] Jun Wang, Chang Tang, Zhenglai Li, Xinwang Liu, Wei Zhang, En Zhu, and Lizhe Wang. Hyperspectral band selection via region-aware latent features fusion based clustering. *Information Fusion*, 79:162–173, 2022.

[8] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[9] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[10] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, MSR-TR-2000-65, Microsoft Research, 2000.

[11] Ian Davidson and Sugato Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from data*, 1:1–41, 2007.

[12] Jana Schmidt, Elisabeth Maria Brandle, and Stefan Kramer. Clustering with attribute-level constraints. In *2011 IEEE 11th International Conference on Data Mining*, pages 1206–1211. IEEE, 2011.

[13] Baptiste Lafabregue, Jonathan Weber, Pierre Gançarski, and Germain Forestier. Deep constrained clustering applied to satellite image time series. In *ECML/PKDD Workshop on Machine Learning for Earth Observation Data (MACLEAN)*, Würzburg, Germany, 2019.

[14] Chao-Lung Yang and Thi Phuong Quyen Nguyen. Constrained clustering method for class-based storage location assignment in warehouse. *Industrial Management & Data Systems*, 116(4):667–689, 2016.

[15] Son T Mai, Sihem Amer-Yahia, Sébastien Bailly, Jean-Louis Pépin, Ahlame Douzal Chouakria, Ky T Nguyen, and Anh-Duong Nguyen. Evolutionary active constrained clustering for obstructive sleep apnea analysis. *Data Science and Engineering*, 3(4):359–378, 2018.

[16] MA Balafar, R Hazratgholizadeh, and MRF Derakhshi. Active learning for constrained document clustering with uncertainty region. *Complexity*, 2020, 2020.

[17] Jian Gao, Xiaoxia Tao, and Shaowei Cai. Towards more efficient local search algorithms for constrained clustering. *Information Sciences*, 621:287–307, 2023.

[18] Jean Rosenfeld, Yves De Smet, Olivier Debeir, and Christine Decaestecker. Assessing partially ordered clustering in a multicriteria comparative context. *Pattern Recognition*, 114:107850, 2021.

[19] Sergio González, Salvador García, Sheng-Tun Li, Robert John, and Francisco Herrera. Fuzzy k-nearest neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise. *Neurocomputing*, 439:106–121, 2021.

[20] José-Ramón Cano, Pedro Antonio Gutiérrez, Bartosz Krawczyk, Michał Woźniak, and Salvador García. Monotonic classification: An overview on algorithms, performance measures and data sets. *Neurocomputing*, 341:168–182, 2019.

[21] Weiwei Pan. Fraudulent firm classification using monotonic classification techniques. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 9, pages 1773–1776. IEEE, 2020.

[22] Alexander Chistyakov, Ekaterina Lobacheva, Alexander Shevelev, and Alexey Romanenko. Monotonic models for real-time dynamic malware detection. *arXiv preprint arXiv:1804.03643*, 2018.

[23] Jose-Ramon Cano, Naif R Aljohani, Rabeeh Ayaz Abbasi, Jalal S Alowidbi, and Salvador Garcia. Prototype selection to improve monotonic nearest neighbor. *Engineering Applications of Artificial Intelligence*, 60:128–135, 2017.

[24] David Leslie. Understanding artificial intelligence ethics and safety. *arXiv preprint arXiv:1906.05684*, 2019.

[25] Germán González-Almagro, Pablo Sánchez Bermejo, Juan Luis Suarez, José-Ramón Cano, and Salvador García. Monotonic constrained clustering: A first approach. In Hamido Fujita, Philippe Fournier-Viger, Moonis Ali, and Yinglin Wang, editors, *Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence*, pages 725–736, Cham, 2022. Springer International Publishing.

[26] Jean Rosenfeld, Dimitri Van Assche, and Yves De Smet. Lexicographic constrained multicriteria ordered clustering. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 453–464. Springer, 2021.

[27] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[28] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 577–584. Morgan Kaufmann Publishers Inc., 2001.

[29] Martin HC Law, Alexander Topchy, and Anil K Jain. Clustering with soft and group constraints. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 662–670. Springer, 2004.

[30] Wojciech Kotlowski and Roman Slowinski. On nonparametric ordinal classification with monotonicity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2576–2589, 2012.

[31] Bernard Roy. *Multicriteria methodology for decision aiding*, volume 12. Springer Science & Business Media, 1996.

[32] Avgoustinos Vouros and Eleni Vasilaki. A semi-supervised sparse k-means algorithm. *Pattern Recognition Letters*, 142:65–71, 2021.

[33] Isaac Triguero, Sergio González, Jose M Moyano, Salvador García, Jesús Alcalá-Fdez, Julián Luengo, Alberto Fernández, Maria José del Jesús, Luciano Sánchez, and Francisco Herrera. KEEL 3.0: an open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems*, 10(1):1238–1249, 2017.

[34] Hong Zhu, Xizhao Wang, and Ran Wang. Fuzzy monotonic k-nearest neighbor versus monotonic fuzzy k-nearest neighbor. *IEEE Transactions on Fuzzy Systems*, 2021.

[35] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[36] Sergio González, Francisco Herrera, and Salvador García. Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Generation Computing*, 33(4):367–388, 2015.

[37] Germán González-Almagro, Julián Luengo, José-Ramón Cano, and Salvador García. Dils: constrained clustering through dual iterative local search. *Computers & Operations Research*, page 104979, 2020.

[38] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[39] Jacinto Carrasco, Salvador García, María del Mar Rueda, and Francisco Herrera. rnpbst: An r package covering non-parametric and bayesian statistical tests. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 281–292. Springer, 2017.

[40] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[41] Jacinto Carrasco, Salvador García, MM Rueda, S Das, and Francisco Herrera. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, 54:100665, 2020.

Bibliography

[AK20] Altan A. and Karasu S. (2020) Recognition of covid-19 disease from x-ray images by hybrid model consisting of 2d curvelet transform, chaotic salp swarm algorithm and deep learning technique. *Chaos, Solitons & Fractals* 140: 110071.

[AKB19] Altan A., Karasu S., and Bekiros S. (2019) Digital currency forecasting with chaotic meta-heuristic bio-inspired signal processing techniques. *Chaos, Solitons & Fractals* 126: 325–336.

[Alt20] Altan A. (2020) Performance of metaheuristic optimization algorithms based on swarm intelligence in attitude and altitude control of unmanned aerial vehicle for path following. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 1–6. IEEE.

[AT+99] Alba E., Troya J. M., *et al.* (1999) A survey of parallel distributed genetic algorithms. *Complexity* 4(4): 31–52.

[Bai13] Bair E. (2013) Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics* 5(5): 349–361.

[BB06] Bae E. and Bailey J. (2006) Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In *Sixth International Conference on Data Mining (ICDM'06)*, pp. 53–62. IEEE.

[BDW08] Basu S., Davidson I., and Wagstaff K. (2008) *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press.

[BN06] Bishop C. M. and Nasrabadi N. M. (2006) *Pattern recognition and machine learning*, volumen 4. Springer.

[CAA+17] Cano J.-R., Aljohani N. R., Abbasi R. A., Alowidbi J. S., and Garcia S. (2017) Prototype selection to improve monotonic nearest neighbor. *Engineering Applications of Artificial Intelligence* 60: 128–135.

[CGK+19] Cano J.-R., Gutiérrez P. A., Krawczyk B., Woźniak M., and García S. (2019) Monotonic classification: An overview on algorithms, performance measures and data sets. *Neurocomputing* 341: 168–182.

[CLSR18] Chistyakov A., Lobacheva E., Shevelev A., and Romanenko A. (2018) Monotonic models for real-time dynamic malware detection. *arXiv preprint arXiv:1804.03643* .

[CLVV+14] Coello C. A. C., Lamont G. B., Van Veldhuizen D. A., *et al.* (2014) *Evolutionary algorithms for solving multi-objective problems*, volumen 5. Springer.

[CSP+07] Cios K. J., Swiniarski R. W., Pedrycz W., Kurgan L. A., Cios K. J., Swiniarski R. W., Pedrycz W., and Kurgan L. A. (2007) Unsupervised learning: association rules. *Data Mining: A Knowledge Discovery Approach* pp. 289–306.

[CSZ10] Chapelle O., Schlkopf B., and Zien A. (2010) *Semi-Supervised Learning*. The MIT Press, 1st edition.

[DB07] Davidson I. and Basu S. (2007) A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from data* 1: 1–41.

[DH+06] Duda R. O., Hart P. E., *et al.* (2006) *Pattern classification*. John Wiley & Sons.

[DR05a] Davidson I. and Ravi S. (2005) Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 59–70. Springer.

[DR05b] Davidson I. and Ravi S. (2005) Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 International Conference on Data Mining*, pp. 138–149. SIAM.

[DR09] Davidson I. and Ravi S. S. (2009) Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. *Data mining and knowledge discovery* 18(2): 257–282.

[DS98] Draper N. R. and Smith H. (1998) *Applied regression analysis*, volumen 326. John Wiley & Sons.

[DSOM+19] Del Ser J., Osaba E., Molina D., Yang X.-S., Salcedo-Sanz S., Camacho D., Das S., Suganthan P. N., Coello C. A. C., and Herrera F. (2019) Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation* 48: 220–250.

[DWB06] Davidson I., Wagstaff K. L., and Basu S. (2006) Measuring constraint-set utility for partitional clustering algorithms. In *European conference on principles of data mining and knowledge discovery*, pp. 115–126. Springer.

[ESA+20] Ezugwu A. E., Shukla A. K., Agbaje M. B., Oyelade O. N., José-García A., and Agushaka J. O. (2020) Automatic clustering algorithms: a systematic review and bibliometric analysis of relevant literature. *Neural Computing and Applications* pp. 1–60.

[GGL+21] González S., García S., Li S.-T., John R., and Herrera F. (2021) Fuzzy k-nearest neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise. *Neurocomputing* 439: 106–121.

[GGLGH19] García-Gil D., Luengo J., García S., and Herrera F. (2019) Enabling smart data: noise filtering in big data classification. *Information Sciences* 479: 135–152.

[GLH15] García S., Luengo J., and Herrera F. (2015) *Data preprocessing in data mining*. Springer.

[GP10] Gendreau M. and Potvin J.-Y. (2010) *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition.

[HA85] Hubert L. and Arabie P. (1985) Comparing partitions. *Journal of classification* 2(1): 193–218.

[HKP12] Han J., Kamber M., and Pei J. (2012) Data mining concepts and techniques third edition. *University of Illinois at Urbana-Champaign Micheline Kamber Jian Pei Simon Fraser University* .

[HLZC19] He L., Li W., Zhang Y., and Cao Y. (2019) A discrete multi-objective fireworks algorithm for flowshop scheduling with sequence-dependent setup times. *Swarm and Evolutionary Computation* 51: 100575.

[JGGF16] José-García A. and Gómez-Flores W. (2016) Automatic clustering using nature-inspired metaheuristics: A survey. *Applied Soft Computing* 41: 192–213.

[JMF99] Jain A. K., Murty M. N., and Flynn P. J. (1999) Data clustering: a review. *ACM computing surveys (CSUR)* 31(3): 264–323.

[KABA20] Karasu S., Altan A., Bekiros S., and Ahmad W. (2020) A new forecasting model with wrapper-based feature selection approach using multi-objective optimization technique for chaotic crude oil time series. *Energy* 212: 118750.

[KKM02] Klein D., Kamvar S. D., and Manning C. D. (2002) From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. Technical report, Stanford.

[KS12] Kotlowski W. and Slowinski R. (2012) On nonparametric ordinal classification with monotonicity constraints. *IEEE Transactions on Knowledge and Data Engineering* 25(11): 2576–2589.

[KWH22] Kuncheva L., Williams F., and Hennessey S. (2022) A bibliographic view on constrained clustering. *arXiv preprint arXiv:2209.11125* .

[Les19] Leslie D. (2019) Understanding artificial intelligence ethics and safety. *arXiv preprint arXiv:1906.05684* .

[M+67] MacQueen J. *et al.* (1967) Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volumen 1, pp. 281–297. Oakland, CA, USA.

[Mie12] Miettinen K. (2012) *Nonlinear multiobjective optimization*, volumen 12. Springer Science & Business Media.

[Mir12] Mirkin B. (2012) *Clustering: a data recovery approach*. CRC Press.

[NP14] Nanda S. J. and Panda G. (2014) A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary computation* 16: 1–18.

[Pan20] Pan W. (2020) Fraudulent firm classification using monotonic classification techniques. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volumen 9, pp. 1773–1776. IEEE.

[PF91] Piateski G. and Frawley W. (1991) *Knowledge discovery in databases*. MIT press.

[Ran71] Rand W. M. (1971) Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66(336): 846–850.

[RDSDD21] Rosenfeld J., De Smet Y., Debeir O., and Decaestecker C. (2021) Assessing partially ordered clustering in a multicriteria comparative context. *Pattern Recognition* 114: 107850.

[Roy96] Roy B. (1996) *Multicriteria methodology for decision aiding*, volumen 12. Springer Science & Business Media.

[SGH21] Suárez J. L., García S., and Herrera F. (2021) A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing,* 425: 300–322.

[SSZL05] Sheng W., Swift S., Zhang L., and Liu X. (2005) A weighted sum validity function for clustering with a hybrid niching genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 35(6): 1156–1167.

[ST14] Subramanya A. and Talukdar P. P. (2014) Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8(4): 1–125.

[TGH15] Triguero I., García S., and Herrera F. (2015) Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems* 42(2): 245–284.

[VEH20] Van Engelen J. E. and Hoos H. H. (2020) A survey on semi-supervised learning. *Machine Learning* 109(2): 373–440.

[WFH+05] Witten I. H., Frank E., Hall M. A., Pal C. J., and DATA M. (2005) Practical machine learning tools and techniques. In *Data Mining*, volumen 2.

[XJRN03] Xing E. P., Jordan M. I., Russell S. J., and Ng A. Y. (2003) Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pp. 521–528.

[ZDX19] Zhong G., Deng X., and Xu S. (2019) Active informative pairwise constraint formulation algorithm for constraint-based clustering. *IEEE Access* 7: 81983–81993.

[ZG09] Zhu X. and Goldberg A. B. (2009) Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning* 3(1): 1–130.

[Zho21] Zhou Z.-H. (2021) Semi-supervised learning. In *Machine Learning*, pp. 315–341. Springer.

[Zhu05] Zhu X. J. (2005) Semi-supervised learning literature survey. *Synthesis lectures on artificial intelligence and machine learning* .

[ZL11] Zheng L. and Li T. (2011) Semi-supervised hierarchical clustering. In *2011 IEEE 11th International Conference on Data Mining*, pp. 982–991. IEEE.