UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Nuevos Retos en Minería de Reglas de Asociación: un Enfoque Basado en Programación Genética

Memoria de Tesis presentada por

José María Luna Ariza

como requisito para optar al grado de
Doctor en Informática

Directores

Sebastián Ventura Soto

José Raúl Romero Salguero

Cristóbal Romero Morales

Granada — Enero de 2014

Editor: Editorial de la Universidad de Granada
Autor: José María Luna Ariza
D.L.: GR 1057-2014
ISBN: 978-84-9028-958-7

UNIVERSITY OF GRANADA

Department of Computer Science
and Artificial Intelligence

New Challenges in Association Rule Mining: an Approach Based on Genetic Programming

A Thesis presented by

José María Luna Ariza

as a requirement to aim for the degree of
Ph.D. in Computer Science

Advisors

Sebastián Ventura Soto

José Raúl Romero Salguero

Cristóbal Romero Morales

Granada 2014, January

La memoria titulada "*Nuevos Retos en Minería de Reglas de Asociación: un Enfoque Basado en Programación Genética*", que presenta José María Luna Ariza para optar al grado de Doctor en Informática, ha sido realizada dentro del Programa Oficial de Doctorado en Tecnologías de la Información y la Comunicación del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada, bajo la dirección de los doctores Sebastián Ventura Soto, José Raúl Romero Salguero y Cristóbal Romero Morales cumpliendo, en su opinión, los requisitos exigidos a este tipo de trabajos y respetando los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, Enero de 2014

El Doctorando

Fdo: José María Luna Ariza

El Director

Fdo: Dr. Sebastián Ventura Soto

El Director

Fdo: Dr. José Raúl Romero Salguero

El Director

Fdo: Dr. Cristóbal Romero Morales

Mención de Doctorado Internacional

Esta tesis cumple los criterios establecidos por la Universidad de Granada para la obtención del Título de Doctor con Mención Internacional:

1. Estancia predoctoral mínima de 3 meses fuera de España en una institución de enseñanza superior o centro de investigación de prestigio, cursando estudios o realizando trabajos de investigación relacionados con la tesis doctoral:

   Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, the Netherlands.
   Responsable de la estancia: **Ph.D. Paul de Bra**, Professor and Chair.

2. La tesis cuenta con el informe previo de dos doctores expertos y con experiencia investigadora acreditada pertenecientes a alguna institución de educación superior o instituto de investigación distinto de España:

   - **Ph.D. Alessandro Provetti**, Professor, Department of Mathematics & Computer Science, Università degli Studi di Messina, Messina, Italy.
   - **Ph.D. Krzysztof Cios**, Associate Professor, Department of Computer Science, Virginia Commonwealth University, Virginia, United States.

3. Entre los miembros del tribunal evaluador de la tesis se encuentra un doctor procedente de una institución de educación superior distinto de España y diferente del responsable de la estancia predoctoral:

   Ph.D. Mykola Pechenizkiy, Assistant Professor, Department of Computer Science, Eindhoven University of Technology, Eindhoven, the Netherlands.

4. Parte de la tesis doctoral se ha redactado y presentado en dos idiomas: castellano e inglés.

Granada, Enero de 2014

El Doctorando

Fdo: José María Luna Ariza

Tesis Doctoral parcialmente subvencionada por la Comisión Interministerial de Ciencia y Tecnología (CICYT) con el proyecto **TIN2011-22408** y por la Junta de Andalucía con fondos FEDER con el proyecto de excelencia **P08-TIC-3720**.

Asímismo ha sido subvencionada por el programa predoctoral de Formación del Profesorado Universitario (FPU) del Ministerio de Educación (referencia **AP-2010-0041**), convocatoria publicada en el B.O.E. N^0 20 de 24 de enero de 2011 y resuelta en el B.O.E. N^0 305 de 20 de diciembre de 2011.

A person who never made a mistake never tried anything new.

Albert Einstein

Acknowledgements

"*Success is a journey, not a destination. The doing is often more important than the outcome*". Along the development of this Doctoral Thesis, I have learnt many important things about me that were hidden. Thus, it is hard to express my entire gratitude to all the people that deserve it, so I really doubt that I could ever be able to fully convey my appreciation.

I would like to express my gratitude to my mentor, Dr. Sebastian Ventura, responsible for pointing me out the best way to take along my research life. Dr. Ventura is more a father and a friend than a professor. I owe him my eternal gratitude. A very special thanks goes out to another of my supervisors, Dr. José Raúl Romero, for the assistance he has provided me during my research. I really appreciate his advices and support that have led me to reach this point. I also would like to thank Dr. Cristóbal Romero, who joined us a bit later. Dr. Romero is that kind of person that makes you feel better regardless of the situation.

I also would thank to anyone in the KDIS research group, specially to *the lab guys*. We have shared many funny moments, and not that funny situations as well. We have laughed and cried together. I will never forget them: Juan, Juanlu, Alberto and Aurora; thank you so much.

I also have to acknowledge Mykola, Yulia, Lahm, Vicinius and Pedro, my Spanish colleague. All of you made me feel really comfortable during my short stay in Eindhoven. You are responsible for leaving me good memories about Eindhoven, which I consider now as my second city.

All my thanks go out to my friends, which fortunately could be hardly enumerated, and my relatives for their suggestions. Last but not least, I want to acknowledge Maite, my future wife; thank you for being my fellow traveler.

Finally, I would like to dedicate this PhD Thesis to my father, my mother and my sister, without their love and encouragement, I would not have finished it.

"*Lost time is never found again*". `Benjamin Franklin`

Resumen

En la actualidad, existe un gran interés en el almacenamiento de datos de cualquier índole, tanto a nivel personal como empresarial. En ocasiones, estos datos son almacenados sin reportar ninguna información de interés, por lo que muchos de estos datos son almacenados pero nunca utilizados. La extracción de reglas de asociación es una tarea de la minería de datos que permite describir comportamientos y relaciones entre grandes conjuntos de datos. Es en esta tarea en la que se centra la presente Tesis Doctoral, dando solución a muchos de los interrogantes abiertos en el campo y que han sido previamente estudiado y analizados.

En primer lugar, se ha llevado a cabo un estudio detallado del estado del arte en la campo de la minería de asociación, describiendo su origen, la evolución de las propuestas, así como los temas abiertos. Las primeras propuestas se basaron en una búsqueda exhaustiva del espacio de soluciones, por lo que el dominio de búsqueda debía ser finito. Numerosas investigaciones se han llevado a cabo en este campo, proponiendo algoritmos que permitan buscar soluciones en dominios numéricos donde las soluciones no son finitas. Sin embargo, no sólo la búsqueda en dominios numéricos es un tema de estudio en minería de asociación, sino que el uso de asociaciones negativas, o incluso raras o poco frecuentes, son temas abiertos en este área.

La presente Tesis Doctoral incluye una serie de propuestas para la extracción de reglas de asociación mediante una metodología basada en programación genética gramatical. El objetivo es proponer nuevos algoritmos de asociación que permitan la extracción de reglas en un único paso y de un modo eficiente. El uso de gramáticas para la codificación de las soluciones permite que el conocimiento adquirido sea muy flexible, y que las reglas descubiertas puedan incluir condiciones tanto positivas como negativas, o incluso numéricas.

En una primera propuesta, se presenta un algoritmo de programación genética gramatical para la extracción de reglas de asociación frecuentes, el cual es conocido como G3PARM. Este algoritmo ha sido específicamente diseñado para la tarea de asociación, incluyendo dos operadores genéticos basados en la frecuencia de aparición de las condiciones. El análisis de este algoritmo demuestra su enorme eficiencia, extrayendo reglas en un único paso y mediante un proceso evolutivo. G3PARM ha sido comparado con algoritmos clásicos o deterministas, así como con propuestas posteriores basadas en metodologías evolutivas. Dicha comparación ha sido llevada a cabo mediante un análisis estadístico, cuyos resultados corroboran la importancia de G3PARM en el campo de la minería de asociación.

Debido a los buenos resultados de G3PARM, se propone un posterior modelo para la extracción de reglas asociación poco frecuentes. Este tipo de asociaciones no han sido muy estudiadas, aunque son de gran relevancia, describiendo comportamientos anómalos pero no por ello poco interesantes. Este tipo de comportamientos son realmente interesantes en medicina o en educación, donde relaciones poco frecuentes puede describir el comportamiento de personas con dificultades en su proceso de aprendizaje.

En minería de asociación, existen multitud de medidas que determinan la calidad de las reglas obtenidas. Muchas de estas son contrapuestas, por lo que la optimización de una determinada medida implica la ausencia de otra medida. Existen muchos problemas donde es necesario llegar a un compromiso entre las medidas, por lo que el uso de metodologías multi-objectivo son de gran interés. La presente Tesis Doctoral describe dos algoritmos multi-objetivo que son capaces de optimizar dos medidas de asociación a la vez. Estos modelos están basados en dos modelos de gran relevancia en la optimización multi-objectivo, como son NSGA-2 y SPEA-2.

Por último, una de las principales dificultades en el uso de algoritmos evolutivos es la necesidad de optimizar los parámetros que se requieren para su ejecución, lo que lleva a que la mayoría de sus usuarios son expertos en la materia. En este sentido, se proponen dos algoritmos evolutivos que auto-ajustan sus parámetros en función de las características de los datos. Estas propuestas resultan de gran interés para usuarios no expertos en el campo, permitiendo que cualquier tipo usuario pueda hacer uso de estos algoritmos y de las ventajas que conllevan.

Las propuestas presentadas en esta Tesis Doctoral han sido llevadas a cabo en un campo de aplicación real. Uno de los campos de mayor interés en el descubrimiento de relaciones de interés entre patrones es el ámbito de la educación. El objetivo es descubrir relaciones que permitan describir comportamientos (tanto frecuentes como anómalos) que ayuden al profesor o tutor en su proceso de enseÃ±anza. Estas relaciones descubiertas permite describir y ayudar a alumnos con determinados problemas en el aprendizaje y mejorar sus resultados.

Todos los algoritmos propuestos en la presente Tesis Doctoral han sido estadísticamente evaluados mediante una serie de tests no paramétricos. Los algoritmos han sido comparados con otros algoritmos de reconocido prestigio dentro del campo de la minería de asociación, demostrando la enorme eficiencia de los modelos propuestos.

En un capítulo final se presentan una serie de nuevas líneas de investigación que han surgido con la elaboración de la presente Tesis Doctoral. Todas estas nuevas líneas están en mayor o menor medida relacionadas con la minería de reglas de

asociación. Ha surgido la necesidad de extraer reglas de asociación en entornos relacionales. El creciente interés en la recolección de datos ha dado lugar a su almacenamiento en entornos relacionales, por lo que los datos se encuentran separados en diferentes tablas o incluso diferentes localizaciones físicas. En segundo lugar, ha surgido la necesidad de extraer reglas de asociación que sean sensibles al contexto, por lo que un mismo dato puede tener connotaciones diferentes en base a su contexto. También se ha propuesto, como línea de investigación nueva, la extracción de relaciones excepcionales entre los datos. Una relación es excepcional si el comportamiento que representa es completamente opuesto al comportamiento esperado. Por último, se propone la aplicación de reglas de asociación al campo del descubrimiento de subgrupos, es decir, describir grupos de elementos cuyas características están estrechamente relacionadas.

Abstract

This Doctoral Thesis involves a series of approaches for mining association rules by means of a grammar-guided genetic programming based methodology. The ultimate goal is to provide new algorithms that mine association rules in only one step and in a highly efficient way. The use of grammars enables the flexibility of the extracted knowledge to be increased. Grammars also enable obtaining association rules that comprise categorical, quantitative, positive and negative attributes to be mined.

Firstly, as for the mining of frequent association rules, a novel grammar-based algorithm, called G3PARM, has been proposed. It is able to discover rules having positive, negative, categorical and quantitative attributes. The evolutionary model is able to perform the mining process in one single step. This PhD Thesis also includes a model for mining rare or infrequent association rules, as well as two multi-objective approaches that optimize two different quality measures at time. Additionally, two novel algorithms that self-adapt their parameters are considered. In this sense, a previous tuning of the parameters would not be required, as they are adjusted depending on the data under study. Finally, the developed methodologies have been applied to the educational field to discover interesting information that could be used to improve the courses.

All the algorithms proposed in this Doctoral Thesis have been evaluated in a proper experimental framework, using different types of datasets and comparing their performance against other published methods of proved quality. Results have been verified by applying non-parametric statistical tests, demonstrating the many benefits of using a grammar-based methodology to address the association rule mining problem.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AR Association Rule

ARM Association Rule Mining

ARMGA Association Rule Mining Genetic Algorithm

ARMMGA Association Rule Mining Multi-objective Genetic Algorithm

CAR Class Association Rules

CD Critical Difference

CFG Context-Free Grammar

DM Data Mining

EA Evolutionary Algorithm

EMO evolutionary multi-objective optimization

GA Genetic Algorithm

GAR Genetic Association Rules

GBAP-ARM Grammar-Based Ant Programming for Association Rule Mining

GENAR Genetic Association Rules

GGGP Grammar Guided Genetic Programming

G3P Grammar Guided Genetic Programming

G3PARM Grammar Guided Genetic Programming Association Rule Mining

G3P-MI Grammar-Guided Genetic Programming algorithm for Multiple Instance learning

GP Genetic Programming

IRAR Imperfectly Rare Association Rule

KDD Knowledge Discovery in Databases

MODENAR Multi-Objective Differential Evolution algorithm for mining Numeric Association Rules

MOGBAP-ARM Multi-Objective Grammar-Based Ant Programming for Association Rule Mining

NAR Negative Association Rule

NSGA Non dominated Sort Genetic Algorithm

POF Pareto Optimal Front

PRAR Perfectly Rare Association Rule

QAR Quantitative Association Rule

QARGA Quantitative Association Rules Genetic Algorithm

RAR Rare Association Rule

SPEA Strength Pareto Evolutionary Algorithm

UCI University of California Irvine

1

Introduction

Generally, data stored in different areas lacks of interest, and an in-depth analysis and study is required to extract knowledge that was hidden in the raw data. Currently, many research studies have been focused on the extraction of useful knowledge, giving rise to the field known as knowledge discovery in databases (KDD).

KDD is concerned with the development of methods and techniques for making sense of data [1], and this process comprises a set of steps like data selection, data cleaning and preprocessing, data transformation, data mining (DM), and interpretation of the extracted knowledge. DM is one of the most important steps in the KDD process, being responsible for mining non-trivial information hidden in data. *W. J. Frawley et al.* [2] described the DM task as follows:

> *DM is a non-trivial extraction of implicit, previously unknown, and potentially useful information from data. Given a set of facts, a pattern is defined as a statement in a specific language that describes, in a simple way, relationships among the existing facts with a certainty. A pattern that is interesting and certain enough (according to the users' criteria) is called knowledge.*

DM refers to a multidisciplinary learning task where many ideas from varied disciplines converge, and this task could be classified into two main learning methods: unsupervised [3] and supervised [4]. The former includes approaches for descriptive mining tasks, which characterize the data properties. The latter comprises models for predictive mining tasks that categorize the space in a determined number of regions with respect to a target variable, and perform inference on the current data in order to make predictions.

Association rule mining (ARM), an important technique in the DM field, has received enormous attention since its introduction by *Agrawal et al.* [5] in the early 90s. This descriptive mining task searches for strong relationships among patterns, also known as items, that are usually hidden in data. An association rule (AR) is defined as an implication of the form $Antecedent \rightarrow Consequent$, both $Antecedent$ and $Consequent$ being disjoint item-sets. If the antecedent of an AR is satisfied, then it means that it is highly probable that the consequent will be also satisfied.

The ARM task was originally designed for market basket analysis to obtain relations between products, like $diapers \rightarrow beer$, which describes the high probability of someone acquiring diapers also buying beer. It would allow shop-keepers to exploit this specific relationship by moving their products closer together on the shelves.

The initial proposals for the extraction of ARs follow an exhaustive search methodology based on the algorithm proposed by *R. Agrawal and R. Srikant* in 1994 [6]. This algorithm, known as Apriori, divides the ARM problem into two sub-problems: (1) obtaining all the existing frequent item-sets in the data and (2) extracting all the ARs from the item-sets obtained in the previous step. For a better understanding of the computational and memory requirements, let us consider a dataset containing k items. In such a dataset, $2^k - 1$ item-sets and $3^k - 2^{k+1} + 1$ rules could be generated, so the use of huge datasets implies a prohibitively hard process of mining. An improved version of Apriori, called FP-Growth, was proposed later by *J. Han* [7]. This approach stores the frequent-patterns into a tree structure, which is used as a compressed database. Nevertheless, these approaches suffer from the data size and the number of extracted rules. Note that, despite the fact that many authors have considered the ARM problem from an exhaustive search point of view, both the memory requirements and the computational time required still are

an important handicap. It must also be noted that exhaustive search algorithms could be hardly applied on raw data, which usually consist of numerical values. Therefore, the search space is often so large that this sort of algorithms are unable to be properly executed.

For the sake of addressing the optimization problem, evolutionary computation has been widely used in DM tasks, and specifically, in ARM in order to address this optimization problem. Actually, most of the existing evolutionary approaches are based on genetic algorithms (GA) [8, 9], having a fixed-length linear genotype, which lacks of flexibility. For instance, GAs do not usually move the location of the gene within the genotype, so the representation is quite restrictive in this sense. Besides, changes in the desired rules imply different genotype representations and substantial changes in the whole algorithm.

The use of evolutionary algorithms (EA) [10] have opened the way for mining not only ARs that comprise positive and categorical conditions but also quantitative and negative conditions. ARs that comprise this sort of attributes have been named by some authors as quantitative [11] and negative association rules [12], respectively. EAs are highly recommended under many situations, especially when they are used by qualified data miners. Nevertheless, the chances of ARs to be mined by high-performance algorithms without the need to specify a large number of parameters is a common requirement for non-expert users.

Additionally, only a few authors have explored the issue of mining reliable ARs that do not frequently occur in data, namely rare association rules (RAR) [13], in order to differentiate them from ARs that appear more frequently. RARs are of great interest in many different areas and algorithms in this context require a higher computational time, being the number of rules to be mined extremely large.

Finally, it should be noted that not only the high-performance and the type of rules to be mined is of interest in the ARM field, but also the right choice of the quality evaluation measures [14] is a major concern. Most of the approaches were based on the extraction of frequent and reliable rules, but it does not imply at all that the rules discovered are interesting. That brings up the idea of proposing new quality measures [15] that determine the real interest of the knowledge discovered to the data miner , and many of these metrics are based on the correlation between the antecedent and consequent of the rule.

1.1 Objectives

The ultimate aim of this PhD Thesis is to provide a new paradigm to deal with ARM, facing up to existing problems in the field, some of which have been addressed by researchers in different ways. Firstly, we conduct a detailed bibliographic review of ARM, including the existing algorithms and discussing about their major drawbacks. This study will help the reader to better understand the real problems and difficulties found in the field, and providing a general overview of the solution proposed to overcome them.

One of the main issues in ARM, specially in those algorithms based on an exhaustive search methodology, are the computational and memory requirements. The computational complexity is a major handicap in the mining of ARs, since the search space could be so large that it might be hardly maintainable. This problem, together with the memory requirements, makes exhaustive search approaches to become useless mainly due to the growing interest in storage where data are continuously increasing in the number of both records and attributes. Thereby, it is presented a series of evolutionary approaches for mining ARs that make use of a context-free grammar (CFG) that characterizes the type of rules to be mined. It also enables the search space to be reduced as the flexibility of the extracted knowledge increases.

We propose different approaches to deal with different types of attributes. Raw data are used without requiring any previous preprocessing step to transform them into a specific representation (as determined by other algorithms). For instance, numerical attributes are not required to be discretized as exhaustive search algorithms do. Additionally, the proposed approaches are able to deal with both negative attributes and RARs.

The task of looking for rules of interest should not be reduced to a search for frequent and reliable rules, since these two metrics do not always imply that the rule is of interest. Thus, we include not only metrics for determining the reliability and frequency of occurrence of the rules, but also to measure their interest. For this purpose, we also propose multi-objective models that optimize two measures at time.

In addition, the use of EAs means optimizing a huge number of parameters. That becomes uninteresting for non-expert users, who would require some prior knowledge about evolutionary computation. The idea consists in proposing models that provide the possibility of mining ARs in a simple and highly efficient way. Therefore, the proposed evolutionary models should automatically adjust their own parameters to the specific data under study.

Finally, we apply the proposed models to the educational field. Thus, we propose a set of models to overcome the existing drawbacks, and we also apply these models to discover interesting knowledge that is of interest in a specific field.

To sum up, the objectives to be reached in this PhD Thesis are the followings:

- To analyse the state of the art in ARM, including the most representative algorithms in the area, and their major advantages and drawbacks.
- To provide solutions that overcome the computational and memory requirements. The goal is to develop new algorithms that mine ARs in only one step, achieving a highly efficient solution.
- To use a CFG that increases the flexibility of the solutions. The grammar enables ARs that comprises positive, quantitative and negative conditions to be mined.
- To deal with ARs that do not frequently appear in data, known as RARs, which can be of high interest in many different fields.
- To address the ARM problem under a multi-objective perspective, optimizing different quality measures at time.
- To propose new evolutionary models that do not require as many parameters as existing EAs do, self-adapting their parameters to the data under study.
- To apply the aforementioned concept and models to the educational field.

1.2 Organization and Document Structure

The rest of this document is organized as follows. Chapter 2 provides the formal conceptual framework of ARs, as well as a description of the ARM task. Furthermore, an interesting analysis of different quality measures is carried out. In such an analysis, the relation between the most representative metrics in the ARM field is depicted. Additionally, the historical background of ARM is presented, considering both deterministic and stochastic methodologies.

Chapter 3 describes the use of G3P in ARM. In this chapter, a first approach for mining ARs using G3P is proposed, named G3PARM. Here, a proper description and experimental study is carried out, pointing out the excellent performance of G3PARM with regard to existing ARM algorithms.

Additionally, a detailed description about the extraction of RARs by using G3P is given in Chapter 4. In this chapter, four fitness functions to avoid noisy rules are proposed, giving rise to an interesting evolutionary algorithm for mining ARs that do not frequently occur in data.

Chapters 5 and 6 present two multi-objective models and two parameter-free algorithms for mining ARs, respectively. In some situations, it is of interest to optimize not only one measure but more than one at time. Thus, Chapter 5 proposes two multi-objective algorithms for mining ARs by means of G3P. These algorithms are based on two well-known models in the multi-objective field, which provide promising results when optimizing two quality AR measure at time. Additionally, in Chapter 6, two novel self-adaptive algorithms for mining ARs are presented.

Chapter 7 presents the application of the proposed approaches to the educational field, which allow the instructors to improve their courses. Finally, in Chapter 8, some concluding remarks and future work drawn from our experiments are discussed.

2

Background

In this chapter, the formal conceptual framework used in this PhD Thesis is properly described. It also includes the description of the most interesting quality measures to evaluate the extracted ARs, and the relationships among them. Additionally, a number of cases where the use of ARM is highly applicable are presented. Then, the historical background of ARM is explored in depth, from those approaches based on exhaustive search methodologies to those proposals that use evolutionary computation to solve the exhaustive search problems. Also, we describe the genetic programming methodology and how it could serve to solve many of the existing problems in the ARM field. Finally, we present the use of multi-objective methodologies for mining ARs, achieving a trade-off between more than one quality measure at time.

2.1 Association Rule Mining

2.1.1 Knowledge Representation and Extraction

ARM is considered an important technique in the DM field, and it has received enormous attention since its introduction by *Agrawal et al.* [5] in the early 90s.

ARM is considered as a descriptive mining task that seeks for frequent, interesting and strong relationships among patterns that are usually hidden in data. In its definition, an AR is an implication of the form $Antecedent \rightarrow Consequent$, both $Antecedent$ and $Consequent$ being sets with no items in common. An AR determines that it is highly probable to satisfy the consequent of the rule once the antecedent was previously satisfied. Formally speaking, the AR concept could be defined as follows:

Let $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ be a set of items, and let A and C be item-sets, that is, $A = \{i_1, ..., i_j\} \subset \mathcal{I}$ and $C = \{i_1, ..., i_k\} \subset \mathcal{I}$. An association rule is an implication of the type $A \rightarrow C$ where $A \subset \mathcal{I}$, $C \subset \mathcal{I}$, and $A \cap C = \emptyset$.

ARM was originally designed for market basket analysis, describing high correlations between products, so it allows shop-keepers to exploit the discovered relationships in order to increase the sales. Thus, ARM was considered as an important task to describe the customers' behavior.

Quantitative association rules. Real-world data usually comprise quantitative attributes (e.g. age, size, etc). Therefore, the need for mining this kind of attributes gave rise to a new concept of ARs, known as quantitative association rules (QARs) [11, 16]. A major issue of this task is that numerical attributes are usually defined on a wide range of values, and it is useless to work on all possible values as done for categorical values. The mining of QARs could not be considered as an extension of discovering nominal ARs, so the goal is to dynamically divide their domains into intervals during the mining process so as to satisfy some quality criteria [17]. Let $\mathcal{I} = \{i_i, i_2, ..., i_n\}$ be the set of items, and $\mathbb{R}$ be the set of real numbers, then $\mathcal{I}_{\mathbb{R}} = \{(x, l, r) | x \in \mathcal{I}, l \in \mathbb{R}, u \in \mathbb{R}, l \leq x \leq r\}$. A triple $(x, l, r) \in \mathcal{I}_{\mathbb{R}}$ denotes either a quantitative attribute x with a value interval $[l, r]$, or a categorical attribute with a value l $(l = r)$. A QAR is an AR, that is, an implication of the form $A \rightarrow C$, with the peculiarity that $A \subset \mathcal{I}_{\mathbb{R}}$, $C \subset \mathcal{I}_{\mathbb{R}}$, and $A \cap C = \emptyset$.

Negative association rules. Sometimes, it is of interest to discover items that are not positively associated, that is, the presence of a set of items implies that other set of items is highly unlikely to be present [18]. This sort of ARs is considered

as negative association rules (NARs), which are rules that appear in the form $\neg A \rightarrow C$, $A \rightarrow \neg C$ or $\neg A \rightarrow \neg C$. Thus, the entire antecedent or consequent appears as a conjunction of negated and non-negated attributes.

Class association rules. Additionally, notice that ARM does not constraint the target attribute. Instead, the process finds any relationship that appears in the data regardless of which items are used as conditions in the antecedent or consequent. However, in some application domains, the use of targets could be of great interest. For instance, suppose that a company requires the analysis of text documents where the extraction of relations between topics is a challenge. In such a situation, the goal is to find out the words that are associated with specific topics in order to be able to group topics by words. This sort of ARs is known as class association rules (CAR), and they are considered as subsets of ARs with the specified targets (also known as classes) as their consequents.

Rare association rules. Finally, the looking for rules that do not frequently appear in a dataset have given rise to the concept of RARs. The process of generating RARs generates two types of rules as described in [19]: perfectly rare association rule (PRAR) and imperfectly rare association rule (IRAR). Both kind of rules are considered as RARs since they are satisfied by less than a maximum number of times (support quality measure). A rule is considered as PRAR if the frequency of occurrence is greater than the frequency of occurrence of each item within the rule, that is, $\forall x : x \in (A \cup C), frequency(x) < maximum\_frequency$. On the other hand, a rule is considered as IRAR if the frequency of occurrence of the complete rule is less than a maximum predefined threshold, and there is at least one pattern in the rule that appears with a frequency greater than or equal to a maximum predefined threshold, that is, $\exists x : x \in (A \cup C), frequency(x) \geq maximum\_frequency$.

2.1.2 Interestingness Measures

In ARM, there are many interestingness measures that could be divided into two main categories: subjective and objective measures. The former takes into account both the data and the user who requires the data. Thus, the user's background about the data is usually considered in this type of measures. As for the objective measures, they are based on only the raw data. Therefore, this document only

deals with objective measures, which do not require any user's background that could be hardly quantified.

As previously stated, the cornerstone of ARM is the search for item-sets that are satisfied at least a certain minimum number of times. This frequency of occurrence is widely known as the support of the item, being one of the major quality measures used in this field. The support measure represents the probability of occurrence of an item i_n in a database D, i.e., it is formally defined as $support(i_n) = |i_n|/|D|$. Similarly, the support of a rule $A \rightarrow C$ is calculated as the support of the item-set formed by $A \cup C$, so $support(A \rightarrow C) \equiv support(A \cup C)$.

In 1991, *Piatetsky-Shapiro* [20] suggested that any accuracy measure $\mathcal{M}$ should verify three specific properties in order to separate strong and weak rules (in the sense of assigning them high and low values, respectively). The properties are the following:

- Property 1: $\mathcal{M}(A \rightarrow C) = 0$ when $support(A \rightarrow C) = support(A) \times support(C)$. This property claims that any accuracy measure $\mathcal{M}$ must test the independence (though values other than 0 could be used, depending on the range of the measure).
- Property 2: $\mathcal{M}(A \rightarrow C)$ monotonically increases with $support(A \rightarrow C)$ when other parameters remain the same.
- Property 3: $\mathcal{M}(A \rightarrow C)$ monotonically decreases with $support(A)$ or with $support(C)$ when other parameters remain the same.

As well as support, confidence is a quality measure that appears in any problem where ARM is applied. This second quality measure determines the reliability or strength of implication of the rule, so the higher its value, the more accurate the rule is. In a formal way, the confidence measure (see Equation 2.1) is defined based on *support* by calculating the proportion of transactions included in $A \cup C$ among those transactions that contain A.

$$confidence(A \rightarrow C) = \frac{support(A \cup C)}{support(A)} \tag{2.1}$$

Support and confidence are broadly conceived as the finest quality measures in ARM and, consequently, a great variety of proposals make use of them. These proposals attempt to discover rules whose support and confidence values are greater than certain thresholds. Nevertheless, many authors have considered that the mere fact of exceeding these quality thresholds does not guarantee that the rules are interesting at all [15]. For instance, the support-confidence framework does not provide a test for capturing the correlation of two item-sets.

Lift is defined in Equation 2.2 as the relation between the confidence of the rule and the expected confidence or support of the consequent. More specifically, if $lift(A \rightarrow C) = 1$, then $support(A \rightarrow C) = support(A) \times support(C)$ so A and C are independent (Property 1 of *Piatetsky-Shapiro*); if $lift(A \rightarrow C) > 1$, then C is positively dependent on A (Property 2 of *Piatetsky-Shapiro*); finally, if $lift(A \rightarrow C) < 1$, then C is negatively dependent on A (Property 3 of *Piatetsky-Shapiro*). Thus, most authors look for a positive correlation among A and C, in such a way that only values greater than 1 are desired, that is, the confidence of the rule is greater than the support of the consequent.

$$lift(A \rightarrow C) = \frac{support(A \rightarrow C)}{support(A) \times support(C)} = \frac{confidence(A \rightarrow C)}{support(C)} \tag{2.2}$$

Similarly to the lift or interest measure, the leverage measure (see Equation 2.3) proposed by *Piatetsky-Shapiro* (also called novelty in [21]) calculates how different is the co-occurrence of the antecedent and consequent from expected, that is, from independence. Leverage takes values in the range [-0.25, 0.25], and its value is zero in those cases where the antecedent and consequent are statistically independent, so values close to zero imply uninteresting rules.

$$leverage(A \rightarrow C) = support(A \rightarrow C) - (support(A) \times support(C)) \tag{2.3}$$

Many other quality measures [22] have been proposed to deal with some of the weaknesses of the support-confidence framework, and conviction (see Equation 2.4) is one of these measures. Conviction represents the degree of implication of a rule, and values far from the unity indicate interesting rules. This quality measure is infinite for logical implications, that is, for ARs with a maximum confidence

value. Similarly to the lift measure, a conviction value of 1 implies an independence between A and C.

$$conviction(A \rightarrow C) = \frac{1 - support(C)}{1 - confidence(A \rightarrow C)} \tag{2.4}$$

In ARM, the right choice of quality measures that properly deal with the data under study is a major issue. Many research studies are focused on the different metrics to determine the interest of the rules mined. Thereby, the knowledge about the relationship among the quality measures (see Figure 2.1) is of great interest to select the most appropriate for each situation.

From this study, it can be deduced that the confidence value is always greater or equal to the support value. Hardly an AR could be discovered with a confidence lower than its probability of occurrence. Furthermore, it could be stated that the higher the support of a rule, the more probable is that this rule is not of interest according to lift, leverage and conviction. Thus, maximum support values, that is, $support(A \rightarrow C) = 1$, implies maximum confidence values, and therefore, lift

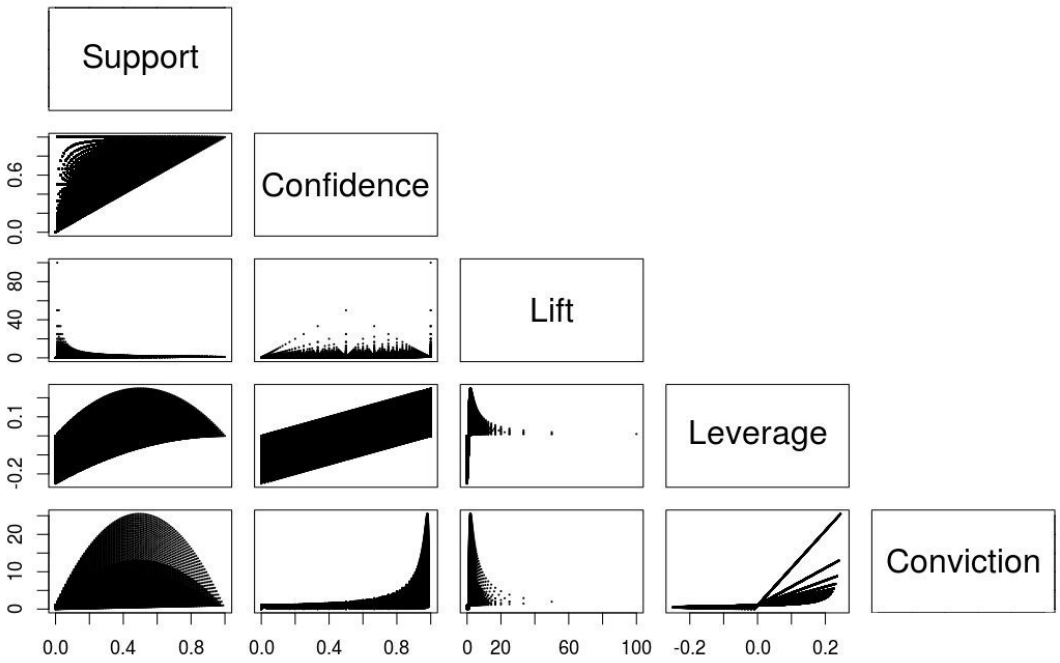


Figure 2.1: Relationship between different quality measures

and conviction values equal to the unity, and leverage values equal to zero. All these relationships, among others, are properly depicted in the chart. It is of high interest for the expert, providing a previous knowledge about the behavior of the quality measures, and how they relate in order to select the most appropriate set of measures.

Finally, once a set of ARs is discovered $(R_1, R_2, ...R_n)$, it is of interest to evaluate the quality of the set, calculating the percentage of instances covered by this set. This measure, known as coverage, evaluates the whole set of rules instead of a specific rule (see Equation 2.5).

$$coverage(R_1, R_2, ...R_n) = \frac{|\{R_1, R_2, ...R_n \subseteq T, T \in D\}|}{|D|} \tag{2.5}$$

2.1.3 Applications of ARM

Since ARM reveals relations among patterns, this task has been broadly used in the business field, where the discovery of strong and interesting relations helps in effective decision making. However, business is not the only area where this challenging DM task has been applied, including medical diagnosis, census data, web fraud detection, credit car business, etc.

In market basket analysis, databases comprise large number of transactions that list all items bought by a specific customer. In this situation, it is of interest the discovery of items that are related and usually purchased together. It allows shopkeepers to move the products closer together on the shelves in order to exploit this specific relationship, or even bring out promotions of these specific products.

Medical diagnosis is another interesting area of knowledge where ARM has been successfully applied. This application field is not as easy as it seems, since diagnosis involves unreliable tests and the presence of noise in many examples. This issue may result in unsatisfactory predictions that hamper the medical process. ARM has been also applied to identify the probability of illness in a certain disease by defining relations between symptoms. Additionally, the recognition of patients who suffer a particular disease that does not occur frequently [23] has been of interest. Here, the extraction of RARs plays an important role.

Census analysis is a very interesting application of ARM [24]. This analysis could bring out interesting relations among education, health, transportation, shopping in malls or street-markets, etc. The use of ARM in this kind of data can help in supporting good public policies.

There exist many other application domains for which the identification of rare patterns is an important issue. Communication failure detection [25], analysis of interesting rare patterns in telecommunication networks [26], or credit card fraud detection [27] are some of the applications where it is interesting to use ARs and, particularly, RARs.

The educational field [28] requires a special interest from the knowledge extraction point of view. Many applications have been proposed by discovering frequent relations in the students' behavior [29]. Additionally, the use of infrequent associations can also be of great interest, since they are related to rare but crucial cases. For instance, they might allow the instructor to verify a set of rules concerning certain abnormal learning problems that should be taken into account when dealing with students with special needs. Thus, this information could help the instructor to discover a minority of students who may need specific support with their learning process.

Association rules can reveal interesting information in web-based educational systems, discovering which contents students tend to access, or which combination of tools they usually use. Many authors have focused their studies on the application of ARM to online educational systems. In [30], *Ha et al.* performed an analysis of the existing associations in virtual knowledge structures, which were formed by learners in their navigation on the web pages. *Li et al.* [31] also proposed recommender agents for e-learning systems. They proposed the use of ARM to discover associations between user actions and visited URLs. The goal was to recommend online learning activities in a course web site based on the learner's access history.

ARM has also been used to confront the problem of continuous feedback in the educational process [32]. They proposed an approach to analyze the learners' behavior in learning management systems in order to offer a learning environment capable of increasing the learning effectiveness of the new mode of learning as well as the efficient organization of the institutional resources. Normally, ARM has been applied to many different situations in the educational field: to study learning data

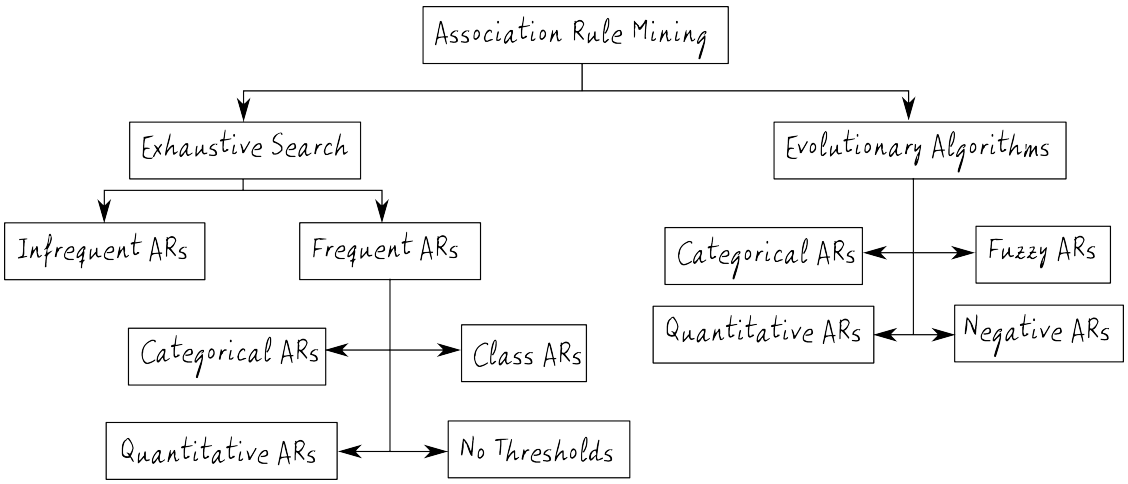


Figure 2.2: Categorization of existing proposals in ARM

in order to figure out the impact of the provided resources on the final marks [33]; to find hidden information that provides teachers a further analysis, refinement and reorganization of their teaching materials [34]; to reinforce the content of the e-learning web platform [35]; etc.

2.2 State-of-Art in Association Rule Mining

In this section, existing proposals for mining ARs are described in depth. These approaches are categorized as depicted in Figure 2.2. Proposals have been grouped into to main categories: exhaustive search and evolutionary algorithms. Both groups are described in the following two subsections. Finally, the use of grammars to represents rules and the mining of ARs by using multi-objective approaches are addressed.

2.2.1 Exhaustive Search Proposals

Frequent association rules. At the beginning of the 90s, a first approach for mining ARs was proposed by *R. Agrawal and R. Srikant* [6], which was based on an exhaustive search methodology. This algorithm, named Apriori, uses prior knowledge of frequent item-set properties to extract ARs. For the sake of determining the

frequent item-sets, this algorithm makes multiple passes over the data. In each iteration, the support for certain item-sets is measured, and they are associated with a counter that stores the sum of transactions in which the corresponding item-set has appeared. This counter is initialized to zero when an item-set is created, and it determines the frequency of occurrence of each item-set. The Apriori algorithm determines that if a length-k item-set is not frequent in data, none of its length-$(k+1)$ super item-sets can be frequent. An item-set becomes a candidate if every one of its subsets is frequent. This property, known as anti-monotone property, determines that every frequent item-set needs to be a candidate too, hence only candidates' frequency is calculated.

Apriori divides the ARM problem into two sub-problems: (1) obtaining all the frequent item-sets in data and (2) extracting all the ARs — according to predefined minimum confidence thresholds — starting from the results obtained in the previous step. This methodology achieves good performance by reducing the size of candidate sets by the anti-monotone property. However, in huge datasets or those with a large number of frequent patterns caused by quite low minimum frequency thresholds, an Apriori-like algorithm may suffer from two non-trivial costs. Firstly, a huge number of candidate item-sets could be generated. Secondly, it is computationally expensive to repeatedly scan the database and check a large set of candidates by pattern matching.

Due to the candidate generation problems, some researchers have focused their studies on a method that mines the complete set of frequent item-sets without a candidate generation step. Thereby, the frequent-pattern growth algorithm is proposed, or simply FP-Growth [7], which is based on a divide-and-conquer strategy. This algorithm firstly compresses the data into a frequent-pattern tree structure, which stores information about frequent patterns. Subsequent frequent-pattern mining will only need to work on the resulting tree instead of the whole data. Additionally, the search technique employed in frequent-pattern mining is a partitioning-based, divide-and-conquer method rather than an Apriori-like level-wise generation of combinations of frequent item-sets. Nevertheless, this approach is not free of difficulties: the memory requirements are significantly increased with the growth of the data size; like the Apriori algorithm, very low minimum support thresholds may cause the discovery of huge numbers of ARs.

Zhenguo et al. [36] have presented an approach that improves FP-Growth both in runtime and the memory consumption. This algorithm is based on a compound single linked list, which is used to improve the structure of the FP-tree. In its mining procedure, the improved version of FP-Growth first scans the data to obtain the set of frequent items and their support counts. This set of frequent items is sorted in descending support count order and determines the header of the single link list. In a second scan of the data, the algorithm processes the items of each transaction and insert the items into the single link list.

Quantitative association rules. Early studies involving exhaustive search algorithms [37] for mining ARs focused on data with categorical items. However, the data in real-world applications usually consist of numerical values (real and integer values), containing many distinct values, so exhaustive search algorithms for ARM cannot be used directly. Because each numerical attribute might have a large amount of values, the ARM problem from numerical data has a much bigger search space than the problem of categorical mining. A well-known solution to deal with numerical values is the data discretization, that is, the division of their domains into intervals followed by applying categorical values to the intervals. One of the shortcomings of using discretization is to choose the correct interval numbers. In [38], *R. Agrawal and R. Srikant* presented the problem of mining QARs, and they pointed out that choosing intervals for numerical attributes is quite sensitive to the support measure and, in consequence, to the confidence measure. If the number of intervals of a numerical attribute is large (intervals with small width), the support of any single interval can be low and rules involving this attribute may not be found.

Class association rules. Some authors [39, 40] have considered the ARM problem under the CARs perspective. *Li et al.* [39] proposed an efficient algorithm for mining the optimal CARs. They proposed an upward closure property of prunning weak ARs before they are generated. This closure property determines that all super-sets of an infrequent item-set must be infrequent. The goal is to discard rules from the resulting set of CARs to obtain a more interpretable set and being as accurate as the original one.

Association rules without thresholds. Some authors [41] have studied the ARM field from the perspective of removing the support and confidence thresholds.

Instead, they have proposed the discovery of the n best rules, so no fixed threshold is previously required. Notice that a knowledge discovery system has to evaluate the relevance of the extracted knowledge so the interest of the n obtained rules should be as high as possible. This algorithm, named Predictive Apriori, presents a trade-off between confidence and support. An important feature of this algorithm is its ability to remove redundant rules. Authors stated that an AR r_1 subsumes another AR r_2 if and only if the antecedent of r_1 is a subset of the antecedent of r_2, and the consequent of r_1 is a superset of r_2, that is, r_1 predicts more attribute values than r_2. The algorithm guarantees that the n rules provided are not redundant.

Similarly to Predictive Apriori, *Lin et al.* [42] proposed an approach for mining ARs particularly well-suited to be used in collaborative recommender systems. Actually, notict that most of the existing ARM algorithms are not suitable for the recommendation domain, since they extract many rules that are not relevant to a given user, and also requiring a minimum support value that often leads to a huge number of rules. The algorithm proposed by *Lin et al.* adjusts the minimum support so that the number of rules discovered is within a previously fixed range. This algorithm avoids excessive computation time and guarantees that enough rules are provided.

Infrequent/rare association rules. Despite the fact that mining frequent patterns is a major task in ARM, currently there has been increasing interest in the extraction of unusual or infrequent patterns. In early studies, the problem of finding infrequent patterns was originally addressed by using algorithms for mining frequent patterns. An algorithm for discovering PRAR, called Apriori-Inverse, was proposed by *Koh and Rountree* [19]. During the search process, Apriori-Inverse keeps those items with a support value greater than a minimum support threshold but less than a maximum value. Then, similarly to Apriori, a set of ARs is obtained by using a confidence threshold over all the possible combinations of items previously obtained. Consequently, this algorithm mines a lot of RARs since the support of each one is less than or equal to that of the item with minimum support. Notice that this algorithm does not find any IRAR, since it would never consider item-sets that have a support value greater than the maximum threshold.

Another RARM algorithm, called ARIMA, was first proposed by *Szathmary et al.* [43] as a naïve approach. Unlike the Apriori-Inverse algorithm, ARIMA is not

limited to calculating PRAR. This algorithm firstly mines the minimal rare item-sets, i.e., those rare item-sets whose proper subsets are frequent. Thus, whenever a candidate k-item-set survives the frequent $k-1$ subset test, but proves to be rare, it is kept as a minimal rare item-set. Then, the algorithm finds their proper supersets, avoiding zero item-sets, i.e., those having a support of zero. Similarly to Apriori, RARs are generated using the set of rare item-sets mined. For this final process, it is necessary to satisfy a minimum confidence threshold.

Authors of the ARIMA algorithm proposed two different ways of mining rare item-sets [43]. A first version, called Apriori-Rare, is a slightly modified version of Apriori, finding all frequent item-sets and storing the minimal rare item-sets discovered. *Szathmary et al.* [44] also proposed a much more efficient proposal, the MRG-Exp algorithm, which reduces the search space by avoiding exploring all frequent item-sets. Instead, it is sufficient to search for frequent generators only. An item-set X is a generator if it has no proper subset with the same support, i.e., $\forall Y \subset X, support(X) < support(Y)$. The Apriori-Rare requires a higher computational time since it lists all frequent item-sets before reaching the minimal rare item-sets whereas MRG-Exp explores only the frequent generators. Both versions obtain association rules from the set of minimal rare item-sets, the number of rules discovered being smaller than the naïve approach.

Although it is more efficient to obtain rare association rules by applying specific algorithms for the extraction of infrequent patterns, it is also possible to follow the classic *Apriori* proposal to obtain rare association rules from the set of infrequent items. An example is *Apriori-Infrequent*, which has been used for intrusion detection [25]. During the candidate generation phase, an infrequent item-set is considered as a candidate if it comprises only frequent elements or items so this algorithm seeks for IRARs. Finally, the resulting set of candidate elements is used to obtain rules that exceed a minimum confidence threshold.

2.2.2 Evolutionary Algorithms

EAs have been widely used in DM tasks, where the process of searching for solutions requires some optimization. Many researchers have focused their solutions to

the ARM problems from an evolutionary perspective [16, 45], facing up to the exhaustive search approaches' computational and memory requirements. Most of the existing evolutionary approaches are based on GAs [8, 12], having a fixed-length chromosome, which is not as flexible as expected.

Categorical association rules. *Olmo et al.* proposed in [46] the first approximation to the extraction of ARs by employing ant programming, a nature-inspired optimization meta-heuristic based on the behavior and organization of ant colonies. This algorithm, named GBAP-ARM (Grammar-Based Ant Programming for Association Rule Mining), measures the quality of the rules discovered by using the weighted average between support and confidence. This measure represents the harmonic mean of support and confidence. The goal is to maximize this average value, which is contained into the interval [0,1]. Despite the excellent results, an important drawback of GBAP-ARM is that it can not be applied on numerical data, so this kind of attributes still require to be discretized.

In ARM, most research studies have focused on improving computational efficiency, but it is also of interest the determination of threshold values for support and confidence. For this purpose, a particle swarm optimization algorithm [47] was proposed by *Kuo et al.*, which searches for corresponding support and confidence values as minimal thresholds. This algorithm works on binary data so transactional data have to be transformed into binary data type, and each chromosome includes the number of the item comprised in the rule. Finally, after finding the best particle, its support and confidence are recommended as the value of minimal support and minimal confidence. These optimal values are employed in ARM to extract valuable information.

Quantitative and Negative association rules. The use of EAs has given rise to different kind of algorithms for mining both QARs and NARs. An example of mining QARs by means of GAs is QuantMiner [11], which uses a genetic algorithm to learn appropriate intervals on a set of rules previously computed and searches for the optimization of a given quality criteria. Individuals are represented as sets of items of the form $attribute_i \in [l_i, r_i]$, where l_i and r_i state for the minimum and maximum selected bounds of the i-th attribute, respectively. The evolutionary process modifies the interval amplitudes and evaluate the solutions by the confidence measure. If the predefined minimum confidence threshold is exceeded, then the

interval amplitudes are considered in favour of those with small amplitudes. Additionally, those rules that infrequently occur are penalized by decreasing drastically their fitness values.

QARGA [45] (Quantitative Association Rules by Genetic Algorithm) is other evolutionary approach that finds existing relationships among numerical attributes. This algorithm is a real-coded GA that does not perform previous attribute discretization, that is, it handles numerical data during the whole mining process. In QARGA, each individual is represented by an array of fixed length n (the number of available continuous attributes in data). Each gene represents the upper and lower limit of the interval of each continuous attribute. Additionally, each gene comprises a value that represents the type of attribute. Thus, each attribute could not belong to the individual, belong to the antecedent, or belong to the consequent. QARGA is based on the iterative rule learning process, so the EA is applied in each iteration obtaining one rule per iteration (the best individual discovered).

The mining of ARs with quantitative conditions has been the cornerstone of many GAs. *Alcalá et al.* [9] studied the effectiveness of GAs for mining QARs. In this study, they present an experimental study of two real-world datasets to show the behaviour of three GAs for mining QARs: ARMGA (Association Rule Mining Genetic Algorithm) [8], GENAR (GENetic Association Rules) [48] and GAR (Genetic Association Rules) [16]. ARMGA [8] is an interesting GA for mining ARs. This algorithm encodes each rule in a single chromosome, where the first gene of each individual chromosome is an indicator that separates the antecedent from the consequent of the rule, and the following genes of the chromosome are each of the items — a length-k rule is represented by k + 1 positive integers. Notice that quantitative attributes are divided into intervals according to a level of granularity predefined by the user. The ARMGA algorithm uses mutation (it changes genes with a certain probability) and crossover (following a two-point strategy) as genetic operators to obtain new individuals in a given generation. Finally, each individual is evaluated according to the relative confidence, which is defined as the degree of relationship between the antecedent and consequent.

As for the GENAR and GAR algorithms, they are very similar approaches for mining ARs. In fact, GAR is an extension of the GENAR algorithm presented in [48]. In both algorithms, individuals are represented as a list of genes grouped in threes.

In each group, the first gene represents the attribute, whereas the remaining genes indicate the minimum and maximum limits of the interval. An important drawback of GENAR is that it is necessary to prepare the data to indicate which attributes form part of the antecedent and which ones are included in the consequent. Nevertheless, this process is not necessary in the GAR algorithm because it seeks for the frequent item-sets and the rules are built from them.

Sometimes, it is useful to mine not only positive ARs but also negative ARs. These ARs include conditions within the antecedent or the consequent being negated. *Alatas et al.* [12] proposed a GA for mining both negative and positive quantitative rules, the rules considered including, at least, one negative condition. Thus, the rules discovered by this algorithm could be of the form $\neg[a_1, a_2] \rightarrow [b_1, b_2]$, $[a_1, a_2] \wedge [b_1, b_2] \rightarrow \neg[c_1, c_2]$, $[a_1, a_2] \wedge \neg[b_1, b_2] \rightarrow [c_1, c_2]$, etc. A negative condition, for instance, $\neg[a_1, a_2]$, determines that the attribute is not in the range $[a_1, a_2]$ and, a_1 and a_2 show the lower and upper bound of the interval. In this GA, each chromosome consists of genes that represent the attributes and intervals. Thus, the i-th gene encodes the i-th attribute. Each gene has a value to determine whether the attribute will be in the antecedent, in the consequent or not be involved in the rule. Moreover, each gene has a value to determine whether the attribute will be positive or negative, and two values to indicate the lower and upper bounds of the interval. In this algorithm, the chromosome length will be $m \times 4$, m being the number of attributes to be mined.

Fuzzy association rules. In the last years, the ARM problem has been addressed by means of the fuzzy set theory. The use of fuzzy sets to describe associations [49] improves the interpretation of the knowledge represented by the rules. *Herrera et al.* [50] proposed a fuzzy algorithm for extracting both ARs and membership functions from quantitative data by means of an evolutionary methodology. This algorithm is based on the well-known CHC evolutionary model [51], which presents a good trade-off between exploration and exploitation.

2.2.3 Grammar-Guided Genetic Programming

Genetic programming (GP) [52] is an evolutionary and very flexible heuristic technique that enables complex pattern representations to be used. GP represents the

solutions to the problem by means of trees, enabling any kind of function and domain knowledge to be considered. Solutions represented as trees include two kind of symbols. Leave nodes correspond to variables and constants, whereas internal nodes correspond to operators and functions.

GP [53] was proposed as an extension of GAs to build computer programs by means of a complex representation language. The original goal of GP was to find an optimized solution from a search space composed by all possible computer programs. Nevertheless, GP is currently used to evolve other types of knowledge, like rule-based systems [54].

Recently, a new extension of GP was established for the sake of formally define the problem constraints [55]. This new methodology, named grammar-guided genetic programming (GGGP or G3P) [56], employs a context-free grammar (CFG) that generates any feasible solution to the problem under study. In this way, the grammar constrains the search space and solutions are constructed by applying a set of productions rules. The grammar is defined based on the problem under study or the kind of solutions to be obtained.

A CFG could be formally defined as a four-tuple (Σ_N, Σ_T, P, S), in which Σ_N is the non-terminal symbol alphabet, Σ_T denotes the terminal symbol alphabet, P stands for the set of production rules, S for the start symbol, and Σ_N and Σ_T are disjoint sets, i.e., $\Sigma_N \cap \Sigma_T = \emptyset$. Any production rule follows the format $\alpha \rightarrow \beta$ where $\alpha \in \Sigma_N$, and $\beta \in \{\Sigma_T \cup \Sigma_N\}^*$. Beginning from the start symbol S, each individual is represented in a derivation syntax-tree as a sentence conformant to the grammar. The benefits of using grammars are the ability to define syntax constraints, providing expressiveness, flexibility, and the ability to restrict the search space. To obtain individuals, a number of production rules are applied from the set P. This process begins from the start symbol S. The maximum number of production rules is properly predefined to control bloat, so there is a maximum tree size enforced. Finally, it should be noted that, in G3P trees, leave nodes correspond to terminal symbols and internal nodes correspond to non-terminal symbols.

Grammars bring a number of benefits to GP [57], one of the most important is the flexibility and ability to restrict the search space. Grammars are mainly used to overcome the problem of generating valid programs and preserving its correctness. There are many fields where grammars could be used to encode reasonably

complex prior knowledge. This prior knowledge is highly important since it provides dimensional consistency and it drastically reduces the number of admissible solutions.

Generally, the functions and operators used in G3P are quite similar to those used in GP. However, there are some differences between both techniques. Firstly, individuals in the initial population should be valid solutions according to the context-free grammar defined by G3P. On the contrary, GP enables invalid individuals to be generated, so it hampers the convergence to the optimal solution. Secondly, the genetic operators should consider the production rules of the grammar. Thus, the crossover operator only swaps compatible sub-trees. Similarly, the mutation genetic operator is required to produce only feasible individuals.

As *McKay et al.* described in [58], the first use of grammars in GP systems were independently implemented at about the same time by three different researchers. *Whigham* [59] proposed the CFG-GP system, in which a CFG was used to generate derivation trees. *GeyerSchulz* [60] derived his very similar G3P approach for learning rules for expert systems. *Wong and Leung* [61] proposed an approach that uses PROLOG to generate programs to learn first order relations.

In the context of DM, G3P has been largely used in supervised learning [62–64], whereas its use in unsupervised learning, like ARM, is still unexplored. Regarding the supervised learning task, *Zafra et al.* proposed the use of grammars to enforce constraint in the GP trees in the field of multiple instance classification [62, 63]. The authors proposed an algorithm that expresses the information in the form of IF-THEN classification rules. An important advantage of this approach, called G3P-MI (Grammar-Guided Genetic Programming algorithm for Multiple Instance learning), is its ability to represent the knowledge discovered in a comprehensible way. Thus, the data miner can understand the results, being able to make a well-informed decision.

G3P has been also applied to multi-label classification [64] obtaining interesting results. The classifier provided by this algorithm is a rule-based that consists of several IF-THEN classification rules obtained from the evolutionary process. In order to obtain the labels for each example, the algorithm uses the aggregation of the consequents from the rules whose antecedent satisfies the example.

To sum up, solutions in G3P are expressed in a tree shape structure conformant to a context-free grammar. This grammar enables solutions to be represented for any domain, and the shape, size and structural complexity to be constrained by that grammar. Indeed, a grammar could be easily adapted to different problems, producing rules with varied structures. The use of G3P in ARM is therefore justified, as the number of different rules could be high enough to be computationally intractable by means of an exhaustive search methodology.

2.2.4 Multi-Objective Proposals

Generally, there are often problems that require simultaneously optimize more than one objective. Under these circumstances, it is not possible to determine only one single best solution to them but a satisfactory trade-off [65]. A sample application domain where multi-objective optimization could be applied is the development of a new vehicle by a company. In this context, there are many issues that should be bore in mind, like the optimal combination of vehicle weight and engine power to efficient driving; that is, it is required to reduce the fuel consumption as much as possible. Additionally, the cost of the development of such a vehicle has to be minimized, whereas maximum performance and comfort are desired.

In multi-objective optimization [66], the goal is to obtain a set of solutions that all equally fit for the optimum, and this set of solutions is known as Pareto Optimal Front (POF). None of the solutions included in the POF is better than the other solutions for all the objectives, so all of them are fairly acceptable.

In the multi-objective field, when a solution is better than any other for each objective, then it is said that such a solution dominates the others. In a formal way, given a set of objective functions $\mathcal{F} = \{f_1, f_2, f_3, ..., f_n\}$, a solution s belongs to the POF if there is no other solution s' that dominates it. A solution s' dominates s if and only if $f_i(s') \geq f_i(s) \forall f \in F$ and $f_i(s') > f_i(s)$ for at least one $f \in F$. However, there are problems where it is necessary to minimize the objectives instead of maximizing them. In such problems, a solution s belongs to the POF if there is no other solution s', where $f_i(s') \leq f_i(s) \forall f \in F$ and $f_i(s') < f_i(s)$ for at least one $f \in F$.

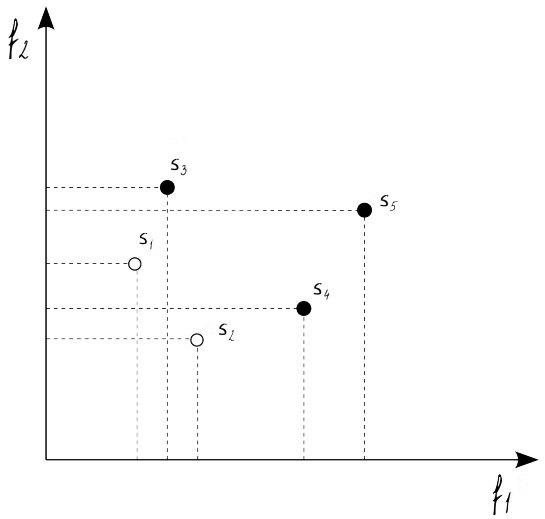


Figure 2.3: Sample Pareto optimal front comprising two solutions from a set of five solutions

For a better understanding, consider five solutions (s_1, s_2, s_3, s_4 and s_5) to a specific problem, where two objectives (f_1 and f_2) are considered to be minimized, as properly depicted in Figure 2.3. Analysing these five solutions, it can be determined that s_1 is not dominated by any other solution because $f_1(s_1)$ is the best value. None of the remaining solutions obtains a lower value for the objective f_1. Similarly, s_2 is not dominated by any other because $f_2(s_2)$ is the best value for the objective f_2. On the other hand, s_3 is dominated by s_1 because $f_1(s_1)<f_1(s_3)$ and $f_2(s_1)<f_2(s_3)$. Note that s_4 is dominated by s_2 because $f_1(s_2)<f_1(s_4)$ and $f_2(s_2)<f_2(s_4)$. Finally, s_5 is dominated by both s_1 and s_2. Therefore, the set of solutions that comprises the POF is formed by s_1 and s_2, i.e. solutions that are equally acceptable because neither of them is better than the other for all the objectives. On the contrary, s_3, s_4 and s_5 could be simply removed, since they are sub-optimal solutions.

Additionally, the nature itself of the multi-objective optimization problems, which could be complex and very time-consuming, has given rise to the use of new disciplines to deal with this kind of problems. In this situation, the use of EAs to solve multi-objective problems has received increasing attention mainly motivated by the population-based nature of EAs, which enables a set of solutions to be obtained in a single execution. This synergy is known as evolutionary multi-objective optimization (EMO) [66].

In ARM problems, it is often of interest to simultaneously optimize more than one quality measure. Thus, not only a frequent rule is of interest, but its reliability or even the high correlation between its antecedent and consequent are also desired. Despite the fact that most existing proposals for mining ARs are designed as a single-objective problem [6, 7, 11], many researchers have focused their studies on the extraction of interesting ARs by means of a EMO methodology [67, 68].

Ghosh et al. [69] proposed a multi-objective algorithm for mining useful and interesting rules from market-basket data. The authors presented a GA to simultaneously optimize comprehensibility, interestingness, and confidence as measures. The proposed GA associates two genes to each attribute. If these two genes are 00 then the attribute is considered for its addition to the antecedent part of the rule, whereas the values 11 indicate that the attribute only appears in the consequent part. Finally, the other two combinations (01 and 10) will indicate the absence of the attribute in either of these parts, antecedent or consequent.

Another relevant approach in multi-objective optimization is called MODENAR [67] (Multi-Objective Differential Evolution algorithm for mining Numeric Association Rules), which extracts numeric association rules. In this approach, a search strategy for mining accurate and comprehensible rules is carried out by applying a multi-objective differential evolution method [70]. During the mining process, four objectives are considered. For each association rule, the support, confidence and comprehensibility need to be maximized, whereas the amplitude of the intervals within each rule is minimized. MODENAR represents individuals by means of decision variables that represent the items and intervals, where the i-th item is encoded in the i-th decision variable. Each decision variable includes three parts: (1) a value that determine whether the item is included in the antecedent, in the consequent, or not considered; (2) the lower bound of the item interval; and (3) the upper bound of the interval.

Kumar et al. [71] proposed an approach for mining association rules by using the well-known multi-objective evolutionary algorithm NSGA-2 [72]. During the evaluation stage, different measures were used, such as interestingness, comprehensibility, support, confidence, etc. Finally, a series of experiments were carried out by taking three different measures each time and making a comparison with the

traditional Apriori algorithm. Authors stated that some of the rules obtained using NSGA-2 cannot be obtained using traditional methods like Apriori because support for such rules is very low.

Similarly to the algorithm proposed by *Kumar et al.*, a multi-objective ant programming algorithm based on NSGA-2 for mining ARs was proposed by *Olmo et al.* [46]. This algorithm, named MOGBAP-ARM (Multi-Objective Grammar-Based Ant Programming for Association Rule Mining), looks for simultaneous acquisition of very frequent and reliable rules. MOGBAP-ARM uses an auxiliary population to keep the POF discovered in a given generation. The individuals of this auxiliary population are merged with those created at the next generation, so a ranking of this population will determine the non-dominated solutions. The algorithm returns the set of non-dominated solutions from this auxiliary population.

3

Grammar-Guided Genetic Programming for Mining Association Rules

In this chapter, we present a proposal based on G3P for mining association rules, called the G3PARM (Grammar-Guided Genetic Programming for Association Rule Mining) algorithm. This proposal is the first approach using G3P in the ARM field. It uses a CFG that allows defining syntax constraints in the programs that represent the ARs and may be used in both numerical and categorical domains. G3PARM makes use of an auxiliary population of individuals (each individual represents an AR) that exceeds certain support and confidence thresholds. This auxiliary population has a fixed size and keeps the best individuals obtained with the passing of generations. The G3PARM algorithm obtains solutions without needing the large amounts of memory normally used by exhaustive search algorithms, and within specified time limits.

To analyse the effectiveness of our proposal, we compare it versus exhaustive search [7] and genetic [8, 73] algorithms for mining ARs by using different real-world datasets. The results of empirical comparison demonstrate that our proposal obtains rules with high support, high confidence and high coverage of data instances. Moreover, the scalability of these algorithms is compared. Finally, their behaviour by increasing the number of attributes and the number of instances in the datasets is studied.

3.1 The G3PARM Algorithm

G3PARM is an approach that serves to obtain ARs for any domain or problem. This algorithm makes use of G3P to define expressive and understandable individuals. This section presents our model with its major characteristics: the representation of individuals using a CFG, the treatment of categorical and numerical attributes, genetic operators, the evaluation process, and the evolutionary algorithm.

3.1.1 Encoding

In G3P, each problem solution is composed of two distinct components: (a) a genotype, represented by a derivation syntax-tree, and (b) a phenotype, that represents the rule. The phenotype represents the complete rule consisting of an antecedent and a consequent. Both the antecedent and consequent of each rule are formed by a series of conditions that contains the values of certain attributes that must all be satisfied.

Figure 3.1 shows the CFG through which the population individuals are encoded in the proposed algorithm. As stated in Chapter 2, a CFG is formally defined as a four-tuple (Σ_N, Σ_T, P, S) where $\Sigma_N \cap \Sigma_T = \emptyset$, Σ_N is the alphabet of non-terminal symbols, Σ_T is the alphabet of terminal symbols or tokens, P is the set of production rules and S stands for the start symbol. Productions have the format $\alpha \rightarrow \beta$ where $\alpha \in \Sigma_N$, and $\beta \in \{\Sigma_T \cup \Sigma_N\}^*$. Individuals are defined as a derivation syntax-tree where the root is the symbol S, the internal nodes contain only non-terminal symbols and the leaf nodes contain only tokens. Each individual is a

G = $(\Sigma_N, \Sigma_T, P, S)$ with:

S = {*Rule*}
Σ_N = {*Rule, Antecedent, Consequent, Comparison, Categorical_Comparator, Categorical_Attribute_Comparison, Numerical_Comparator, Numerical_Attribute_Comparison*}
Σ_T = {*'AND', '!=', '=', '<=', '<', '>=', '>', 'name', 'value'*}
P = {*Rule = Antecedent, Consequent ;*
Antecedent = Comparison | 'AND', Comparison, Antecedent ;
Consequent = Comparison | 'AND', Comparison, Consequent ;
Comparison = Categorical_Comparator, Categorical_Attribute_Comparison | Numerical_Comparator, Numerical_Attribute_Comparison;
Categorical_Comparator = '!=' | '=' ;
Numerical_Comparator = '<=' | '<' | '>=' | '>' ;
Categorical_Attribute_Comparison = 'name', 'value' ;
Numerical_Attribute_Comparison = 'name', 'value' ;}

Figure 3.1: Context-free grammar expressed in extended BNF notation

sentence generated by the grammar and represented by a derivation syntax-tree. To generate the sentence, the derivation syntax-tree is obtained by applying a series of derivation steps.

Grammars can be used either to define the valid expressions of a computer language, to impose restrictions, or to describe constraints on interactions within systems. In particular, the use of a CFG allows individuals to be obtained for each specific problem or domain [58]. Validating the CFG proposed (see Figure 3.1) requires the analysis of the language denoted by the grammar. Thus, consider the following language definition $L(G3PARM) = \{(AND\ comp)^n\ comp \rightarrow (AND\ comp)^m\ comp : n >= 0, m >= 0\}$, where the symbol "$\rightarrow$" is depicted to represent the existing separation between the antecedent and consequent in the derivation syntax tree, and *comp* states for a comparison expression comprised by the AR. Notice that n is greater than 0 if the rule has more than one condition in the antecedent. Similarly, if the consequent contained more than one condition, then m would be greater than 0. In short, any string resulting from this expression belongs to the context free language denoted by the grammar, which represents a rule (i.e. individual)

Table 3.1: Metadata defined in the weather dataset.

Attributes	**Values**
outlook	sunny, overcast, rainy
temperature	hot, mild, cool
humidity	high, normal
wind	true, false
play	yes, no

containing at least one condition in the antecedent and at least one condition in the consequent (the clause AND is used to connect a sequence of conditions).

The chromosome encodes its expression using a pre-order traversal of the parse tree. It should be noted that the terminal grammar symbol *'name'* is determined in the feasible attributes. For each one, the assigned value is determined among their available values. For example, using the well-known *weather*[1] dataset, a set of feasible values is obtained for each attribute as shown in Table 3.1.

Figure 3.2 shows a sample representation of an individual in the G3PARM algorithm using the aforementioned *wheather* dataset (see Table 3.1). The syntax-tree structure is obtained according to the previously defined grammar. As mentioned above, the phenotype represents the rule and is obtained by eliminating non-terminal genotype symbols: $(outlook\ !=\ sunny\ AND\ windy\ !=\ false) \rightarrow (play = no)$.

The generation of an individual is carried out from the start symbol of the grammar. From this symbol, the algorithm searches for a feasible solution through the random application of production rules belonging to set P, until a valid derivation chain is reached. The maximum number of derivations is determined by a value provided in the algorithm configuration parameters.

To carry out the derivation of the non-terminal symbols that appear in the grammar, we use the cardinality concept which is defined as the number of derivation steps necessary for creating only terminal symbols. If a non-terminal symbol can be derived in several ways, the cardinality of this non-terminal symbol will be

[1]This dataset is included in the Weka dataset collection, which can be obtained from the Weka machine learning project. http://www.cs.waikato.ac.nz/~ml/index.html.

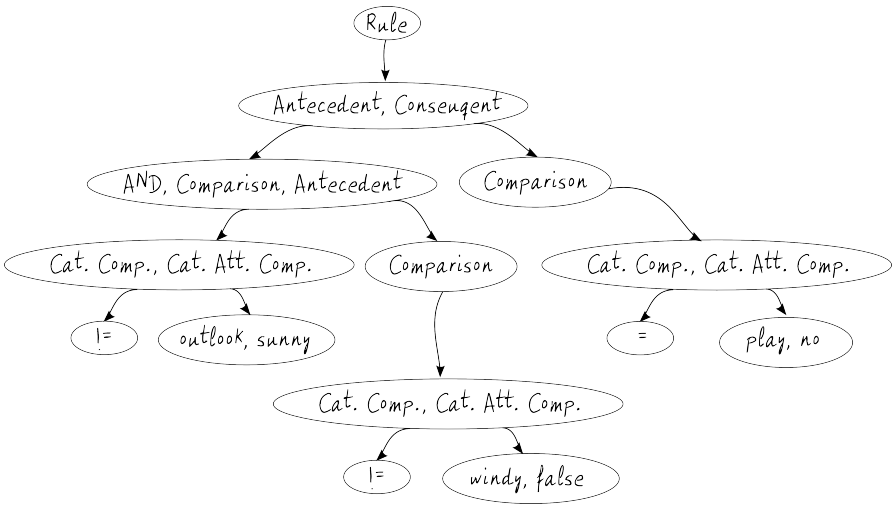


Figure 3.2: Sample individual genotype represented in a syntax-tree structure

determined by the sum of the cardinalities of each of the possible derivations of this symbol. In the derivation process, a production between the minimum and maximum number of derivation steps is randomly selected. Once a production is chosen, the maximum number of derivation steps is decreased by one. The higher the cardinality of a symbol, the higher the probability of choosing this symbol as a production, taking into account the maximum number of derivations.

3.1.2 Dealing with Categorical Attributes

Categorical attributes are commonly used in most algorithms for mining ARs. A categorical attribute Att of an AR is an attribute that has a discrete unordered domain $\mathcal{D}$. In our proposal, an expression of the form $Att = u$ or $Att \neq u$ is allowed, where Att is a categorical attribute and u is a value in the domain $\mathcal{D}$ of Att. The expression $Att \neq u$ indicates that Att takes any value in $\mathcal{D} \backslash \{u\}$.

For a domain $\mathcal{D}$ of a categorical attribute, the support of any value u in this domain might be very low. Using the operator $\neq$ it is possible to obtain a higher support. For example, using the attribute Att in a domain $\mathcal{D} = \{a, b, c, d\}$, the support of $Att = a$ is 0.14, whereas the support of $Att \neq a$ will be 0.86.

3.1.3 Dealing with Numerical Attributes

Numerical attributes are usually defined within a wide range of numerical values. It is useless to work on all possible numerical values, as done with categorical values, because in most cases, a given numerical value will not appear frequently and each numerical attribute might have an enormous number of values appearing in the dataset. A classical way to deal with numerical attributes is to divide their domains into different intervals by the discretization pre-processing operation. This process may create new problems, e.g., the greater the number of intervals, the greater the execution time since the number of items per record soars. There is a trade-off between faster execution time with fewer intervals and reducing information loss with more intervals.

The algorithm proposed uses a new alternative to deal with numerical attributes using different equal-width points in the range of values and using these points as the value of the numerical attribute. The upper and lower bounds of the range of values will not be taken into account to avoid rules that always occur, which will therefore be uninteresting rules since they do not provide new information. For example, having the numerical attribute Att, if we obtain five points from the range of values $[1000, 2000]$, then we will have as possible values: 1250, 1500 and 1750. Thus, following expressions could be obtained: $Att >= 1250$, $Att < 1250$, $Att > 1250$, etc. A rule always occurs if we take the uppermost (value 2000) or lowest (value 1000) bounds of the range (e.g., $Att >= 1000$).

One of the main problems when dividing the range of values into intervals is the correct definition of the width of each interval. Small intervals could imply that the support for any single interval might be low and rules involving this attribute may not be found. On the other hand, big intervals could imply uninteresting rules. Our proposal avoids this problem since it allows interval rules to be obtained by defining a different interval width in one same attribute. For example, using the attribute Att mentioned above, we can obtain the interval $Att > 1250$ AND $Att < 1500$ for $(1250, 1500)$ or $Att > 1250$ AND $Att <= 1500$ for $(1250, 1500]$.

Unlike the discretization method, our proposal deals directly with numerical attributes and does not divide the numerical attributes into intervals by assigning a categorical value to each interval. It avoids the problem of obtaining small intervals

with possibly low support and where rules involving this attribute might be not found. Therefore, for any distance between points in the range of an attribute's values, it is possible to obtain support that exceeds the minimum support threshold. In addition, it is not necessary to perform a previous discretization step (data preprocessing), which is another advantage of our proposal.

3.1.4 Genetic Operators

In this section, two new genetic operators, developed to generate new individuals in a given generation of the EA process, are presented. With these genetic operators, we try to obtain rules with higher support than the original ones. The support of an AR depends on the frequency of its attributes, so the greater the frequency of occurrence of rule attributes, the greater the probability of increasing the support of the entire rule. However, increasing the frequency of occurrence of an attribute does not always imply an increase in the support of the entire rule.

Crossover. The pseudo-code of this genetic operator is shown in Algorithm 1, and Figure 3.3 shows this process graphically. Like crossover operators in most GP approaches [58], the main idea of this operator is to obtain two new individuals

Algorithm 1 G3PARM crossover

Require: $parents$
Ensure: $offspring$
1: $offspring \leftarrow \emptyset$
2: **for all** individuals in $parents$ **do**
3: $parent1, parent2 \leftarrow$ getIndividuals($parents$)
4: **if** random() < crossoverProbability **then**
5: $attribute1 \leftarrow$ getAttributeMaximumSupport($parent1$)
6: $attribute2 \leftarrow$ getAttributeMinimumSupport($parent2$)
7: **if** $attribute2$ is not contained in $parent1$ **then**
8: $offspring \leftarrow offspring \cup$ exchange($parent1$,$attribute1$, $attribute2$)
9: **else**
10: $offspring \leftarrow offspring \cup parent1$
11: **end if**
12: **if** $attribute1$ is not contained in $parent2$ **then**
13: $offspring \leftarrow offspring \cup$ exchange($parent2$,$attribute2$, $attribute1$)
14: **else**
15: $offspring \leftarrow offspring \cup parent2$
16: **end if**
17: **end if**
18: **end for**
19: **return** $offspring$

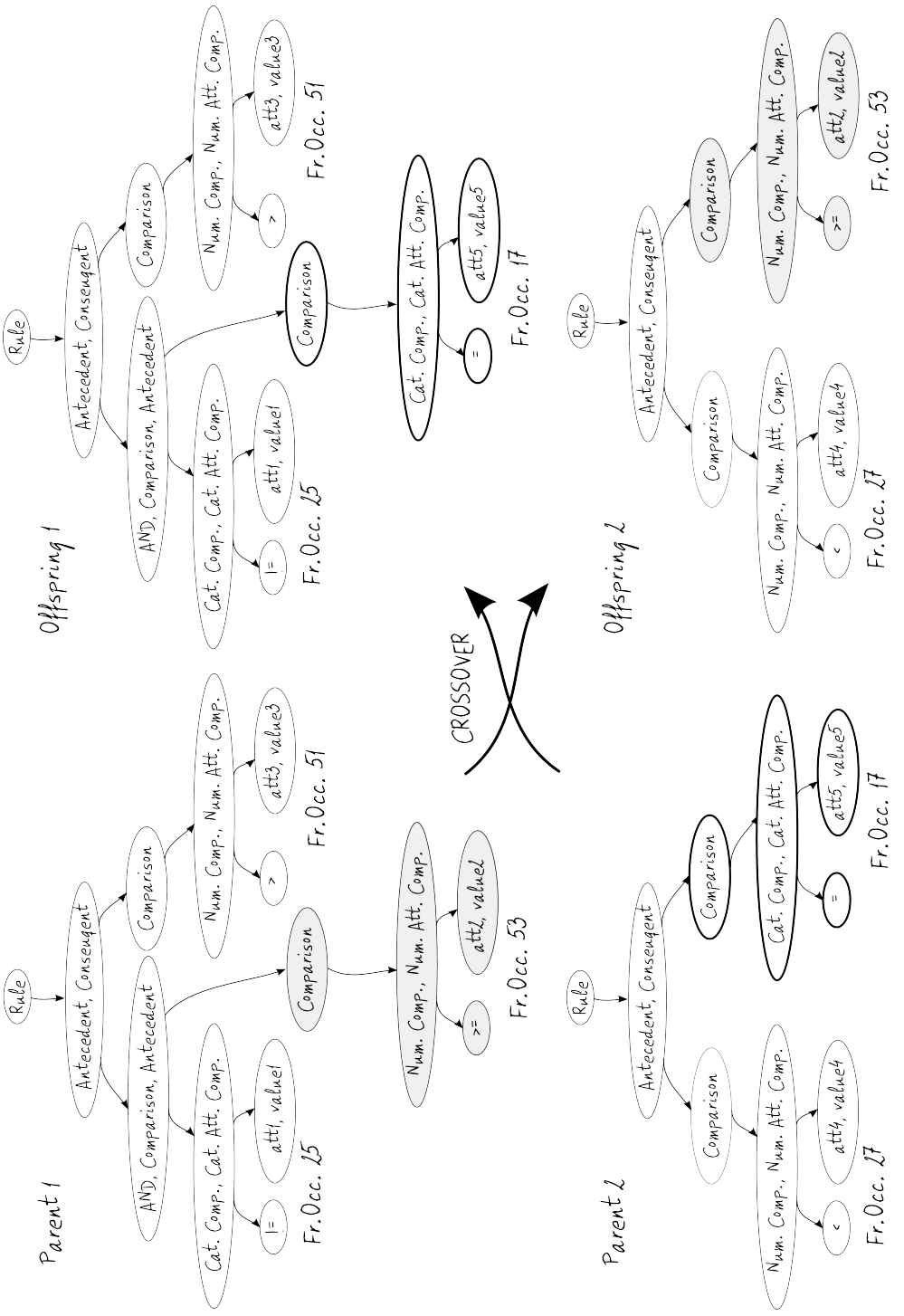


Figure 3.3: Example of crossover operation

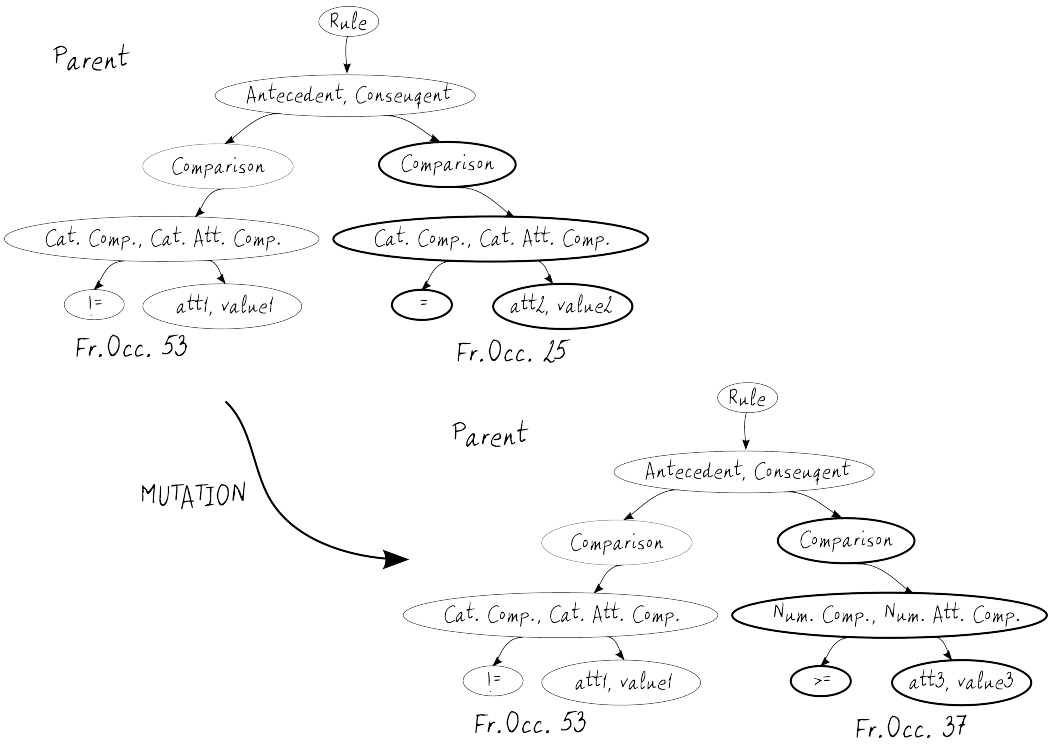
from two different parents. In this proposal, the attribute with the lowest frequency of occurrence in one parent is swapped with the highest one in the other parent. This enables us to obtain at least one individual whose attributes appear more frequently than at least one of the parents.

The operator creates new individuals by exchanging the derivation sub-trees of two parents ($Parent1$ and $Parent2$) from two selected nodes with the same non-terminal symbol in each of them. To select a node from $Parent1$, the attribute of the rule with the highest support is chosen for exchange and then we take the non-terminal symbol *Comparison* from which this attribute is derived. To select a node from $Parent2$, the attribute of the rule with lowest support is chosen to be exchanged and the non-terminal symbol *Comparison* is taken. In this example, we consider the frequency of occurrence of the attributes are: 25 for *'att1 != value1'*, 53 for *'att2* $\geq$ *value2'*, 51 for *'att3* $>$ *value3'*, 27 for *'att4* $<$ *value4'*, and 17 for *'att5* $=$ *value5'*.

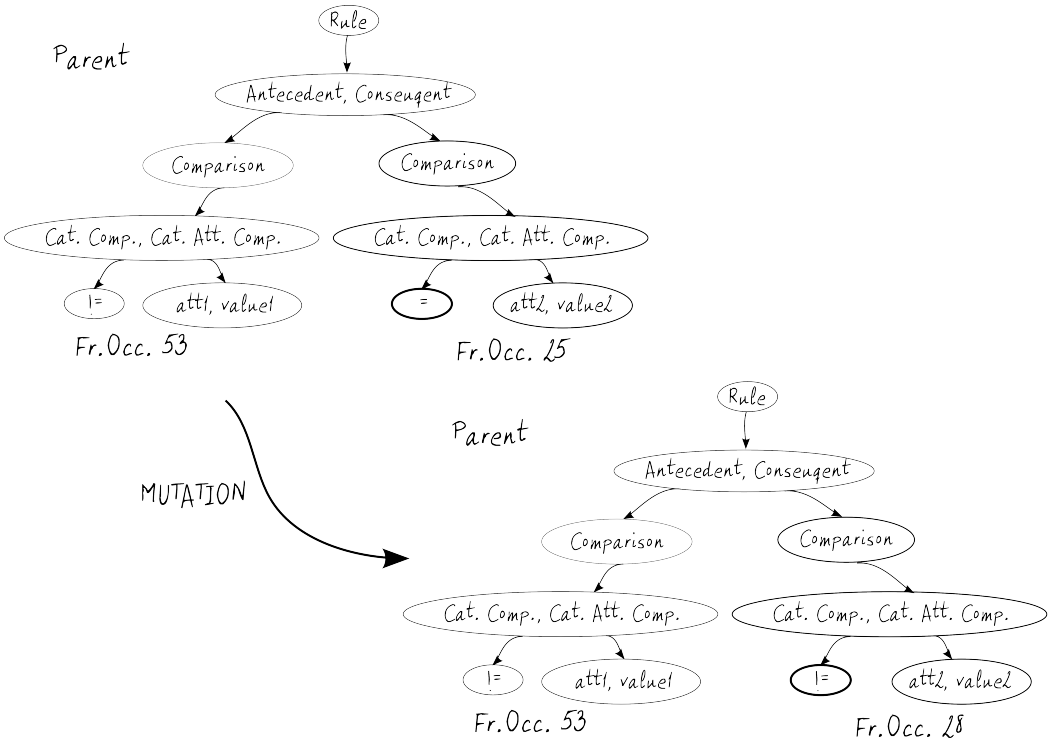
Mutation. The mutator operator process is shown graphically in Figure 3.4 and the pseudo-code is shown in Algorithm 2. The main idea of this genetic operator is an attempt to obtain an individual with greater support. In this way, the attribute with the lowest frequency of occurrence is mutated to try to obtain an attribute with a higher frequency. Based on this attribute, a node from this sub-tree is selected randomly and the next action is based on the symbol type. There are two possibilities: if the node is a non-terminal symbol (e.g. *Comparison*), a new derivation is performed from this symbol (see Figure 3.4(a)). On the other hand, if the node selected is a terminal symbol, it changes the value of the terminal symbol

Algorithm 2 G3PARM mutation

Require: $parents$
Ensure: $offspring$
1: $offspring \leftarrow \emptyset$
2: **for all** individuals in $parents$ **do**
3: $parent \leftarrow$ getIndividual($parents$)
4: **if** random() $<$ mutatorProbability **then**
5: $attribute \leftarrow$ getAttributeMinimumSupport($parent$)
6: $offspring \leftarrow offspring \cup$ getNewDerivation($attribute$)
7: **end if**
8: **end for**
9: **return** $offspring$



(a) Example of mutation operator selecting a non-terminal symbol



(b) Example of mutation operator selecting a terminal symbol

Figure 3.4: Examples of mutation operation

at random (Figure 3.4(b)). In this example, we consider that the frequency of the attribute occurrence is 53 for *'att1 != value1'* and 25 for *'att2 = value2'*.

3.1.5 Evaluation

One of the main processes in any evolutionary model is the evaluation procedure (see Algorithm 3), which determines how promising a certain individual is, that is, how close a given solution comes to achieving the aim. The evaluation process is therefore responsible for assigning a fitness function to each feasible solution, and this procedure is executed in each generation of the evolutionary model. The first step in the evaluation of an association rule is to check whether the discovered expression is consistent with a valid AR, that is, if there is no coincidence between features in the antecedent and consequent of the rule ($A \cap C = \emptyset$). If there is a

Algorithm 3 Evaluation process

Require: *dataset*, *individual_genotype*
1: *count* $\leftarrow 0$
2: *instances_number* $\leftarrow 0$
3: $satisfy \leftarrow TRUE$
4: **if** antecedent and consequent in *individual_genotype* are disjoint sets **then**
5: **for all** *instances* in *dataset* **do**
6: *instances_number* $++$
7: $satisfy \leftarrow TRUE$
8: **for all** *elements* in *individual_genotype* **do**
9: **if** *element* is a terminal symbol **then**
10: **if** *element* is a logic operator **then**
11: *attribute_operator* $\leftarrow$getOperator(*element*)
12: **else**
13: *attribute_name* $\leftarrow$getName(*element*)
14: *attribute_value* $\leftarrow$getValue(*element*)
15: **if** *element* does not satisfy the *instance* **then**
16: $satisfy \leftarrow FALSE$
17: **end if**
18: **end if**
19: **end if**
20: **end for**
21: **if** $satisfy = TRUE$ **then**
22: *count* $++$
23: **end if**
24: **end for**
25: $fitness \leftarrow count/instances\_number$
26: **else**
27: $fitness \leftarrow 0$
28: **end if**

match, then the antecedent and consequent of an individual are not disjoint itemsets and the individual is discarded. Following this, the aptitude of the individual is measured. In this proposal, we determine that the higher the fitness value, the better the rule discovered. In order to discard an individual, a zero fitness value is assigned.

The evaluation process of each individual is performed by obtaining the *fitness* function value. It will be the support, which was properly defined in Section 2.1. Another heuristic that we will use is the confidence of the rule (see Equation 2.1 in Section 2.1).

3.1.6 Algorithm

G3PARM follows a classical generational scheme. It uses two populations with a fixed size, one for the individuals obtained throughout generations and the other one (the auxiliary population) for the individuals that exceed a certain minimum quality threshold. This auxiliary population is updated in each generation keeping the best individuals throughout the evolutionary process. The pseudo-code of the algorithm is shown in Algorithm 4.

The algorithm starts producing the population by randomly generating individuals from the CFG defined in Figure 3.1 and not exceeding the maximum number of derivations. In the initial generation, the auxiliary population is empty. Then, a set of individuals is selected via a binary tournament from the merging of the current population and the auxiliary population. This selector works by selecting two individuals at random from the current population and after comparing them, it keeps the best of both. Individuals are selected to act as parents with a certain probability of crossover and a new set of individuals is obtained. The next step is to perform the mutation of the resulting set of individuals obtained in the previous crossover step, with a certain probability as the crossover operator. Once we have obtained the new population by crossover and mutation, the auxiliary population is updated by merging the previous auxiliary population and the current population. Then, individuals are ranked according to their support. In a filtering step, those individuals that are equal are eliminated. The G3PARM algorithm considers that

Algorithm 4 G3PARM algorithm

Require: $max\_generations, N$
Ensure: A
1: $P \leftarrow$generateIndividuals(N)
2: $A \leftarrow \emptyset$
3: $C \leftarrow \emptyset$
4: $M \leftarrow \emptyset$
5: **while** $num\_generations < max\_generations$ **do**
6: $P \leftarrow$ Select parents ($P \cup A$)
7: $C \leftarrow$ Crossover (P)
8: $M \leftarrow$ Mutation (C)
9: Evaluate (M)
10: $P \leftarrow M$
11: $C \leftarrow \emptyset$
12: $M \leftarrow \emptyset$
13: $A \leftarrow$ Update auxiliary population ($A \cup P$, $maxAuxPopSize$)
14: $num\_generations++$
15: **end while**
16: **return** A

procedure Update auxiliary population
Require: A, $maxAuxPopulationSize$
Ensure: A'
1: $A' \leftarrow \emptyset$
2: $A \leftarrow$ Order (A)
3: $i \leftarrow 0$
4: **for all** $individuals$ in A **do**
5: **if** $individual_i^A$ is not in A' **then**
6: $A' \leftarrow (A' \cup individual_i^A)$
7: **else**
8: $j \leftarrow 0$
9: $fitness_A \leftarrow$ getFitness($individual_i^A$)
10: **for all** $individuals$ in A' **do**
11: $fitness_{A'} \leftarrow$ getFitness($individual_j^{A'}$)
12: **if** $fitness_A$ is equal to $fitness_{A'}$ **then**
13: **if** $individual_i^A$ is more restrictive than $individual_j^{A'}$ **then**
14: $A' \leftarrow$ eliminate($individual_j^{A'}$)
15: $A' \leftarrow (A' \cup individual_i^A)$
16: break;
17: **end if**
18: **else**
19: **if** $fitness_A$ is greater than $fitness_{A'}$ **then**
20: $A' \leftarrow$ eliminate($individual_j^{A'}$)
21: $A' \leftarrow (A' \cup individual_i^A)$
22: break;
23: **end if**
24: **end if**
25: $j++$
26: **end for**
27: **end if**
28: $i++$
29: **end for**
30: $A' \leftarrow$ getIndividuals(A',$maxAuxPopulationSize$)
31: **return** A'
end procedure

Table 3.2: Sample execution of G3PARM over the Automobile dataset

Rule	Support	Confidence
IF (origin <= 2 AND mpg < 39.08) **THEN** (cylinders >= 4)	0.7857	1.0000
IF (displacement <= 300.20 AND weight <= 4433) **THEN** (horsepower <= 190)	0.7500	1.0000
IF (weight < 3728) **THEN** (horsepower <= 190)	0.7755	0.9967
IF (model_year <= 78) **THEN** (mpg < 39.08)	0.7041	0.9928

two individuals are equal if, despite having different genotypes, they are composed of the same attributes.

For example, rules ($Condition1$ AND $Condition2$)$\rightarrow$ $Condition3$ and ($Condition2$ AND $Condition1$)$\rightarrow$ $Condition3$ are considered the same. If two individuals have the same attributes, the one with the lowest fitness is removed. If these individuals have the same attributes and the same fitness, the the least restrictive one is removed, e.g., the rule $A \geq 3 \rightarrow B < 4$ is removed because the rule $A > 3 \rightarrow B < 4$ is more restrictive but both have the same fitness. Comparing two conditions, it is considered that the most restrictive one represents the smallest number of values. Using the rules mentioned above, an attribute A having the $[A_{initial}, A_{final}]$ range of values, the condition $A \geq 3$ represents the interval $[3, A_{final}]$, whereas the condition $A > 3$ states for $(3, A_{final}]$, so the latter represents the smallest interval and, in consequence, it is the most restrictive. From the resulting set, the individuals that exceed a certain threshold of support and confidence are selected. The algorithm terminates once it reaches a certain number of generations, returning auxiliary population individuals.

Finally, some sample ARs obtained by executing G3PARM over the Automobile Performance and Zoo datasets[2] are illustrated in Tables 3.2 and 3.3. As shown, using G3PARM it is possible to obtain either numerical or categorical rules having a highest confidence value. Additionally, the rules mined are interesting since their lift value is greater than the unity. A further experimental study is carried out in the following section.

[2]These datasets are publicly available for download from the UCI (University of California, Irvine) machine learning repository (http://archive.ics.uci.edu/ml/datasets/).

Table 3.3: Sample execution of G3PARM over the Zoo dataset

Rule	Support	Confidence
IF (backbone != f) **THEN** (legs != 8)	0.8218	1.0000
IF (venomous = f) **THEN** (legs != 8)	0.9109	0.9892
IF (airborne = f) **THEN** (venomous = f)	0.7029	0.9221
IF (fins != t) **THEN** (venomous = f)	0.7623	0.9166

3.2 Experimental Study

Several experiments have been carried out in order to compare our proposal to some exhaustive search algorithms for ARM like Apriori[3] [6], FP-Growth[4] [7] and GAs such as ARMGA [8, 73] and QuantMiner [11] using six real-world datasets. All the experiments were performed on an Intel Core i7 machine with 12GB main memory, running CentOS 5.4 and were written in Java. The G3PARM algorithm was written by using JCLEC[5] [74], a Java library for evolutionary computation.

In the following subsections, first the different real-world datasets and the experimental set-up are described. The results obtained in different experiments are shown next and, finally, we present an analysis of the scalability of the algorithms.

3.2.1 Datasets

In order to analyse the performance of our proposal, a set of executions were performed using the following datasets, which are summarized in Table 3.4.

- *House_16H.* This concerns a study on the prediction of average house prices of houses in a region by considering both the demographic composition and the state of the housing market. For the purpose of this dataset, only a level State-Place was used and data from all states were obtained. This dataset contains 22,784 transactions and 17 continuous attributes.

[3]Weka machine learning project. http://www.cs.waikato.ac.nz/~ml/index.html.

[4]F. Coenen (2003), The LUCS-KDD FP-growth Association Rule Mining Algorithm, Department of Computer Science, University of Liverpool, UK. http://www.cxc.liv.ac.uk/~frans/KDD/Software/FPgrowth/fpGrowth.html.

[5]JCLEC software and documentation is available for download from http://jclec.sf.net

Table 3.4: Datasets and their main characteristics

Dataset	**Abbreviation**	**#Inst.**	**#Attr.**	**Attr. Type**
House 16	HH	22784	17	Num.
CPU activity	CPU	8192	22	Num.
Wisconsin Breast Cancer	W	683	11	Categ., Num.
Wisconsin Diagnostic Breast Cancer	$WDBC$	569	32	Categ., Num.
Automobile Performance	MPG	392	8	Num.
German Credit	Cr	1000	21	Categ., Num.

- *Cpu_act*. The computer activity dataset is a collection of computer system activity measures. The data was collected from a Sun Sparc-station 20/712 with 128 MB of memory running in a multi-user environment in a university department. This dataset contains 8,192 transactions and 22 continuous attributes.

- *Wisconsin*. This breast cancer dataset was obtained from the University of Wisconsin Hospitals, Madison from Dr William H. Wolberg. These are the patients seen by Dr. Wolberg since 1984. This dataset contains 683 transactions and 11 attributes.

- *Wisconsin Diagnostic*. This dataset was obtained from a digitized image of a fine needle aspirate of a breast mass. It describes characteristics of the cell nuclei present in the image. This dataset contains 569 transactions and 32 attributes.

- *Automobile Performance*. This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American Statistical Association Exposition. This dataset contains 398 transactions and 9 attributes.

- *Credit*. For business credit, banks are interested in knowing whether prospective consumers will pay back their loan or not. This dataset contains 1000 transactions, 7 integer attributes and 14 categorical attributes.

3.2.2 Experimental Set-up

In order to obtain the optimal parameters (see Table 3.5) that allow us to reach the best results using the evolutionary proposals, a series of experiments were carried out. For the G3PARM algorithm, the best results were obtained with a population size of 50 individuals, 100 generations, 70% crossover probability, 14% mutation probability, a maximum derivation number of 24, an external population of size 20, a 90% external confidence threshold and a 70% external support threshold. Because most of the algorithms for mining ARs have only one attribute in the consequent, we restrict the G3PARM algorithm in order to obtain rules with only one consequent.

For the ARMGA algorithm, the optimal parameters were: a population size of 50 individuals, 100 generations, 100% selection probability, a 90% crossover probability, 1% mutation probability, a maximum rule length of 4 attributes and a 90% confidence threshold. ARMGA does not use a support threshold.

Finally, the optimal parameters of the QuantMiner algorithm were: a population size of 250 individuals, 100 generations, 40% mutation probability and 50% crossover probability. Like G3PARM and ARMGA, this algorithm uses a 90% confidence threshold and a 70% support threshold. For the sake of a fair comparison with G3PARM, QuantMiner has been configured to return 20 rules at most, those with the best fitness from the final population.

Table 3.5: Parameters established for each algorithm

Dataset	**Apriori**	**FP-Growth**	**QuantMiner**	**ARMGA**	**G3PARM**
Pop. size	-	-	250	50	50
Pool size	-	-	20	-	20
# Generations	-	-	100	100	100
Max. derivations	-	-	-	-	24
Crossover prob.	-	-	0.50	0.90	0.70
Mutation prob.	-	-	0.40	0.01	0.14
Max. lenght	-	-	-	4	-
Min. support	0.70	0.70	0.70	-	0.70
Min. confidence	0.90	0.90	0.90	0.90	0.90

For the Apriori and FP-Growth algorithms, the same support threshold is used as in G3PARM and QuantMiner (70%), and the same confidence threshold as in the aforementioned evolutionary algorithms (90%).

The results obtained by each evolutionary algorithm are the average results obtained running each one ten times using different seeds each time for the EAs.

3.2.3 Analysis of Nominal Datasets

As mentioned above, algorithms like Apriori or FP-Growth are based on categorical attributes, so numerical data must be previously discretized. Equal-width and equal-frequency methods are two well-known unsupervised discretization methods [75]. Equal-frequency discretization requires some previous information about the data distribution, so in this analysis, only the equal-width method is used. It determines the minimum and maximum values of each attribute and then divides the range into a number of discrete equal width intervals.

Several experiments were carried out with 4, 5, 6, 7 and 8 intervals per numerical attribute. Tables 3.6 and 3.7 show the results obtained by four algorithms (Apriori, FP-Growth, ARMGA and G3PARM) with these intervals, where $Average\_supp$ is the average support of the rule set; $Average\_conf$ refers to the average confidence of the rule set; $\%Instances$ states the percentage of instances covered by the rules in all the instances in the dataset (expressed on a per unit basis); $Average\_Nrule$ is the average number of rules in the rule set; $HH\text{-}N$ is the *House_16H* dataset discretized in N intervals; $CPU\text{-}N$ is the *Cpu_act* dataset discretized in N intervals; $W\text{-}N$ is the *Wisconsin* dataset discretized in N intervals; and $Cr\text{-}N$ is the *Credit* dataset discretized in N intervals.

Analyzing the results presented in Tables 3.6 and 3.7 (the best results for each measure are set in bold typeface), notice that the G3PARM algorithm obtains rules with better support than the others; and better confidence than Apriori and FP-Growth. The use of the operator '*!=*' allows higher support to be obtained in datasets with infrequent attributes as mentioned in Section 3.1.2. The ARMGA algorithm obtains rules with high confidence since it tries to maximize the relative confidence regardless of the support. Notice that the relative confidence was defined in [8] as shown in Equation 3.1.

Table 3.6: Results obtained by the algorithms by discretizing the datasets

	Average *support*				Average *confidence*			
Dataset	**Apriori**	**FP-Gr.**	**ARMGA**	**G3PARM**	**Apriori**	**FP-Gr.**	**ARMGA**	**G3PARM**
$HH4$	0.769	0.769	0.025	**0.983**	0.943	0.943	**1.000**	0.999
$HH5$	0.765	0.765	0.014	**0.983**	0.955	0.955	**1.000**	0.999
$HH6$	0.781	0.781	0.015	**0.992**	0.962	0.962	**1.000**	0.999
$HH7$	0.792	0.792	0.025	**0.993**	0.960	0.960	**1.000**	0.999
$HH8$	0.770	0.770	0.036	**0.999**	0.949	0.949	**1.000**	**1.000**
$CPU4$	0.850	0.904	0.694	**0.999**	0.945	0.959	**1.000**	**1.000**
$CPU5$	0.855	0.870	0.823	**0.999**	0.940	0.951	**1.000**	**1.000**
$CPU6$	0.831	0.830	0.577	**0.999**	0.963	0.945	**1.000**	**1.000**
$CPU7$	0.755	0.787	0.539	**0.999**	0.973	0.943	**1.000**	**1.000**
$CPU8$	0.751	0.751	0.397	**0.999**	0.945	0.945	**1.000**	**1.000**
$W4$	0.744	0.744	0.030	**0.908**	0.980	0.980	**1.000**	0.998
$W5$	0.872	0.872	0.029	**0.926**	0.996	0.996	**1.000**	0.991
$W6$	0.872	0.872	0.024	**0.951**	0.996	0.996	**1.000**	0.999
$W7$	0.872	0.872	0.022	**0.961**	0.996	0.996	**1.000**	0.999
$W8$	0.872	0.872	0.018	**0.970**	0.996	0.996	**1.000**	0.999
$Cr4$	0.765	0.765	0.024	**0.944**	0.939	0.939	**1.000**	0.999
$Cr5$	0.773	0.773	0.020	**0.986**	0.942	0.942	**1.000**	**1.000**
$Cr6$	0.780	0.780	0.023	**0.995**	0.941	0.941	**1.000**	**1.000**
$Cr7$	0.780	0.780	0.018	**0.997**	0.941	0.941	**1.000**	**1.000**
$Cr8$	0.780	0.780	0.014	**0.997**	0.941	0.941	**1.000**	**1.000**
Ranking	2.500	2.500	4.000	**1.000**	3.475	3.425	**1.250**	1.850

$$relative\_confidence(A \cup C) = \frac{support(A \cup C) - (support(A) \times support(C)}{support(A) \times (1 - support(C))} \tag{3.1}$$

The G3PARM algorithm gets rules with confidence close to that obtained. For example, notice that using *Cpu_act* and *Credit* datasets, G3PARM gets rules with maximum confidence. Furthermore, G3PARM obtains rules that cover 100% of the dataset instances with few rules (a maximum of 20 given by the maximum auxiliary population size). Only using the *Wisconsin* dataset discretized in 8 intervals, G3PARM obtains a set of rules without completing the auxiliary population. Using this dataset, G3PARM obtains an average number of rules of 19.8. It should be noted that the results that are shown are the average results obtained running each algorithm ten times using different seeds each time. Moreover, it should be noted

Table 3.7: Results obtained by the algorithms by discretizing the datasets

	% Instances covered (on a per unit basis)				Average number of rules			
Dataset	**Apriori**	**FP-Gr.**	**ARMGA**	**G3PARM**	**Apriori**	**FP-Gr.**	**ARMGA**	**G3PARM**
*HH*4	**1.000**	**1.000**	0.492	**1.000**	1.34E5	1.34E5	50.0	20.0
*HH*5	**1.000**	**1.000**	0.412	**1.000**	1.92E4	1.92E4	50.0	20.0
*HH*6	**1.000**	**1.000**	0.390	**1.000**	7.05E3	7.05E3	50.0	20.0
*HH*7	**1.000**	**1.000**	0.494	**1.000**	1.86E3	1.86E3	50.0	20.0
*HH*8	**1.000**	**1.000**	0.555	**1.000**	3.11E3	3.11E3	50.0	20.0
*CPU*4	**1.000**	**1.000**	0.803	**1.000**	2.00E6	2.00E6	50.0	20.0
*CPU*5	**1.000**	**1.000**	0.935	**1.000**	2.00E6	2.00E6	50.0	20.0
*CPU*6	**1.000**	**1.000**	0.616	**1.000**	2.00E6	2.00E6	50.0	20.0
*CPU*7	**1.000**	**1.000**	0.639	**1.000**	2.00E6	2.00E6	50.0	20.0
*CPU*8	**1.000**	**1.000**	0.549	**1.000**	1.77E6	1.77E6	50.0	20.0
*W*4	0.954	0.954	0.362	**1.000**	17.0	17.0	50.0	20.0
*W*5	0.872	0.872	0.488	**1.000**	1.0	1.0	50.0	20.0
*W*6	0.872	0.872	0.369	**1.000**	1.0	1.0	50.0	20.0
*W*7	0.872	0.872	0.356	**1.000**	1.0	1.0	50.0	20.0
*W*8	0.872	0.872	0.335	**1.000**	1.0	1.0	50.0	19.8
*Cr*4	0.989	0.989	0.399	**1.000**	14.0	14.0	50.0	20.0
*Cr*5	0.989	0.989	0.335	**1.000**	11.0	11.0	50.0	20.0
*Cr*6	0.987	0.987	0.378	**1.000**	10.0	10.0	50.0	20.0
*Cr*7	0.987	0.987	0.305	**1.000**	10.0	10.0	50.0	20.0
*Cr*8	0.987	0.987	0.279	**1.000**	10.0	10.0	50.0	20.0
Ranking	2.250	2.250	4.000	**1.500**	-	-	-	-

that ARMGA returns, from the regular population, those rules that satisfy a 90% confidence threshold (not requiring any support threshold). In every dataset, this algorithm returns the complete regular population (i.e., 50 individuals previously established in the configuration parameter stage) because all of them satisfy the confidence threshold. Finally, notice that ARMGA does not make a comparison between rules to determine the most restrictive one as G3PARM does. As shown in Table 3.7, in the *Wisconsin* and *Credit* datasets, Apriori and FP-Growth get fewer rules than the others because there are few rules that exceed the support threshold. However, with the same support threshold, the G3PARM algorithm obtains more rules by using the operator '*!=*'. The ARMGA algorithm does not have any support threshold, so it gets rules with a very low support. Only in the *Cpu_act* dataset does the ARMGA algorithm get rules with high support. This is because this dataset has a lot of frequent attributes.

In order to analyse the results obtained, a series of statistical tests [76, 77] were carried out. The Friedman test is used to compare the results obtained and to be able to precisely analyse whether there are significant differences among the three algorithms. This test first ranks the j-th of k algorithms on the i-th of N datasets, and then calculates the average rank according to the F-distribution (F_F) throughout all the datasets (see average rankings in Tables 3.6 and 3.7), and calculates the Friedman statistics. If the Friedman test rejects the null-hypothesis indicating that there are significative differences, then a Bonferroni–Dunn test is performed to reveal these differences. The performance of G3PARM is evaluated by comparing it to the other algorithms in terms of their average support, average confidence and the percentage of instances covered by each algorithm.

The Friedman average ranking statistics for average support measures distributed according to F_F with $k-1$ and $(k-1)(N-1)$ degrees of freedom is 171.000; 59.593 for an average confidence measure; and 39.461 for a coverage (percentage of instances covered) measure. None of them belong to the critical interval $[0, (F_F)_{0.05,3,57} = 2.766]$. Thus, we reject the null-hypothesis that all algorithms perform equally well for these three measures. In order to analyse if there are significant differences among the three algorithms, the Bonferroni–Dunn test is used to reveal the difference in performance, 1.198 being the critical difference (CD) value for $p = 0.01$.

The results indicate that for the support measure (see Figure 3.5), at a significance level of $p = 0.01$ (i.e., with a probability of 99%), there are significant differences

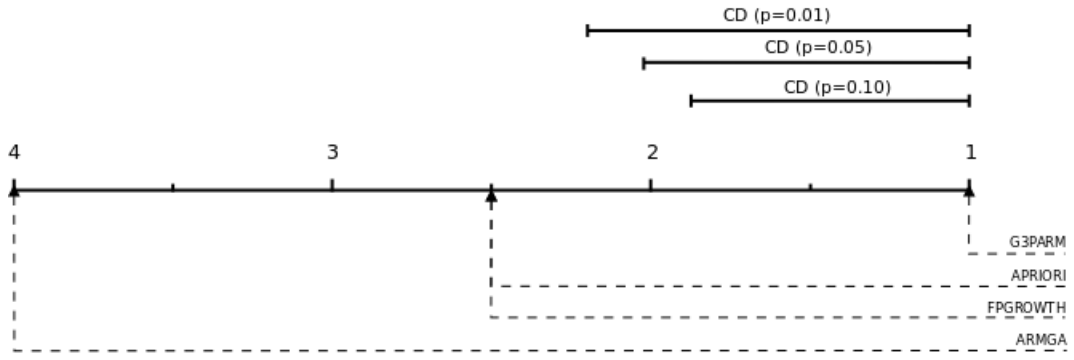


Figure 3.5: Critical difference obtained with the Bonferroni-Dunn test for the support measure

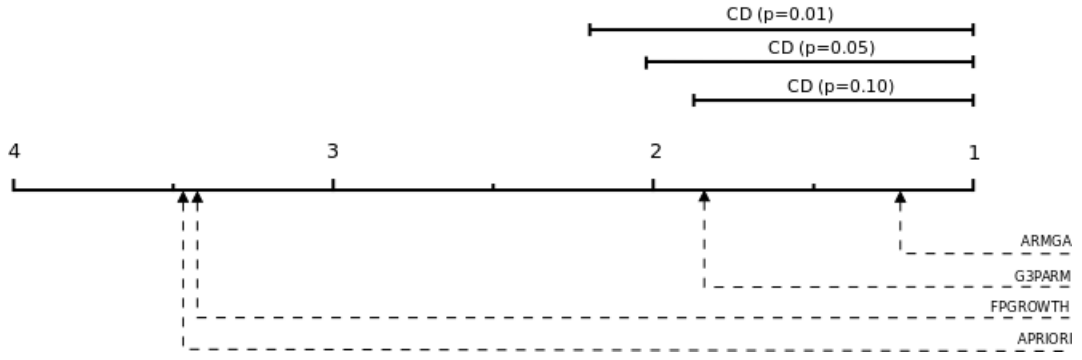


Figure 3.6: Critical difference obtained with the Bonferroni-Dunn test for the confidence measure

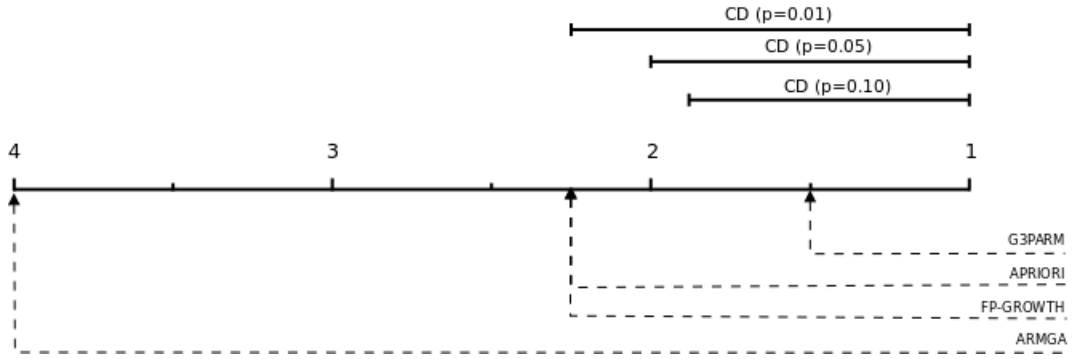


Figure 3.7: Critical difference obtained with the Bonferroni-Dunn test for the coverage measure

between G3PARM and the other algorithms, the performance of G3PARM being statistically better. Focusing on the confidence measure (see Figure 3.6), at a significance level of $p = 0.01$, there are significant differences between G3PARM and the Apriori and FP-Growth algorithms. On the other hand, there are no significant differences between G3PARM and ARMGA for the confidence measure. Finally, if we focus on the coverage measure (see Figure 3.7), at a significance level of $p = 0.01$ there are significant differences between G3PARM and ARMGA. Concerning exhaustive search algorithms for ARM (Apriori and FP-Growth), G3PARM remains as the control algorithm and is also competitive, obtaining the best ranking.

3.2.4 Analysis of Numerical Datasets

As mentioned above, an advantage of our proposal is that it provides understandable rules and allows a context-free grammar to be adapted and applied to each specific problem or domain and eliminates the problems raised by discretization. Since exhaustive search algorithms for ARM like FP-Growth or Apriori are based on categorical attributes, they cannot be used with the original dataset (if the dataset has numerical attributes). Therefore, we compare ARMGA, QuantMiner and G3PARM using real-world datasets without any previous pre-processing step to demonstrate that our proposal performs very well when it is applied directly to the original datasets. In this case, Table 3.8 shows the results obtained in terms of average support, average confidence, percentage of instances covered by the rules of total instances in the dataset using different algorithms (ARMGA, QuantMiner and G3PARM), and the average number of rules obtained. HH is the *House_16H* dataset; CPU is the *Cpu_act* dataset; W is the *Wisconsin* dataset; Cr is the *Credit* dataset; MPG is the *Automobile Performance* dataset; and $WDiag$ is the *Wisconsin Diagnostic* dataset.

Table 3.8: Results obtained by the algorithms with the original datasets

	Average *support*			**Average *confidence***		
Dataset	**ARMGA**	**QuantMiner**	**G3PARM**	**ARMGA**	**QuantMiner**	**G3PARM**
HH	0.009	0.712	**0.971**	**1.000**	0.970	0.999
CPU	0.823	0.707	**0.976**	**1.000**	0.958	0.999
W	0.069	0.753	**0.905**	**1.000**	0.966	0.996
Cr	0.034	0.882	**0.972**	**1.000**	0.963	0.995
MPG	0.020	0.729	**0.895**	**1.000**	0.977	0.997
$WDiag$	0.065	0.702	**0.986**	**1.000**	0.954	0.999
Ranking	2.833	2.166	**1.000**	**1.000**	3.000	2.000

	% Instances covered (on a per unit basis)			**Average number of rules**		
Dataset	**ARMGA**	**QuantMiner**	**G3PARM**	**ARMGA**	**QuantMiner**	**G3PARM**
HH	0.327	**0.999**	**0.999**	50.0	20.0	19.8
CPU	**1.000**	0.958	0.999	50.0	20.0	18.9
W	**1.000**	0.966	0.996	50.0	20.0	19.2
Cr	**1.000**	0.963	0.995	50.0	20.0	19.2
MPG	0.223	**1.000**	0.998	50.0	20.0	19.7
$WDiag$	0.525	0.986	**0.997**	50.0	20.0	19.4
Ranking	2.000	2.250	**1.750**	-	-	-

On analysing the results presented, note that the G3PARM algorithm obtains rules with better support than QuantMiner or the ARMGA algorithm. The ARMGA algorithm does not have a support threshold, so it gets rules with a very low support but a higher confidence. As previously commented, only in dataset *Cpu_act* does the ARMGA algorithm get rules with high support because this dataset has a lot of frequent attributes. Furthermore, in most cases, ARMGA obtains rules that cover 100% of the dataset instances while G3PARM obtains close results. Focusing on the number of rules mined, the three algorithms obtain a homogeneous set of rules. Only when applying G3PARM to the *Cpu_act* dataset does the resultant set of rules comprise less than 19 rules. Moreover, note that ARMGA returns the complete regular population (50 rules), because all of them satisfy the confidence threshold. It should be noted that this population size was previously established during the set-up. On the other hand, QuantMiner returns the best 20 rules from the final population, i.e., those with the highest fitness value. However, if this algorithm is not constrained to obtain the best rules, it may return a huge set of rules based on the number of individuals. Finally, neither ARMGA nor QuantMiner compare the rules to determine the most restrictive one as G3PARM does.

Then, the Friedman rank test [78] is used to analyse and compare the results obtained, and the Bonferroni–Dunn test is performed to reveal the differences that exist when the Friedman rank test rejects the null-hypothesis. By using numerical attrributes, G3PARM and the other algorithms are compared in terms of average support, average confidence and the percentage of instances covered. The average ranking for each algorithm is also shown in Table 3.8.

The Friedman average ranking statistic for average support measures distributed according to F_F with $k-1$ and $(k-1)(N-1)$ degrees of freedom is 31.000; ∞ for the average confidence measure; and 0.333 for the coverage measure. The Friedman rank test shows that for $k = 3$ algorithms and $N = 6$ datasets, the function has a value of 9.000 with alpha 0.01. Support and confidence measures have a higher value so we reject the null-hypothesis that all algorithms perform equally well for these two measures. The Bonferroni–Dunn test is used to reveal the difference in performance: the CD value is 1.293 with $p = 0.05$. Finally, when focusing on the coverage measure, there are no significant differences between the three algorithms.

The results indicate that for support measures (see Figure 3.8), at a significance level of $p = 0.05$ (i.e., with a probability of 95%), there are significant differences between G3PARM and the other algorithms, the performance of G3PARM being statistically better. Then, if focusing on the confidence measure (see Figure 3.9),

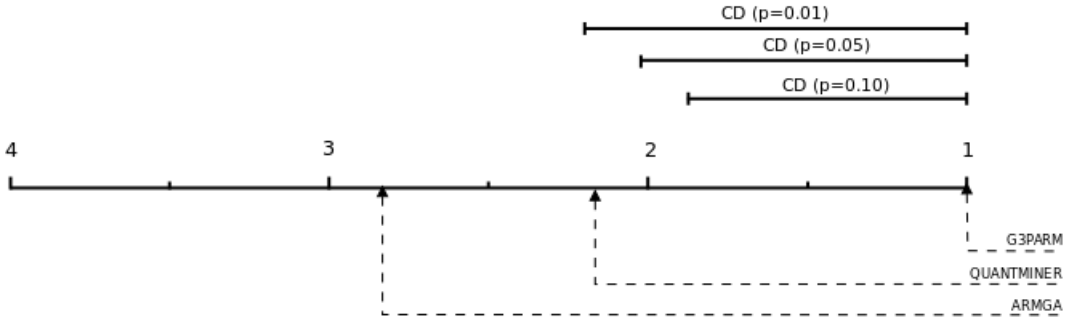


Figure 3.8: Critical difference obtained with the Bonferroni-Dunn test for the support measure

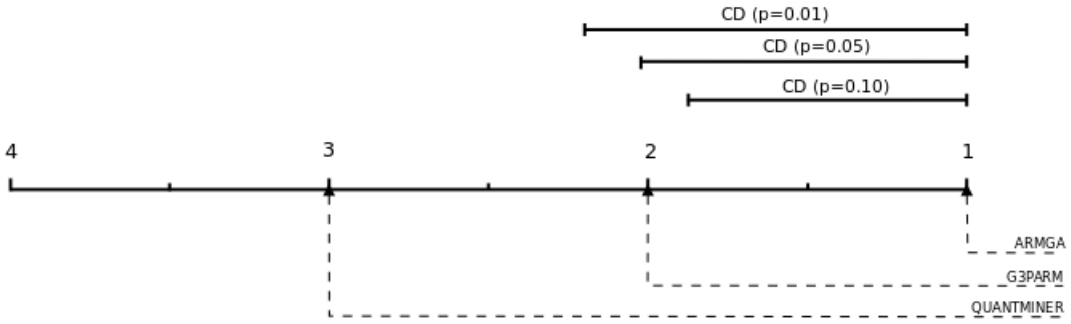


Figure 3.9: Critical difference obtained with the Bonferroni-Dunn test for the confidence measure

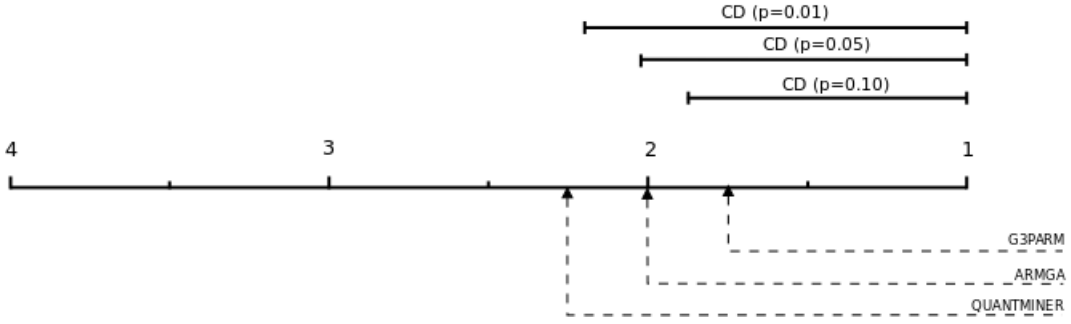


Figure 3.10: Critical difference obtained with the Bonferroni-Dunn test for the coverage measure

there are significant differences between QuantMiner and the ARMGA algorithm at a significance level of $p = 0.05$, ARMGA being statistically better than QuantMiner. On the other hand, ARMGA is better than the other algorithms although G3PARM obtains values close to ARMGA for the confidence measure. Finally, there are no significant differences for the coverage measure (see Figure 3.10).

3.2.5 Analysis of Scalability

A set of different experiments were also carried out to analyse the computation time of the algorithms using the *House_16H* dataset discretized in four intervals. Figure 3.11 shows the relation between the runtime and the number of instances. The Y axis represents time in milliseconds, whereas the X axis stands for the percentage of instances using all attributes. In the same way, Figure 3.12 shows the relation between the runtime and the number of attributes. The Y axis represents time in milliseconds and the X axis, the number of attributes using 100% of the instances.

Figure 3.11 clearly shows how the runtime of the exhaustive search algorithms for ARM increases as the size of the dataset increases compared to the runtime of the GAs. In other EA-based proposals, the execution time remains almost constant.

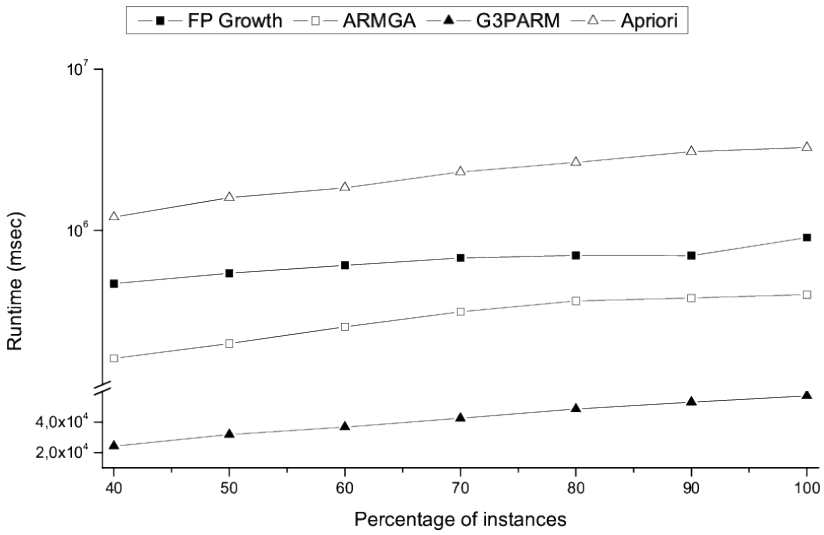


Figure 3.11: Relation between the runtime and the percentage of instances over the discretized *House_16H* dataset

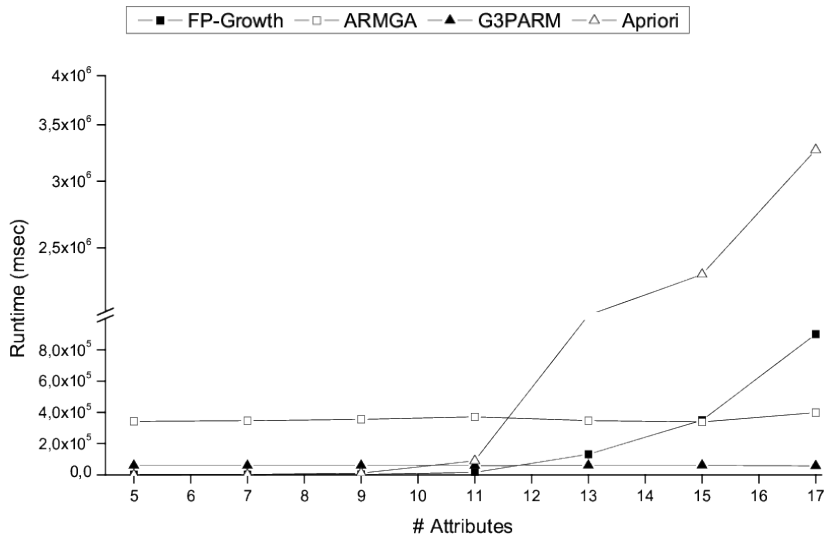


Figure 3.12: Relation between the runtime and the number of attributes over the discretized *House_16H* dataset

Figure 3.12 shows how the exhaustive search algorithms for ARM expend a large amount of time mining ARs when the number of attributes is high. By contrast, the results plotted in these figures show that GAs scale quite linearly for the dataset used in the experiment. G3PARM and ARMGA scale better than the larger number of attributes in comparison with FP-Growth and Apriori because the greater the number of attributes, the greater the combinatorial explosion caused to obtain frequent itemsets. GAs are not influenced by this combinatorial explosion as are the exhaustive search algorithms for ARM. The computation time in GAs increases with the number of evaluations for each individual independently of the number of attributes.

Several experiments were also carried out to analyse the computation time of the algorithms using the *House_16H* dataset without any pre-processing step. Like above mentioned figures, Figures 3.13 and 3.14 show the relation between the runtime and the number of instances and attributes. Examining these figures, we can see how the runtime scales quite linearly when using EAs over numerical attributes. The results plotted in this figure show that QuantMiner and the G3PARM algorithm scale better than ARMGA. However, EA proposals with numerical attributes have a computation time higher than those obtained with categorical attributes.

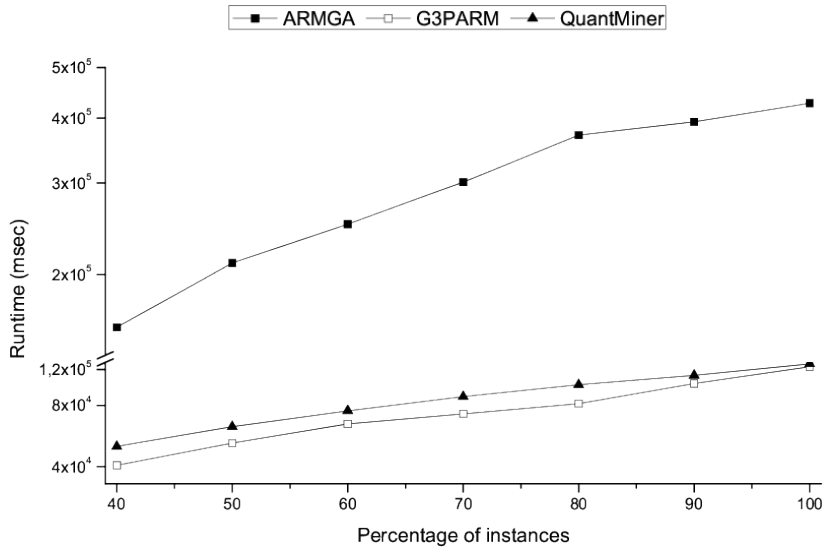


Figure 3.13: Relation between the runtime and the percentage of instances over the original *House_16H* dataset

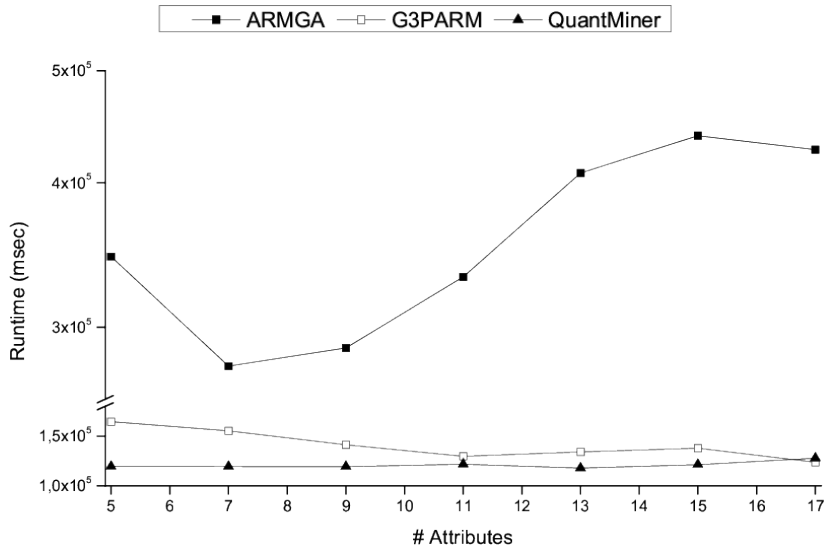


Figure 3.14: Relation between the runtime and the number of attributes over the original *House_16H* dataset

Concluding this analysis, G3PARM behaves quite linear when increasing both the number of attributes and instances. Regardless the type of dataset, that is, having numerical or discrete attributes, the proposed algorithm behaves quite similar. Only QuantMiner similarly behaves to G3PARM.

3.3 Conclusions

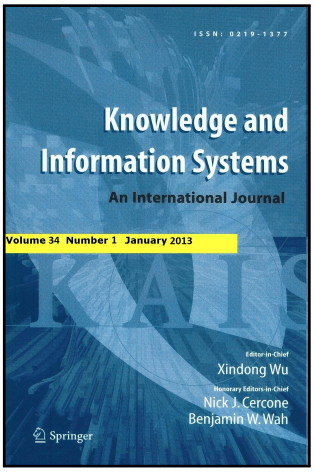
In this chapter, the first algorithm for mining ARs by using G3P has been described in depth. The use of G3P enables both categorical and numerical attributes to be defined without a previous preprocessing process. Additionally, the proposed algorithm, called G3PARM, extracts frequent and highly reliable ARs in only one step, not requiring a previous mining of the frequent pattern. The experimental analysis have demonstrated its high performance when increasing the number of both attributes and instances.

3.4 Publications Associated with this Chapter

- **International Journal:**

  TITLE: *Design and Behaviour Study of a Grammar Guided Genetic Programming Algorithm for Mining Association Rules* [79]

  AUTHORS: J. M. LUNA, J. R. ROMERO AND S. VENTURA



  KNOWLEDGE AND INFORMATION SYSTEMS, VOL. 32, ISSUE 1 (2012), PP. 53-76

  RANKING:

    IMPACT FACTOR (JCR 2011): 2.225

    KNOWLEDGE AREA:

COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE: 21/111

COMPUTER SCIENCE, INFORMATION SYSTEMS: 18/135

- **International Conference:**

  J. M. LUNA, J. R. ROMERO AND S. VENTURA. *Analysis of the Effectiveness of G3PARM Algorithm.* PROCEEDINGS OF THE 5TH INTERNATIONAL CONFERENCE ON HYBRID ARTIFICIAL INTELLIGENCE SYSTEMS, HAIS 2010. PP. 27-34, ISBN: 978-3-642-13802-7 [80]

4

G3PARM for Mining Rare Association Rules

Motivated by the promising performance of G3PARM, a full study of its adaptability to RARM, named Rare-G3PARM, is presented in this chapter. Particularly, we study how effective different fitness functions are for ARs identification and separating them from noise, and how a new genetic operator performs in guiding the search process. In this proposal, the resulting set only comprises the best rules discovered along the execution, and the number of rules to be discovered tends to be the number previously specified by the data miner. In addition, this proposal considers the lift measure together with support and confidence to overcome the problems of most algorithms [6, 7, 37] and also G3PARM, where only the support–confidence framework is followed [15]. For the sake of analysing the effectiveness of Rare-G3PARM, we compare our proposal to some existing algorithms in the RARM field. Results show that this proposal obtains rare and reliable relations between patterns, avoiding the discovery of noisy ones, in an efficient way.

4.1 The Rare-G3PARM Algorithm

This section introduces the mechanisms employed for adjusting G3PARM for the extraction of RARs and the new features required to make this algorithm an interesting RARM approach are described in depth. In this work, support, confidence and lift measures avoid obtaining misleading rules, playing an important role in the mining process. Furthermore, a new genetic operator serves to guide the search process in an effective way. It should be noted that Rare-G3PARM uses the same CFG as G3PARM, which was properly represented in Figure 3.1. In order to make G3PARM suitable for finding RARs, the following sections describe its new features.

4.1.1 Evaluation

The process of evaluating ARs in a specific problem is not an elementary issue since ARM algorithms base their extraction process on the quality of the rules. Some objective measures for evaluating the interest of these rules have been proposed by different researchers [22]. Two of the most important and widely used measures in this field are support and confidence (see Section 2.1).

Despite the fact that most proposals in ARM are based on a support–confidence framework, e.g. G3PARM, these measures are not sufficient to select interesting associations between patterns [15] in some application domains. For example, let us assume that 75% of the customers in a market bought onions (i.e., *customer* $\rightarrow$ *onions* with a confidence value of 75%), and 70% of the customers that bought tomatoes also bought onions (i.e., *tomatoes* $\rightarrow$ *onions* with a confidence value of 70%). In this situation, the fact of buying *tomatoes* does not provide an increment on buying *onions* so the rule is misleading. In other words, the occurrence of the antecedent does not imply an increment in the occurrence of the consequent. The measure 'lift' was defined to solve this problem (see Equation 2.2 in Section 2.1). It establishes how many times the antecedent and the consequent occur together more often than would be expected if they were statistically independent. An association rule is interesting if its confidence is higher than the support of its consequent. On the other hand, if the confidence of the rule is

equal to the support of its consequent, then both antecedent and consequent are independent.

The lift measure verifies three properties proposed by *Piatetsky-Shapiro* [20] and described in Section 2.1.2. Furthermore, the lift measure is symmetric (i.e., $lift(A \rightarrow C) = lift(C \rightarrow A)$). Sometimes, association rules require to measure the strength of implication in both directions, not only the degree of dependence. Nevertheless, lift is not free of problems either. The main drawback of this measure is that its range is not bounded [15], i.e., its domain is $[0, \infty)$, and it is not easy to compare the values of several rules because differences between them are not meaningful. However, in RARM, the necessity of discovering interesting rules makes the use of the lift measure specially appropriate in order to weight up the real interest of each rule mined. The goal of RARM is to obtain interesting rules with low support and which comprise patterns that are associated together.

In some application domains, the existing RARM approaches discover rules that comprise extremely infrequent patterns, so these rules actually cause noisy association rules instead of rare ones. In such situations, it is essential to correctly establish a boundary that allows of separating rare from noisy rules. But not only the distinction between rare and noisy rules can be complex. In some situations, depending on the application domain, it is really difficult to properly distinguish between frequent and rare rules. All these situations should be taken into account to find a solution to these problems, so it is necessary to be especially careful in the choice of a proper evaluation function that guides the search process. Therefore, in subsequent sections, a number of fitness functions, conceived especially for RARM, will be described in depth.

4.1.2 Genetic Operator

In any EA, genetic operators play an important role in the evolutionary process. These operators allow to maintain the genetic diversity along the search for the optimal solutions. In RARM, these operators are especially important as they serve to guide the search process, and to mine rules having lower support values than the original one. In this proposal, a new genetic operator, which modifies the highest support condition of a rule to obtain a new condition having a lower

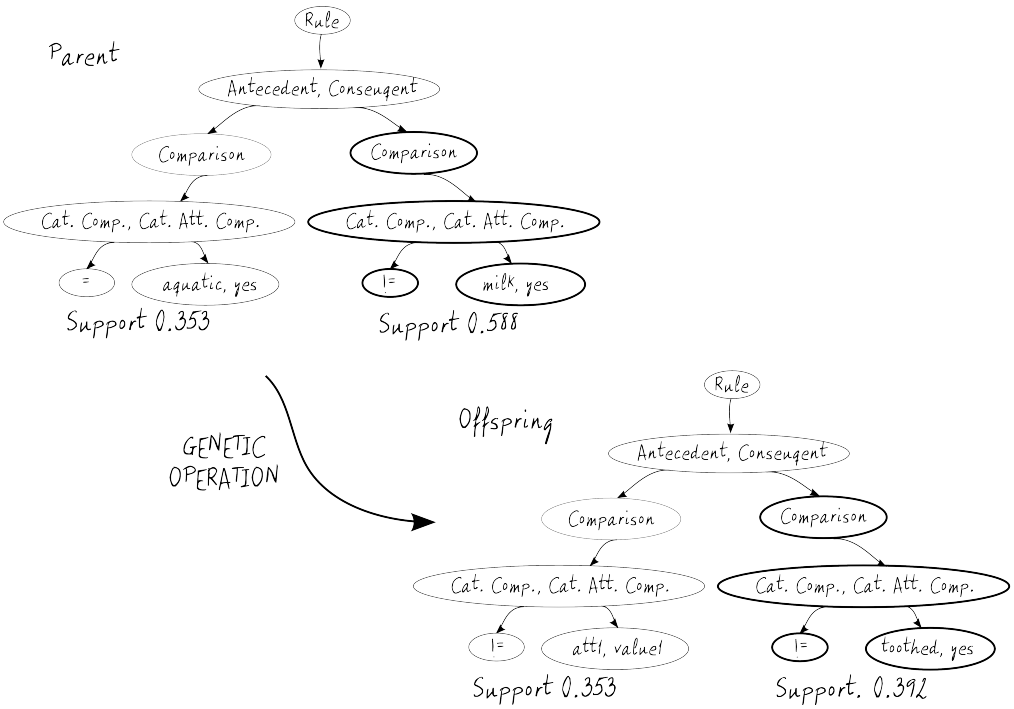


Figure 4.1: Example of the genetic operation

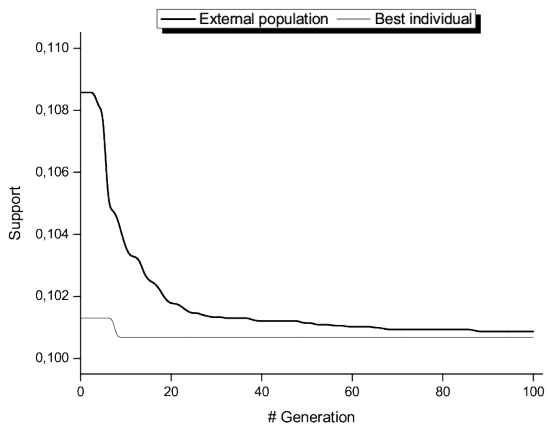
support value, has been implemented. Notice that the lower the support value of the conditions, the lower the support value of the entire rule.

With this purpose, the operator mutates the condition with the highest frequency of occurrence to obtain a new individual with a lower support (see Figure 4.1 for a real example[1]). The genetic operator provides two possibilities of changing a condition: (1) to obtain a new complete condition or (2) to obtain a new value for a terminal symbol.

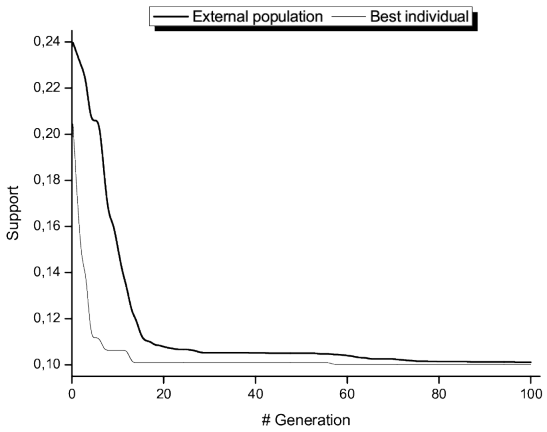
To study the convergence of the genetic operator, a number of different experiments were carried out using diverse datasets[2]. Figure 4.2 depicts the average support value of both the elite population (also known as auxiliary or external population in this document) and the best individuals obtained for every generation. The Y-axis represents the support, whereas the X-axis stands for the number of generations. Regardless of the dataset used, notice that the lowest support value is obtained

[1]The dataset used in this example is a real dataset (http://archive.ics.uci.edu/ml/datasets/Zoo) that will be used later in the experimental study.

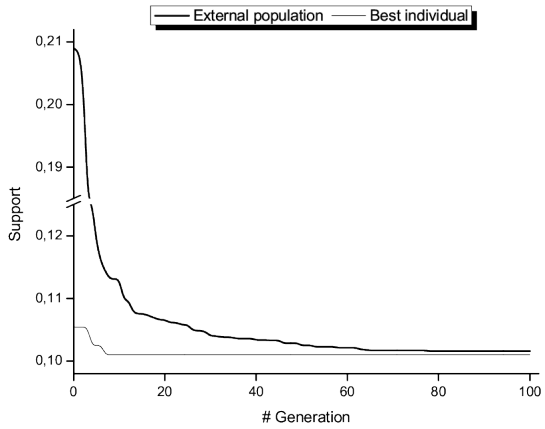
[2]Ankara weather, mushroom, soybean and vote datasets are available for download from the UCI machine learning repository (http://archive.ics.uci.edu/ml/datasets).



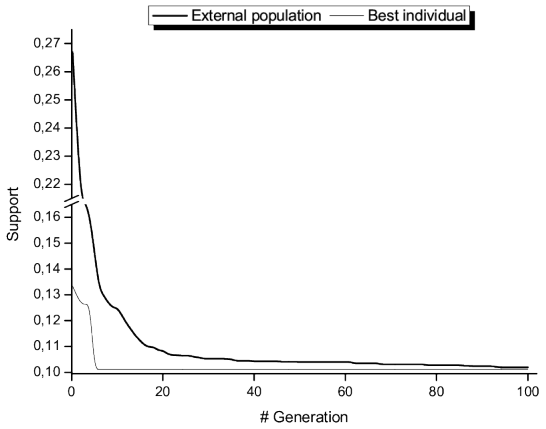
(a) Convergence of the genetic operator using the Ankara Weather dataset



(b) Convergence of the genetic operator using the Mushroom dataset



(c) Convergence of the genetic operator using the Soybean dataset



(d) Convergence of the genetic operator using the Vote dataset

Figure 4.2: Average auxiliary population support and best individual support obtained in each generation by using the genetic operator over a set of datasets

before the generation 60. Focusing on the best individual, it is discovered in early generations, i.e. between the generation 5 and 15, in most datasets.

Finally, an analysis of the behaviour of this operator is performed as its probability is increased. Figure 4.3 depicts the average support values calculated for the elite population using the four sample datasets. There, the Y-axis represents the average support, and the X-axis stands for the probability of the genetic operator. As shown, there is no significant difference among the average support values obtained using the different probabilities, the probability value of 0.80 approaching to the optimal one.

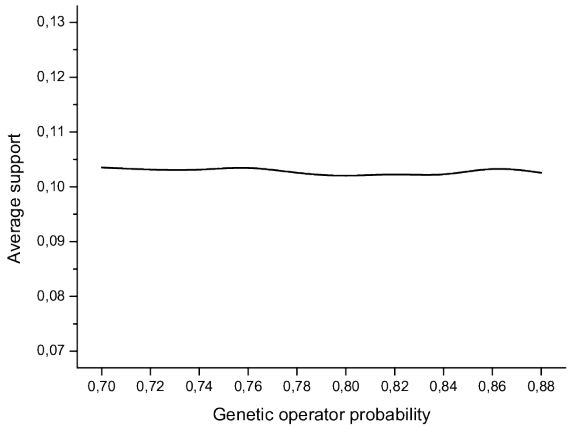


Figure 4.3: Average elite population support by using the genetic operator with different probabilities

4.1.3 Algorithm

The algorithm proposed (see Figure 4.4 for a general sketch and Algorithm 5 for the pseudocode) for the extraction of RARs follows a generational schema. Rare-G3PARM starts by generating a set of new individuals conformant to the specified grammar (see line 1 in the pseudocode). Each individual is encoded in a tree shape through the application of production rules. Notice that each terminal symbol adopts the name and value of any of the dataset attributes. Once a set of individuals is generated compiling the general population for the algorithm, the generational schema is executed. Several steps are performed for each generation: (1) in order to obtain new individuals, the algorithm selects individuals from the general population and the pool to act as parents (line 6) and a genetic operator is applied over them immediately afterwards with a certain probability (see line 7). Next, (2) these new individuals are evaluated (line 8). In the following step, (3) the elite population or pool – only for the first generation the elite population is empty–, which comprises the n most reliable individuals obtained along the evolutionary process, and the population are combined to form a new set. Then, this new set is ranked by their fitness, so only the best ones are selected until the new population is completed (see lines 9 and 10). Following, (4) the update procedure is carried out (line 13), ranking by confidence the new set of individuals (see line 2 in the procedure), this ranking serving to select the best n individuals from the new set for the updating process. Only those individuals having a fitness value greater

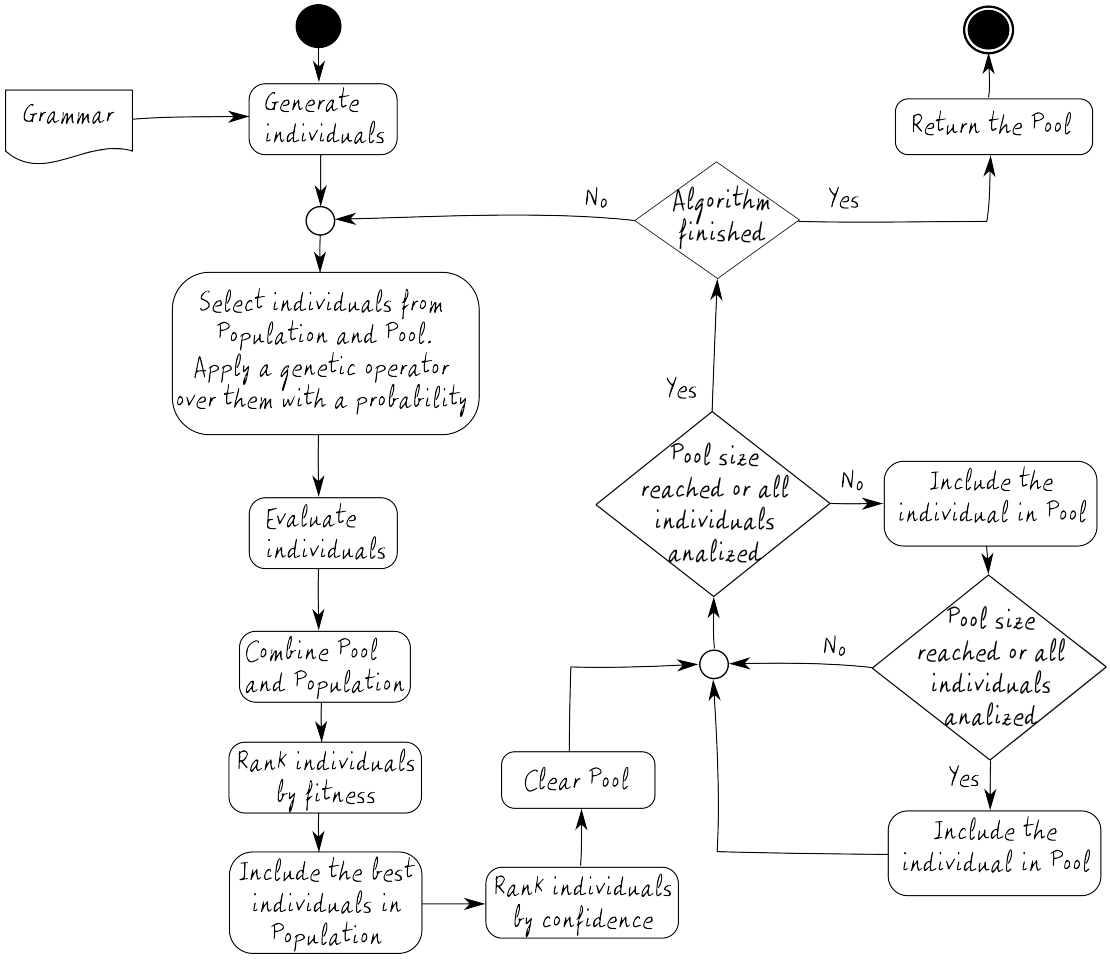


Figure 4.4: The flowchart for the Rare-G3PARM algorithm

than zero, a confidence value greater than the minimum confidence threshold, and a lift value greater than unity are considered prompting the discovery of infrequent, reliable and interesting association rules (see lines 6 to 8).

An important feature of Rare-G3PARM is the use of the lift measure, which represents the interest of a given AR. Traditionally, ARM proposals make use of a support-confidence framework, including G3PARM, attempting to discover rules whose support and confidence values are greater than certain thresholds. However, the mere fact of having rules that exceed these thresholds does not guarantee that the rules are interesting at all [15]. Even when the support and confidence thresholds are satisfied by a rule, this rule could be misleading if it acquires a confidence value less than its consequent support. In such a situation, the occurrence of the

Algorithm 5 Rare-G3PARM algorithm

Require: $max\_generations, num\_individuals, max\_Pool\_size, confidenceTreshold$
Ensure: A
1: $P \leftarrow$generateIndividuals($num\_individuals$)
2: $A \leftarrow \emptyset$
3: $M \leftarrow \emptyset$
4: $num\_generations \leftarrow 0$
5: **while** $num\_generations < max\_generations$ **do**
6: $P \leftarrow$ selectParents ($P \cup A$)
7: $M \leftarrow$ geneticOperator (P)
8: Evaluate (M)
9: $P \leftarrow$ rankIndividualFitness ($M \cup A$)
10: $P \leftarrow$ getBestIndividual (P, $num\_individuals$)
11: $M \leftarrow \emptyset$
12: $A \leftarrow \emptyset$
13: $A \leftarrow$ updateAuxiliaryPopulation (P, max_Pool_size, $confidenceTreshold$)
14: $num\_generations++$
15: **end while**
16: **return** A

procedure updateAuxiliaryPopulation
Require: A, max_Pool_size, $confidenceTreshold$
Ensure: A'
1: $A' \leftarrow \emptyset$
2: $A \leftarrow$ rankiIndividualConfidence (A)
3: $i \leftarrow 0$
4: **for all** $individuals \in A$ **do**
5: **if** $individual_i^A$ is not in A' **then**
6: **if** getFitness($individual_i^A$) > 0 **then**
7: **if** getConfidence($individual_i^A$) $> confidenceTreshold$ **then**
8: **if** getLift($individual_i^A$) > 1 **then**
9: $A' \leftarrow (A' \cup individual_i^A)$
10: **end if**
11: **end if**
12: **end if**
13: **end if**
14: $i++$
15: **if** getSize(A') $= max\_Pool\_size$ **then**
16: **return** A'
17: **end if**
18: **end for**
19: **return** A' **end procedure**

antecedent does not imply an increment in the occurrence of the consequent. For a better understanding, it is shown two sample rules obtained when running the

algorithm using the Zoo dataset[3]. The rule *(breathes = f) → (legs != 8)* is obtained as a RARs, having a support value of 0.198 and a confidence value of 0952. Despite the fact that this rule could be extracted as a reliable RAR, it is discarded since the lift value is 0.972, i.e., the occurrence of the antecedent does not imply and increment in the consequent. The same occur with the rule *(legs = 4) → (type != 3)*, having a support of 0.356, a confidence of 0.947, but a lift value of 0.996, so the rule is misleading and it is uninteresting.

Similarly to G3PARM, in Rare-G3PARM, the mining process only requires one step for obtaining association rules through the use of a grammar, not requiring a previous step for mining rare patterns. Other interesting feature of Rare-G3PARM is its ability for discarding rules having the same meaning. This is not an easy task because individuals having different genotypes may represent the same rule – two rules could represent the same semantic concepts although they represent different syntactic concepts. Therefore, individuals are compared based on their conditions and those with the same conditions are not kept in the pool. A simple example is shown using a real individual obtained from the Zoo dataset, e.g., the individual *(toothed != t AND legs != 5) → (airborne = f)*, which has a support value of 0.168. This individual, and this other individual *(legs != 5 AND toothed != t) → (airborne = f)* represent the same rule, so only one of them is kept in the pool. More complex examples can be produced, depending on the individuals generated, and are equally detected and removed.

Finally, some sample ARs obtained by executing Rare-G3PARM over the Automobile Performance[4] and Zoo datasets are illustrated in Tables 4.1 and 4.2. As shown, using Rare-G3PARM it is possible to obtain either numerical or categorical rules having a highest confidence value. Additionally, the rules mined are interesting since their lift value is greater than the unity. A further experimental study is carried out in the following section.

[3]The Zoo dataset comprises 102 instances and 17 categorical attributes, and it is publicly available for download from the UCI machine learning repository (http://archive.ics.uci.edu/ml/datasets/Zoo).

[4]The Automobile Performance dataset comprises 392 instances and 8 numerical attributes, and it is publicly available for download from the UCI machine learning repository (http://archive.ics.uci.edu/ml/datasets/Automobile).

Table 4.1: Sample execution of Rare-G3PARM over the Automobile Performance dataset.

Rule	Support	Confidence	Lift
IF (displacement >= 222.8 AND displacement <= 300.2 AND cylinders > 4) **THEN** (weight > 2318)	0.1531	1.0000	1.4359
IF (horsepower <= 118 AND displacement >= 222.8 AND cylinders > 4) **THEN** (weight > 2318)	0.1479	1.0000	1.4358
IF (displacement <= 300.2 AND displacement >= 222.8 AND horsepower > 82 AND origin < 2) **THEN** (cylinders > 4)	0.1454	1.0000	2.0741
IF (model_year > 72 AND horsepower >= 118) **THEN** (cylinders >= 6)	0.734	1.0000	2.1075
IF (origin > 2) **THEN** (acceleration <= 21.44)	0.2015	1.0000	1.0288

Table 4.2: Sample execution of Rare-G3PARM over the Zoo dataset.

Rule	Support	Confidence	Lift
IF (backbone != f AND aquatic = f AND toothed != t AND fins = f) **THEN** (milk != t)	0.1485	1.0000	1.6833
IF (hair = f AND tail != t) **THEN** (feathers != t)	0.1584	1.0000	1.2469
IF (tail != t AND toothed != t) **THEN** (backbone != t)	0.1683	1.0000	5.6111
IF (catsize = t AND predator = t AND toothed = t AND fins = f) **THEN** (airborne = f)	0.1485	1.0000	1.3117
IF (catsize = t AND predator = t AND fins = f AND hair != f) **THEN** (legs != 6)	0.1584	1.0000	1.1099

4.2 Experimental Study

Selecting a good fitness function is an important task in RARM, as previously mentioned. In this Section, four different fitness functions are described and an

Table 4.3: Datasets and their main characteristics

Dataset	Abbreviation	#Inst.	#Attr.	Attr. Type
Automobile Performance	*Autom*	392	8	Num.
Ankara Weather	*Ankara*	1608	10	Num.
German Credit	*Credit*	1000	21	Categ., Num.
Mushroom	*Mush*	8124	23	Categ.
Primary Tumour	*Prim*	339	18	Categ.
Soybean	*Soyb*	683	36	Categ.
Vote	*Vote*	435	17	Categ.
Wisconsin Breast Cancer	*WBC*	683	11	Categ., Num.
Wisconsin Diagnostic Breast Cancer	*WDatBC*	569	31	Categ., Num.
Wisconsin Prognostic Breast Cancer	*WPBC*	194	34	Categ., Num.
Zoo	*Zoo*	102	17	Categ.

analysis of their behaviour is carried out. Finally, a complete study of the effectiveness of the proposed algorithm compared to existing proposals in RARM, e.g., Apriori-Infrequent [25], Apriori-Inverse [19] and ARIMA [43], is accomplished.

4.2.1 Datasets

To analyse the performance of this proposal, a number of executions were performed over diverse datasets[5]; see Table 4.3), which provide different sizes and number of attributes. As mentioned above in previous sections, the existing RARM algorithms are brute-force algorithms, performing an exhaustive search process, so this kind of algorithm can hardly be executed with huge datasets composed of large number of attributes. However, our proposal does not have this limitation and it could be executed over any dataset with different sizes and number of attributes.

4.2.2 Experimental Set-up

Any evolutionary proposal should be configured with a set of adjustable parameters. All these parameters require previous study to determine those considered

[5]All these datasets are publicly available for download from the UCI machine learning repository (http://archive.ics.uci.edu/ml/datasets).

Table 4.4: Parameters established for each algorithm

Dataset	**Apriori-Inv.**	**ARIMA**	**MRG-Exp**	**Apriori-Inf.**	**Rare-G3PARM**
Pop. size	-	-	-	-	50
Pool size	-	-	-	-	20
# Generations	-	-	-	-	75
Max. derivations	-	-	-	-	24
Genetic prob.	-	-	-	-	0.80
Min. support	-	-	-	-	0.10
Max. support	0.40	0.40	0.40	0.40	0.40
Min. confidence	0.90	0.90	0.90	0.90	0.90

optimal, i.e., those that allow us to obtain the best global results. The final configuration was adopted after performing several tests using different rank values for each parameter, using several representative datasets, and then analysing which specific set-up globally yielded the best results. It is worth mentioning that no single combination of parameter values performed better for all datasets. Notice that these parameters are not particularized for each dataset. As is shown in Table 4.4, the best results for our approach were obtained with a population size of 50 individuals obtained using a CFG with a maximum derivation size of 24. An in depth analysis of the final configuration revealed that solutions are hardly improved after 75 generations—Figures 4.2(a), 4.2(b), 4.2(c) and 4.2(d) show some experiments carried out by using a subset of the whole experimental collection of datasets—so the evolutionary process should be carried out using this value for the number of generations. In this process, new individuals are obtained with a probability of 0.80 for the genetic operator, as previously discussed. The best individuals, i.e., those that exceed certain quality thresholds, are kept in the external population, whose size is prefixed and determining the maximum number of rules to be discovered by the algorithm. Notice that this pool size could be changed as the data miner's requirements vary. Anyhow, a pool size of 20 rules is established in this experiment. The remaining values of the quality thresholds are set as follows: 0.90 for the confidence, 0.10 and 0.40 for the minimum and maximum support thresholds, respectively, and a lift value greater than one. In order to carry out a fair comparison, the algorithms used in the experimental stage are executed using the

same confidence threshold (i.e., 0.90) and the same maximum support threshold of 0.40.

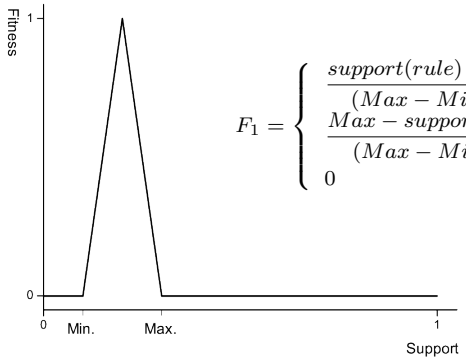
Notice that existing algorithms do not provide any minimum threshold, and exhaustive RARM algorithms only require two parameters: the support and confidence thresholds. This might be considered either a drawback or an advantage at the same time. On the one hand, requiring only two parameters to execute these algorithms seems to be an advantage for the final user, since no additional knowledge is required. On the other hand, this type of algorithms is normally used by data mining experts, who often require the ability to finely configure and tune the algorithm execution. Hence, only Rare-G3PARM provides the ability to fine-tune the length of the rules by setting the total number of rules to be mined and restricting the search space by means of the CFG.

All the experiments were performed on an Intel Core i7 machine with 12GB main memory and running CentOS 5.4. In addition, all the RARM proposals were written in Java. Other exhaustive RARM algorithms were obtained using RM-Tool [3], a data mining tool that accurately permits to perform the association rule mining task. Finally, similarly to G3PARM, the proposal presented in this work was coded using JCLEC [74].

4.2.3 Study of the Fitness Functions

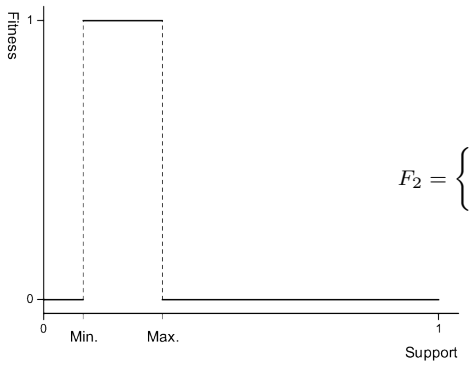
It is very important to properly differentiate between rare and noisy association rules in the process of mining them. In this section, four novel fitness functions that correctly separate rare from noisy rules are presented. To analyse the performance of these fitness functions, a number of experiments were carried out and then, the four fitness functions are compared based on confidence, lift, and the number of rules discovered. Notice that support values are analysed but not compared since the four fitness functions have different goals, also depending on the expert expectations.

The first fitness function (see Figure 4.5(a)) provides a maximum fitness value in the middle of a certain interval provided by a minimum and a maximum support value. The closer the support of a rule is to the interval limits, the lower its fitness value is. Out of this interval, a zero value is assigned. For a better understanding, it is



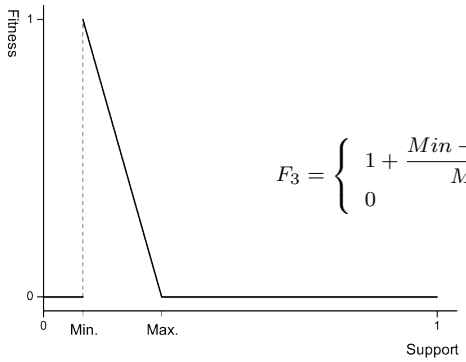
$$F_1 = \begin{cases} \dfrac{support(rule) - Min}{(Max - Min)/2} & \text{if } Min \leq support(rule) \leq (Max + Min)/2 \\ \dfrac{Max - support(rule)}{(Max - Min)/2} & \text{if } (Max + Min)/2 \leq support(rule) \leq Max \\ 0 & \text{otherwise} \end{cases}$$

(a) First fitness function



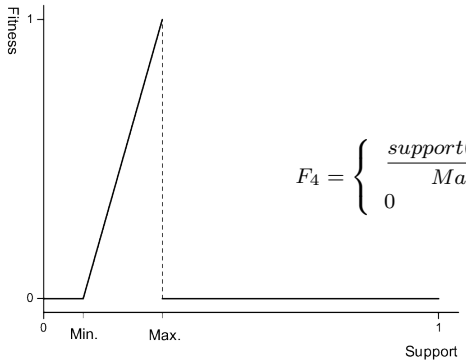
$$F_2 = \begin{cases} 1 & \text{if } Min \leq support(rule) \leq Max \\ 0 & \text{otherwise} \end{cases}$$

(b) Second fitness function



$$F_3 = \begin{cases} 1 + \dfrac{Min - support(rule)}{Max - Min} & \text{if } Min \leq support(rule) \leq Max \\ 0 & \text{otherwise} \end{cases}$$

(c) Third fitness function



$$F_4 = \begin{cases} \dfrac{support(rule) - Min}{Max - Min} & \text{if } Min \leq support(rule) \leq Max \\ 0 & \text{otherwise} \end{cases}$$

(d) Fourth fitness function

Figure 4.5: Different fitness functions for the RARM process

shown some sample rules obtained by using the Zoo dataset and the configuration parameters (see Table 4.4) previously described. The rule *(breathes = f AND backbone = f AND eggs != f) → (tail != t)* satisfies only 7 instances of the complete dataset, so this rule is considered as noisy according to the parameters fixed in Table 4.4, having a support value of 0.068. Therefore, a zero fitness value is assigned to this solution. On the contrary, the rule *(fins != f) → (backbone = t)* satisfies 17 instances, having a support value of 0.166, so a 0.455 fitness value is assigned to this rule. The farther a solution is to the interval limits, the higher its fitness value is. For example, the rule *(legs = 4) → (catsize = t)* obtains a support value of 0.255, satisfying 26 instances, so a fitness value of 0.950 is assigned to this solution. As shown, this first fitness function is specific for application domains with a large number of both noisy association rules and rules that are close to being considered as rare. In this way, a progressive fitness function is applied, since there is not a clear difference between noisy, rare, and frequent association rules. Therefore, the more distant the support value is from the predefined thresholds, the better is the rule.

The second fitness function, which is defined in Figure 4.5(b), is the most similar one to that previously used in existing RARM proposals. The reason is that the current algorithms do not differentiate among rules based on their support values. A clear difference is the use of a minimum support to determine which rules are noisy and which are RARs. Apart from the support, since rules belonging to the interval are equally promising, it is essential to establish a mechanism to properly differentiate among rules, bringing the confidence and lift measures into play. For example, using the aforementioned dataset, the rule *(feathers != f) → (hair != t)* is obtained, having a support value of 0.196, a confidence of 1.000 and a lift value of 1.741. This rule provides the same support, and therefore, the same fitness function value to the rule *(feathers != f) → (type != 6)*. Therefore, if it is required to select which one is better, the confidence and the lift values are analysed, the last rule having a confidence value of 1.000 and a lift value of 1.086. Therefore, the rule *(feathers != f) → (hair != t)* is more interesting since its lift value is higher, the antecedent implying an increment in the occurrence of the consequent. It should be noted that the main application domains of this fitness function are those where there is a clear difference between noisy and RARs.

Finally, two other fitness functions that respectively minimize and maximize the support within a given interval are presented. The former (see Figure 4.5(c)) assigns a fitness value based on the proximity of the support to the lower limit. This fitness function is specific for domains where there is a clear difference between noisy and RARs but the difference between rare and frequent rules is not that clear. The rule $(\textit{catsize} \mathrel{!}= f) \rightarrow (\textit{aquatic} = t)$ presents a fitness value of 0.904, having a support value of 0.128. Therefore, since the support is closer to the lower limit, the fitness function is closer to the maximum. The latter function, which is defined in Figure 4.5(d), maximizes the fitness value when the support of the rule is closer to the upper limit of the interval. This fitness function allows of differentiating between frequent and RARs in situations where there is a clear limit between both types. On the other hand, it is also useful where there is not a clear differentiation between noisy and rare rules. For a better understanding, the rule $(\textit{legs} = 4) \rightarrow (\textit{fins} = f)$ obtains a fitness function value of 0.921, and a support value of 0.374. Similarly to the fitness functions described above, these two functions provide a zero fitness value if the support value is outside the desired interval.

Following these fitness functions and since no restriction is applied to the support of each condition, both perfectly and imperfectly RARs could be discovered. These two types of rare rules have the maximum support restriction in common, as the approaches described in Section 2.2 do. Therefore, it represents an important advantage over currently existing proposals, where the process of extracting rules is limited to discover only perfectly or imperfectly RARs. Finally, it should be noted that no condition of any rule mined will have less support than the minimum threshold fixed since the support of an association rule allows of identifying the minimum value of any condition within the entire rule.

According to Table 4.5(a), which shows the average support values obtained by different algorithms using these four fitness functions, we can observe that they all behave as previously discussed. F_1 and F_2 discover rules that tend to be located in the middle of the valid interval. However, F_3 mines rules that can be found closer to the lowest bound of the interval and F_4 extracts rules closer to the highest limit.

In this experimental stage, it is analysed which fitness function behaves better in terms of average confidence, average lift and average number of rules. Focusing on the confidence and lift measures, a fitness function is better than another if

Table 4.5: Results obtained (presented in a per unit basis) by different algorithms using support and confidence as objectives to be maximized

(a) Average support values obtained with different datasets

Average *support*				
Dataset	F_1	F_2	F_3	F_4
Ankara	0.202	0.141	0.114	0.251
Autom	0.230	0.189	0.141	0.292
Credit	0.245	0.208	0.209	0.239
Mush	0.231	0.185	0.149	0.257
Prim	0.198	0.193	0.177	0.212
Soyb	0.235	0.159	1.133	0.265
Vote	0.249	0.234	0.182	0.254
WBC	0.234	0.207	0.142	0.316
WDatBC	0.221	0.196	0.155	0.259
WPBC	0.238	0.166	0.115	0.296
Zoo	0.215	0.183	0.136	0.269

(b) Average confidence values obtained with different datasets

Average *confidence*				
Dataset	F_1	F_2	F_3	F_4
Ankara	0.999	0.999	**1.000**	0.999
Autom	0.997	0.998	0.995	**0.999**
Credit	0.976	0.980	0.968	**0.985**
Mush	0.996	**1.000**	**1.000**	0.998
Prim	0.972	0.972	0.959	**0.976**
Soyb	**1.000**	0.999	0.996	**1.000**
Vote	0.970	0.971	0.968	**0.974**
WBC	**0.999**	0.998	0.998	0.998
WDatBC	0.995	**0.998**	0.997	0.996
WPBC	**1.000**	**1.000**	**1.000**	**1.000**
Zoo	**1.000**	**1.000**	**1.000**	**1.000**
Ranking	2.727	2.727	2.954	**2.045**

(c) Average lift values obtained with different datasets

Average *lift*				
Dataset	F_1	F_2	F_3	F_4
Ankara	1.311	**1.539**	1.475	1.225
Autom	1.754	1.843	**1.935**	1.590
Credit	1.026	1.033	**1.036**	1.018
Mush	1.161	**1.307**	1.231	1.213
Prim	1.054	1.060	**1.074**	1.047
Soyb	1.222	1.953	**1.958**	1.573
Vote	1.956	1.950	**2.082**	1.934
WBC	1.213	**1.904**	1.345	1.161
WDatBC	1.471	1.452	**1.589**	1.444
WPBC	**1.388**	1.279	1.242	1.198
Zoo	2.007	**2.883**	2.873	1.981
Ranking	2.818	1.818	**1.545**	3.818

(d) Average number of rules obtained with different datasets

Average *# rules*				
Dataset	F_1	F_2	F_3	F_4
Ankara	**20.0**	19.7	**20.0**	19.9
Autom	19.7	19.9	18.1	**20.0**
Credit	19.1	18.6	19.4	**20.0**
Mush	**20.0**	**20.0**	**20.0**	19.9
Prim	16.9	15.0	13.6	**18.5**
Soyb	**20.0**	**20.0**	**20.0**	**20.0**
Vote	18.0	18.7	18.7	**20.0**
WBC	19.8	**20.0**	**20.0**	**20.0**
WDatBC	**20.0**	**20.0**	**20.0**	**20.0**
WPBC	**20.0**	**20.0**	**20.0**	**20.0**
Zoo	**20.0**	**20.0**	**20.0**	**20.0**
Ranking	2.682	2.682	2.545	**2.091**

it obtains higher values for these measures. As for the average number of rules discovered, the best fitness function will be the one that obtains a number of rules closer to the maximum previously established by the data miner. In such a way, and in order to determine whether there exist significant differences among these four functions, a series of statistical tests [76, 77] were carried out.

The Friedman statistical test is used to compare the results obtained and to be able to precisely analyze whether there are significant differences among the four fitness functions. If the Friedman test rejects the null-hypothesis indicating that there are significant differences, then a Bonferroni-Dunn test is performed to reveal these differences.

The Friedman average ranking statistics for the average confidence measure (see Table 4.5(b)), distributed according to F_F with $k-1$ and $(k-1)(N-1)$ degrees of freedom, is 1.152; 18.009 for the average lift measure (see Table 4.5(c)); and 0.494 for the number of rules mined (see Table 4.5(d)). The results reveal that both the confidence and the number of rules belong to the critical interval $[0, (F_F)_{0.01,3,27} = 4.510]$, so the null-hypothesis that all the fitness functions perform equally well for these measures is not rejected using $\alpha = 0.01$. In consequence, focusing on the confidence and the average number of rules mined, it should be noted that the four fitness functions described in this work are equally valid for mining RARs. Nevertheless, F_4 provides the best ranking for both measures. Concerning the lift measure, F_3 obtains the best ranking since this function was conceived to obtain those rules that are close to the minimum support threshold. Notice that the lift measure tend to be higher as the support value decreases (see Equation 2.2). Besides, F_4 provides the worst ranking for the lift measure. This fitness function searches for rules close to the maximum support threshold.

The Friedman average ranking statistics for the average lift measure, distributed according to F_F, is equal to 18.009, which does not belong to the critical interval $[0, (F_F)_{0.01,3,27} = 4.510]$. Thus, we reject the null-hypothesis that all these fitness functions perform equally well for this measure. In order to determine whether there are significant differences among these fitness functions, the Bonferroni-Dunn test is used to reveal the difference in performance, 1.171 being the critical difference (CD) value for p = 0.1; 1.318 for p = 0.05; and 1.616 for p = 0.01. The results indicate that for the lift measure (see Figure 4.6), at a significance level of p = 0.01 (i.e., with a probability of 99%), there are significant differences between F_3 and F_4, the performance of F_3 being statistically better. Additionally, using a significance level of p = 0.10 (i.e., with a probability of 90%), there are significant differences between F_1 and F_3, the performance of F_3 being statistically better. Finally, it is

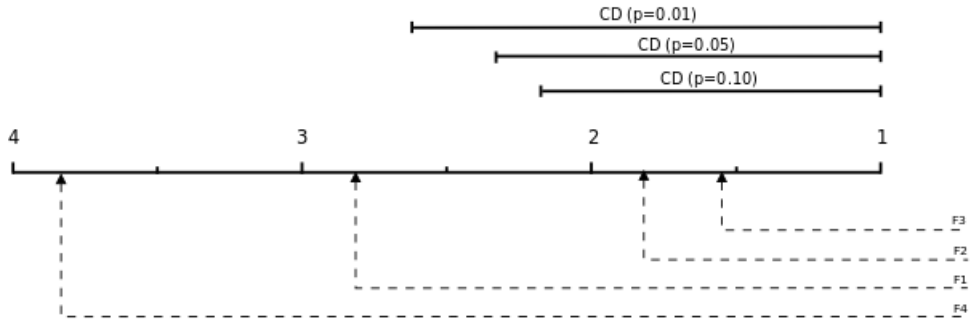


Figure 4.6: Critical difference obtained with the Bonferroni-Dunn test for the lift measure

not possible to assert that there are significant differences between F_2 and F_3, even when the latter obtains a better ranking.

Concluding the analysis, it is possible to state that despite the fact F_4 provides the best ranking for two out of three measures, it also enables rules of interest to be discovered. Furthermore, using the Friedman test is not possible to reject the null-hypothesis that all fitness functions perform equally well for confidence and average number of rules mined. Therefore, when the domains under application do not provide a clear difference between noisy, rare, and frequent rules, F_3 is the best fitness function to be used, discovering rules that are interesting and very reliable, and providing confidence values close to the maximum.

4.2.4 Comparing Different Algorithms

In this section, a comparison between other relevant RARM algorithms and our proposal, using the aforementioned F_3 fitness function, is performed. It is important to notice that the algorithm here presented does not require any previous step for discretizing numerical attributes, in contrast to the other proposals.

For the evaluation of each algorithm, a number of experiments were performed. Since the existing RARM algorithms use an exhaustive search methodology and to make a fair comparison, only a subgroup of the datasets could be selected: one had to avoid those that would discover a huge number of rules in exceeding any acceptable computation time. Table 4.6 shows the results obtained with the different algorithms. The datasets with numerical attributes were discretized using

Table 4.6: Results obtained by different algorithms

Dataset	Apriori-Inv	ARIMA	Apriori-Inf	MRG-Exp	Rare-G3PARM
Average *support*					
*Automobile*5	0.013	0.013	0.015	0.114	0.141
*Automobile*10	0.007	0.007	0.008	0.052	0.131
*Automobile*15	0.006	0.006	0.007	0.044	0.133
Vote	0.032	0.043	0.036	0.277	0.182
*Wisconsin*5	0.004	0.004	0.004	0.118	0.184
*Wisconsin*10	0.004	0.004	0.004	0.114	0.175
*Wisconsin*15	0.004	0.004	0.004	0.301	0.217
Zoo	0.051	0.056	0.062	0.234	0.136
Average *confidence*					
*Automobile*5	0.997	0.996	0.997	1.000	0.993
*Automobile*10	0.999	0.999	0.999	1.000	0.975
*Automobile*15	0.999	0.999	0.999	1.000	0.972
Vote	0.988	0.978	0.988	1.000	0.968
*Wisconsin*5	0.999	0.999	0.999	1.000	0.974
*Wisconsin*10	0.999	0.999	0.999	1.000	0.972
*Wisconsin*15	0.999	0.999	0.999	1.000	0.976
Zoo	0.998	0.996	0.996	1.000	1.000
Average number of rules					
*Automobile*5	2925.0	8519.0	1206.0	12.0	19.5
*Automobile*10	7836.0	15505.0	3300.0	24.0	18.4
*Automobile*15	6021.0	12248.0	2837.0	38.0	19.6
Vote	32.0	63253.0	24.0	6.0	18.7
*Wisconsin*5	20378.0	61436.0	6199.0	40.0	15.2
*Wisconsin*10	16851.0	47855.0	6365.0	97.0	15.4
*Wisconsin*15	21489.0	49881.0	8410.0	24.0	11.3
Zoo	815.0	159598.0	368.0	54.0	20.0
Time (sec)					
*Automobile*5	60.350	293.208	1.240	0.561	1.417
*Automobile*10	350.179	2505.419	6.894	0.735	0.983
*Automobile*15	207.264	2270.975	8.559	1.143	0.942
Vote	0.293	11113.984	0.166	0.202	1.343
*Wisconsin*5	1821.290	2966.342	14.611	1.337	1.726
*Wisconsin*10	865.962	8398.223	20.829	1.607	1.583
*Wisconsin*15	1257.343	7127.010	28.627	1.278	1.709
Zoo	0.629	8669.034	0.304	0.788	0.606

equal-width discretization techniques in order to be able to make a comparison between the existing RARM algorithms. The results show the average support, the average confidence, the number of rules mined, and the average runtime for each algorithm. Here, $D - N$ signifies that the dataset D was discretized into N intervals. Finally, it should be noted that the lift has been omitted in the comparison, since current exhaustive search algorithms in the RARM field do not consider this lift measure. Therefore, including the lift would lead to an unfair comparison.

Analysing the results, notice that the existing RARM proposals obtain rules with very low support, which could be considered as noise, not considering the interest of the rules mined by using the lift measure. On the other hand, our proposal obtains a set of RARs with support values in the range $[0.1, 0.4]$, which is the interval used to determine whether a rule is rare. This interval could vary, based on the specific domain under application and the data miner's needs. In addition, the algorithm here presented provides interesting rare rules since the lift measure has made its appearance on the scene. Only those rules providing antecedents with a positive influence on their consequent are considered interesting.

An important issue is that Apriori-Infrequent obtains an average support greater than Apriori-Inverse. As mentioned in previous sections, Apriori-Infrequent obtains rules from the infrequent item-sets mined in the classical Apriori algorithm. Therefore, the rules obtained have at least one condition with a support value greater than the threshold. Notice that no superset is mined from the infrequent item-sets obtained. On the other hand, Apriori-Inverse discovers only perfectly RARs so their support values are very low. Studying the performance of the ARIMA algorithm, it should be noticed that its results are very similar to those obtained with Apriori-Inverse since their difference is only in the minimal item-sets. As for MRG-Exp, it obtains higher support values than the other exhaustive search algorithms, since it uses minimal item-sets to discover RARs. However, MRG-Exp does not overcome the problem of discovery noisy patterns. For instance, it discover the rule *IF cylinders* $(4.66 - 5]$ *THEN origin* $(1.93 - 2.07]$, which has a support value of 0.01. Focusing on the average confidence, notice that all the proposals used in the analysis obtain reliable rules, with an average confidence above 0.975.

According to the average number of RARs mined, our proposal obtains a uniform set of rules (between 15 and 20 rules) that is close to the number of rules previously specified by the data miner (20 in this experimental stage). On the other hand, the exhaustive search algorithms obtain a heterogeneous set of rules, depending on the datasets used. For example, the Apriori-Inverse algorithm obtains 32 RARs using the Vote dataset and 21489 rules using the Wisconsin dataset. Anyhow, none of these algorithms can ensure that the rules discovered are interesting. Therefore, using some datasets, these algorithms obtain a large number of association rules, which is hardly manageable. Apriori-Infrequent only mines rules from the infrequent patterns discovered using the regular Apriori algorithm. Thus, the number of rules discovered using this algorithm is lower than others. Using the Apriori-Inverse provides a higher number of rules, since it mines infrequent patterns and PRARs. ARIMA is the algorithm that discovers the highest number of association rules, since it mines not only PRARs but also IRARs. Finally, MRG-Exp discovers the lower number of rules, since it only uses minimal item-sets. Concluding the analysis of Table 4.6, it should be noted the strength of the proposed algorithm, which is able to obtain up to a predefined maximum number of rules, even in those cases when this number would become unmanageable.

Focusing on the runtime for each algorithm, notice that MRG-Exp and Rare-G3PARM have an average execution time lower than those obtained by the other algorithms. In exhaustive search algorithms, the execution time is not uniform, but depends directly on the dataset used. For instance, using ARIMA, the execution time may vary from 293 to 11,113 seconds. However, the proposal presented in this work obtains rules in a uniformly short time, around one second, independently of the used datasets. This is a very important advantage of our approach.

Finally, Figure 4.7 lists a few RARs extracted from the execution of the Rare-G3PARM algorithm over the original Automobile Performance dataset. As shown, the rules discovered are very reliable, exceeding the minimum support threshold (set to 0.100 using a per unit base) to avoid the extraction of noisy rules. Finally, notice that using numerical values and operators, also allows obtaining intervals over certain attributes. For example, rule 1 determines that the *horsepower* attribute has a value in the range $(82, 118]$. This property is an important advantage of the algorithm here proposed over the currently existing proposals in this field.

Table 4.7: Examples of RARs over a numerical dataset

#Rule	RARs over numerical attributes	Support	Confidence	Lift
(1)	**IF** ($origin < 2$)	0.184	1.000	1.960
	AND ($cylinders >= 5$)			
	AND ($horsepower <= 118$)			
	AND ($horsepower > 82$)			
	THEN ($displacement > 145.4$)			
(2)	**IF** ($model\_year <= 76$)	0.179	1.000	2.074
	AND ($horsepower >= 82$)			
	AND ($weight >= 3728$)			
	THEN ($cylinders >= 5$)			
(3)	**IF** ($origin > 2$)	0.138	1.000	1.675
	AND ($model\_year <= 76$)			
	THEN ($displacement < 222.8$)			

4.3 Conclusions

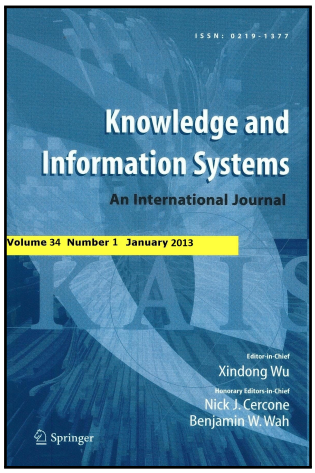
Current algorithms for mining RARs mainly use exhaustive search methods. Therefore, these algorithms cannot be successfully used with huge datasets comprising a large number of attributes because of the large search space generated. Moreover, these algorithms only obtain RARs over categorical domains. Additionally, existing algorithms in this field obtain very infrequent rules that may be considered as noise. In this sense, we have presented a grammar-based algorithm for mining interesting and reliable RARs. Four fitness functions were presented and properly studied, allowing of establishing the boundary between rare and noisy ARs. The experimental analysis of these fitness functions showed that the most interesting and reliable ARs were obtained by means of a fitness function that was specifically designed for domains where there is a clear difference between noisy and RARs but the difference between rare and frequent is not clear.

4.4 Publications Associated with this Chapter

- **International Journal:**

  TITLE: *On the Adaptability of G3PARM to the Extraction of Rare Association Rules* [81]

  AUTHORS: J. M. LUNA, J. R. ROMERO AND S. VENTURA



  KNOWLEDGE AND INFORMATION SYSTEMS, ACCEPTED.
  DOI: 10.1007/s10115-012-0591-9

  RANKING:

  IMPACT FACTOR (JCR 2011): 2.225

  KNOWLEDGE AREA:

  COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE: 21/111

  COMPUTER SCIENCE, INFORMATION SYSTEMS: 18/135

- **International Conference:**

  J.M. LUNA, J.R. ROMERO AND S.VENTURA. *Mining and Representing Rare Association Rules through the Use of Genetic Programming*, PROCEEDINGS OF THE 3RD WORLD CONGRESS ON NATURE AND BIOLOGICALLY INSPIRED COMPUTING, NABIC 2011, PP. 86-91, ISBN: 978-145771123-7 [82]

5

Multi-Objective G3P for Mining Association Rules

In this chapter, we present two new G3P proposals for mining ARs following a multi-objective strategy. These proposals benefit from the advantages of both G3P [58] and consequently EA [10], and combine them with those of multi-objective models [65]. More specifically, the proposals presented here are based on two well-known multi-objective algorithms: the Non dominated Sort Genetic Algorithm (NSGA-2) [72] and the Strength Pareto Evolutionary Algorithm (SPEA-2) [83]. Because of the specific grammar definition, these G3P proposals enable the extraction of rules from both numerical and categorical domains. Finally, in order to demonstrate the usefulness of the proposed algorithms, different measures are considered as objectives to obtain a set of optimal solutions. More specifically, the experiments performed combine both the support-confidence and support-lift measures. The ARs obtained have shown to be very frequent (with support values above 95% in most cases) and reliable (with confidence values close to 100%). Furthermore, for the trade-off between support and lift, the multi-objective proposals also produce very interesting and representative rules.

5.1 The G3PARM Multi-Objective Algorithms

In this section, we propose two multi-objective G3P proposals for extracting ARs from different domains and types of datasets. Both approaches are founded on two well-known multi-objective algorithms: NSGA-2 [72] and SPEA-2 [83]. The use of G3P allows us to define expressive and understandable individuals in both numerical and categorical domains. Both proposals have several characteristics in common, such as the encoding criterion or the genetic operators used throughout the evolutionary process. In this section, the main characteristics of both proposals are outlined.

5.1.1 Encoding

As mentioned in the G3PARM algorithm (see Chapter 3), a CFG is defined as a four-tuple (Σ_N, Σ_T, P, S) where Σ_T represents the alphabet of terminal symbols, Σ_N the alphabet of non-terminal symbols, and $\Sigma_N \cap \Sigma_T = \emptyset$.

Figure 5.1 shows the grammar used to represent each individual, which briefly differ to the one proposed by G3PARM. Now, the consequent comprises only one condition, so the rules discovered are more understandable for the user. Notice that the terminal symbol *"name"* adopts the name of any of the attributes, randomly selected from the set of available attributes. For example, using the sample metadata from Table 5.1, the terminal symbol *"name"* may adopt any value, such as *colour, size, shape, area* or *perimeter*. Once the attribute for this terminal symbol is assigned, a random value is then selected. For instance, the attribute *colour* may be assigned to different values such as *red, green, blue* and *black*.

One of the most important features of these proposals is that they permit us to represent individuals in both numerical and categorical domains. Similarly to the proposals presented in Chapters 3 and 4, the process of producing an individual begins from the start symbol *Rule* and continues by randomly applying production rules belonging to the set P until a valid derivation sequence is successfully completed.

G = $(\Sigma_N, \Sigma_T, P, S)$ with:

S = *Rule*

Σ_N = {*Rule, Antecedent, Consequent, Comparison, Categorical_Comparator, Categorical_Attribute_Comparison, Numerical_Comparator, Numerical_Attribute_Comparison*}

Σ_T = {*'AND', '!=', '=', '<=', '<', '>=', '>', 'name', 'value'*}

P = {*Rule = Antecedent, Consequent ;*
Antecedent = Comparison | 'AND', Comparison, Antecedent ;
Consequent = Comparison ;
Comparison = Categorical_Comparator, Categorical_Attribute_Comparison | Numerical_Comparator, Numerical_Attribute_Comparison;
Categorical_Comparator = '!=' | '=' ;
Numerical_Comparator = '<=' | '<' | '>=' | '>' ;
Categorical_Attribute_Comparison = 'name', 'value' ;
Numerical_Attribute_Comparison = 'name', 'value' ;}

Figure 5.1: Context-free grammar expressed in extended BNF notation

Table 5.1: Meta-data of a sample dataset showing the attributes and their available values

Attributes	**Values**
colour	red, green, blue, black
size	small, normal, big
shape	circle, square, triangle
area	[0, 100]
perimeter	[0, 10]

5.1.2 Genetic Operators

In order to generate new individuals in a given generation of the evolutionary algorithm process, two genetic operators are presented: crossover and mutation. These genetic operators (similarly to the operators of G3PARM) search for individuals with a support value greater than the original ones. To this end, these genetic operators work on the lowest support condition within each rule and obtain another one with a higher support.

Crossover. To facilitate the discovery of new individuals with a higher support, this genetic operator swaps the condition with the lowest frequency of occurrence within a parent with the one that has the greatest frequency of occurrence in another parent.

Mutation. Like the crossover genetic operator, this operator tries to generate a new individual with a higher support than the original one. This operator obtains a new individual from only one parent by working on the lowest frequency of occurrence condition.

5.1.3 The NSGA-G3PARM Algorithm

The proposal called NSGA-G3PARM is founded on the NSGA-2 [72] multi-objective algorithm, which is adapted to the characteristics of G3P. The pseudocode of the NSGA-G3PARM algorithm is shown in Algorithm 6. In this proposal, different measures, which serve to determine the quality of the rules mined, are used as objective functions. Since this algorithm follows an evolutionary strategy, it obtains the subset *parents* of individuals to be crossed and mutated (lines 9 to 10). Also, notice that any repetition will be removed from the *population* resulting from joining the current population with the recently created set, *mutatedPopulation* (line 11). New individuals are evaluated to determine the values of the quality measures (line12). Since the ultimate goal is to return a pre-defined number of optimal solutions, solutions have to be organized in fronts (line 13). Therefore, the algorithm continues to identify those solutions from the entire set that belong to the POF, i.e., those solutions that are not dominated by any other. After obtaining a first front, the process is repeated on the remaining solutions, so new fronts are calculated. Ascertaining the density of the solutions surrounding a particular solution serves to determine which solution is best in each front. To this end, the average distance to each solution around each of its objectives is calculated (line 14). Those solutions having the highest and lowest values of each objective are assigned an infinite distance value. On the other hand, intermediate solutions are assigned a distance value equal to the absolute normalized difference in the objective function values of two adjacent solutions. Finally, the overall distance value for each solution is calculated as the sum of the distance values for each objective function. In the final step of each generation, the algorithm keeps a number of the best solutions, i.e.,

Algorithm 6 NSGA-G3PARM algorithm

Require: $max\_generations, population\_size$
Ensure: $paretoOptimalFront$
1: $population \leftarrow$ generatePopulation($population\_size$)
2: $parents \leftarrow \emptyset$
3: $crossedPopulation \leftarrow \emptyset$
4: $mutatedPopulation \leftarrow \emptyset$
5: $paretoOptimalFront \leftarrow \emptyset$
6: $num\_generations \leftarrow 0$
7: **while** $num\_generations < max\_generations$ **do**
8: $parents \leftarrow$ obtainParents($population$)
9: $crossedPopulation \leftarrow$ crossover($parents$)
10: $mutatedPopulation \leftarrow$ mutation($crossedPopulation$)
11: $population \leftarrow$ duplicateRemoved($population \cup mutatedPopulation$)
12: evaluate($population$)
13: $listOfFronts \leftarrow$ obtainFronts($population$)
14: densityOfSolutions($listOfFronts$)
15: $i \leftarrow 2$
16: $population \leftarrow F_1$
17: $paretoOptimalFront \leftarrow F_1$
18: **while** $|population| + |F_i| \leq population\_size$ **do**
19: $population \leftarrow population \cup F_i$
20: $i{+}{+}$
21: **end while**
22: $population \leftarrow population \cup$ getIndividualsSortingByDistance($population\_size - |population|, F_i$)
23: $num\_generations + +$
24: **end while**
25: **return** $paretoOptimalFront$

those having a higher distance, starting from the first front and continuing with the rest of fronts, if necessary (lines 18 to 22). Once the algorithm reaches a certain number of generations $max\_generations$, the resulting set $paretoOptimalFront$ is returned (line 25).

5.1.4 The SPEA-G3PARM Algorithm

In this case, the SPEA-G3PARM algorithm has been adapted to conform to the SPEA-2 [83] algorithm and the characteristics of G3P. The pseudocode of this algorithm is shown in Algorithm 7. The algorithm starts by obtaining the set *population* of individuals (line 6). Since this algorithm follows an evolutionary

Algorithm 7 SPEA-G3PARM algorithm

Require: $max\_generations$, $population\_size$, $paretoSize$
Ensure: $paretoOptimalFront$
1: $num\_generations \leftarrow 0$
2: $parents \leftarrow \emptyset$
3: $paretoOptimalFront \leftarrow \emptyset$
4: $crossedPopulation \leftarrow \emptyset$
5: $mutatedPopulation \leftarrow \emptyset$
6: $population \leftarrow$ generatePopulation($population\_size$)
7: evaluate($population$)
8: **while** $num\_generations < max\_generations$ **do**
9: $paretoOptimalFront \leftarrow$ obtainParetoFront($population \cup paretoOptimalFront$)
10: **if** size($paretoOptimalFront$) $> paretoSize$ **then**
11: $paretoOptimalFront \leftarrow$ reduceSize($paretoOptimalFront$)
12: **else**
13: **if** size($paretoOptimalFront$) $< paretoSize$ **then**
14: $paretoOptimalFront \leftarrow$ incrementSize($paretoOptimalFront$, $population$)
15: **end if**
16: **end if**
17: $parents \leftarrow$ obtainParents($paretoOptimalFront \cup population$)
18: $crossedPopulation \leftarrow$ crossover($parents$)
19: $mutatedPopulation \leftarrow$ mutation($crossedPopulation$)
20: $population \leftarrow mutatedPopulation$
21: evaluate($population$)
22: $num\_generations++$
23: **end while**
24: **return** $paretoOptimalFront$

strategy, the set *population* evolves through the generations (lines 8 to 23), creating new individuals by means of genetic operators. The main characteristic of this algorithm is that each individual is evaluated according to the Cartesian distance with its k-th nearest neighbours in the population and the number of individuals that dominate each individual (*raw* value). If the individuals establish few dominance relationships among each other (e.g., they all lie in the POF), then large groups of solutions with the same fitness may be obtained, so it is necessary to compute a nearest neighbour density estimation. Thus, given an individual i, the higher the number of individuals dominated by i and the higher the Cartesian distance with its neighbours, the lower the fitness function value reached. It is important to note that the goal of this algorithm is to minimize the fitness value (see Equation 5.1).

$$fitness(i) = \frac{1}{Cartesian\_distance(i, k^{th}\ nearest\ neighbour) + 2} + raw(i) \quad (5.1)$$

In each generation, the algorithm generates the POF, which is stored in the set $paretoOptimalFront$, from the set that results from merging the current population and the old POF (line 9). If the size of this POF is greater than $paretoSize$, then the POF has to be downsized by choosing the best individuals ranked according to the fitness function (lines 10 to 11). On the other hand, if the size of the POF is less than a pre-defined size, then it is necessary to fill the POF with the best solutions from the second front (lines 13 to 14). Once the POF is generated, a set of parents is chosen by merging the current population and the new POF (line 17), and new solutions are obtained with the genetic operators (lines 18 to 20). Finally, after completing a given number of generations, the resulting set $paretoOptimalFront$ is returned (line 24).

5.2 Experimental Study

Different experiments were carried out, the results of which are presented in this section. Firstly, the datasets used and the experimental set-up are explained. Thereafter, a series of analyses are performed to determine the quality of each POF mined and the behaviour of the different quality measures in the proposed G3P multi-objective optimization algorithms.

5.2.1 Datasets

The results [1] shown in this experimental section correspond with the average values calculated after running each algorithm 30 times with different seeds. Ten datasets with different numbers of instances and attributes were used (see Table 5.2). All the experiments were performed on a 12Gb main memory Intel Core i7 machine,

[1] A detailed description of the results can be found at http://www.uco.es/grupos/kdis/kdiswiki/MO-G3P_ARM

Table 5.2: Datasets used in the experimental stage

Dataset	Abbreviation	#Inst.	#Attr.	Attr. Type
Automobile	*Autom*	392	8	Num.
Credit German	*Credit*	1000	21	Categ., Num.
House_16H	*HH*	22784	17	Num.
Minorities at Risk	*MAR*	852	22	Categ., Num.
Minorities at Risk Organ. Behaviour	*MAROB*	1789	50	Categ., Num.
Mushroom	*Mush*	8124	23	Categ.
Soybean	*Soyb*	683	36	Categ.
Wisconsin Breast Cancer	*WBC*	683	11	Categ., Num.
Wisconsin Diagnostic Breast Cancer	*WDatBC*	569	31	Categ., Num.
Wisconsin Prognostic Breast Cancer	*WPBC*	194	34	Categ., Num.

running CentOS 5.4. Similarly to algorithms presented in previous chapters, these algorithms were written in Java using JCLEC [74].

5.2.2 Experimental Set-up

In Chapter 3, the best found combination of parameters was described for the G3PARM algorithm. Thus, the same combination of parameters is used in this experimental stage. Also, since both multi-objective algorithms are based on G3PARM, and in order to make a fair comparison, the same parameters set-up is used for the three algorithms.

Table 5.3: Best combination of parameters found by the authors

Parameter	G3PARM	NSGA-G3PARM	SPEA-G3PARM
Population size	50	50	50
External population	20	-	-
Pareto front size	-	-	20
# Generations	100	100	100
Max. derivations	24	24	24
Nearest neighbour	-	-	5
Crossover prob.	70%	70%	70%
Mutation prob.	14%	14%	14%
Confidence threshold	90%	-	-
Support threshold	70%	-	-

Table 5.3 shows the best found combination of parameters. The best results were obtained using a population size of 50 individuals, 100 generations, 70% crossover probability, 14% mutation probability, and a maximum derivation size of 24. For the G3PARM algorithm, the external or elite population size is 20 and the thresholds of support and confidence are set to 70% and 90%, respectively. For the SPEA-G3PARM, the Cartesian distance was calculated with the fifth nearest neighbour and the maximum Pareto front size was fixed to 20 to perform a fair comparison against G3PARM.

5.2.3 Analysis of the Experiments

In this section, a comparative study between both multi-objective proposals is performed. Firstly, an study of different statistical tests used in this experimental study is carried out. Secondly, an analysis of the POF obtained by each proposal is presented, and finally, the quality of the extracted rules is evaluated.

Analysis of the POF Quality. Many performance measures, which evaluate different POF characteristics, have been proposed in the literature [66]. Three of the most widely used — spacing, hyper-volume and coverage of sets — are analysed. The average results from the datasets mentioned above and using a support-confidence framework are shown in Table 5.4. The spacing measure numerically describes the spread of the solutions in the POF. Analysing the results obtained, the POF of the NSGA-G3PARM algorithm provides a set of solutions more equally spaced than SPEA-G3PARM, the spacing value being the lowest one. Using the hyper-volume, which is defined as the area of the POF coverage with respect to the objective space, the NSGA-G3PARM algorithm obtains the highest value, therefore, its POF covers a higher area than the POF of SPEA-G3PARM.

Table 5.4: Average results obtained for different quality measures of the POF using a support-confidence framework

Algorithm	Spacing	Hyper-volume	Two set coverage
NSGA-G3PARM	0.012	0.987	CS(NSGA-G3PARM,SPEA-G3PARM) = 0.3
SPEA-G3PARM	0.015	0.986	CS(SPEA-G3PARM,NSGA-G3PARM) = 0.1

Finally, the coverage of the two sets is evaluated. This measure determines the relative coverage comparison of the POF from two different algorithms. The results show that NSGA-G3PARM produces the highest value, dominating the outcomes of SPEA-G3PARM. Taking all these measures into account, it could be said that NSGA-G3PARM obtains a higher quality POF when support and confidence measures are used.

Since this analysis is based on the average values of several measures across datasets that have different characteristics, it is not particularly meaningful. Therefore, we performed the Wilcoxon signed rank test [76], obtaining a 0.17 p-value for the spacing, 0.08 for the hyper-volume, and 0.02 for the two set coverage measure. So, at a significance level of $\alpha = 0.01$, there are no significant differences between the two multi-objective approaches.

Studying the POF quality with a support-lift framework (see Table 5.5), it is possible to determine that NSGA-G3PARM provides more equally spaced solutions than SPEA-G3PARM. Focusing on the hyper-volume measure, SPEA-G3PARM produces the greatest value, so its POF covers a greater area. Finally, focusing on the dominance of each POF, NSGA-G3PARM is seen to achieve a greater value than SPEA-G3PARM, so NSGA-G3PARM dominates the outcomes of SPEA-G3PARM. Taking all these results into account, it can be stated that NSGA-G3PARM obtains a higher quality POF than SPEA-G3PARM.

Table 5.5: Average results obtained for different quality measures of the POF using a support-lift framework

Algorithm	**Spacing**	**Hyper-volume**	**Two set coverage**
NSGA-G3PARM	3.6	1.9	CS(NSGA-G3PARM,SPEA-G3PARM) = 0.4
SPEA-G3PARM	24.3	2.1	CS(SPEA-G3PARM,NSGA-G3PARM) = 0.2

Since the results obtained from this analysis are not sufficiently meaningful, the Wilcoxon signed rank test should be performed. The test shows a 0.002 p-value for spacing, 0.037 for hyper-volume, and 0.006 for the two set coverage measure. At a significance level of $\alpha = 0.01$, there are significant differences between the two multi-objective versions for the spacing measure and the NSGA-G3PARM version

is statistically better. On the contrary, at the same significance level, the SPEA-G3PARM version is statistically better for the two set coverage measure. Finally, at a significance level of $\alpha = 0.01$, there are no significant differences between the two multi-objective versions for the hyper-volume measure. Therefore, regardless of whether a support-confidence framework or a support-lift framework is used, no significant differences are seen to exist between the two multi-objective proposals. In consequence, both approaches should be compared with the original G3PARM (see Chapter 3). The results are shown in Tables 5.6 and 5.8 (the best results for each measure are highlighted in bold).

Analysis of the Rules Mined. In this study, we evaluate the performance of the G3P proposals by comparing them in terms of their average support, average confidence, and the coverage (i.e., the percentage of instances covered) for each algorithm (see Table 5.6). The average ranking for each algorithm is shown in Table 5.7. Finally, Figure 5.2 depicts the difference of the corresponding average of rankings for each quality measure, showing the critical difference (CD) for different p values. In such a way, it is easy to determine whether significant differences exist between the algorithms.

Focusing on the results presented in Table 5.6(a), the three algorithms mine highly representative rules, with a support value above 0.95 in most cases. NSGA-G3PARM achieves the highest support values for most datasets. All algorithms obtain very reliable association rules (see Table 5.6(b)) and there are no apparent differences between them. Analysing the coverage measure (see Table 5.6(c)), the SPEA-G3PARM algorithm produces the best results, covering all the instances in most datasets. An analysis of the G3PARM and NSGA-G3PARM algorithms shows that they cover a large amount of instances (above 0.97) and, in some datasets, they manage to cover all the instances, e.g., in the *MAROB* and *Mush* datasets. Finally, the number of rules mined is homogeneous for G3PARM and SPEA-G3PARM. Due to the nature of the SPEA algorithm described in Section 3.4, it discovers a set of rules equal to the size of its POF (20 rules). On the other hand, the G3PARM algorithm also obtains 20 rules at most, constrained by its external population size. However, using some datasets, the latter does not reach its population size limit (e.g., *HH* and *WDatBC* datasets). On the other hand, the NSGA-G3PARM algorithm discovers a heterogeneous set of between 1 and 60 rules, depending on

Table 5.6: Results obtained (presented per unit) by different algorithms using support and confidence as objectives to be maximized

(a) Average support values obtained with different datasets

	Average *support*		
Dataset	**G3PARM**	**NSGA-G3PARM**	**SPEA-G3PARM**
Autom	0.819	**0.876**	0.854
Credit	0.913	**0.994**	0.972
HH	0.922	**0.998**	0.977
MAR	0.999	**1.000**	0.999
MAROB	**1.000**	**1.000**	**1.000**
Mush	0.998	**1.000**	0.999
Soyb	0.883	**0.983**	0.924
WBC	0.991	**0.997**	0.994
WDatBC	0.783	**0.876**	0.790
WPBC	0.963	**0.989**	0.968

(b) Average confidence values obtained with different datasets

	Average *confidence*		
G3PARM	**NSGA-G3PARM**	**SPEA-G3PARM**
0.991	**0.993**	0.991
0.999	**1.000**	0.997
0.999	**0.999**	**0.999**
1.000	**1.000**	0.999
1.000	**1.000**	**1.000**
1.000	**1.000**	0.999
0.997	**0.998**	0.991
0.999	**1.000**	0.999
0.978	**0.994**	0.987
0.999	**1.000**	0.996

(c) Average instances covered with different datasets

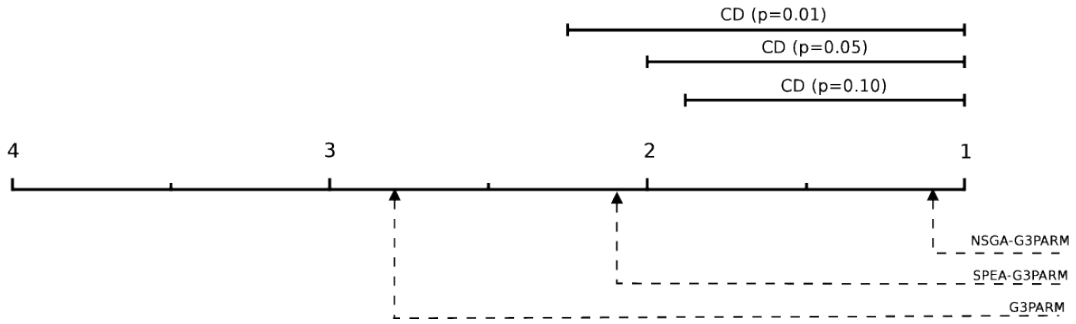
	Instances covered (per unit basis)		
Dataset	**G3PARM**	**NSGA-G3PARM**	**SPEA-G3PARM**
Autom	0.979	0.988	**1.000**
Credit	**1.000**	0.998	**1.000**
HH	0.999	0.999	**1.000**
MAR	**1.000**	**1.000**	**1.000**
MAROB	**1.000**	**1.000**	**1.000**
Mush	**1.000**	**1.000**	**1.000**
Soyb	**1.000**	0.993	**1.000**
WBC	0.998	0.997	**0.999**
WDatBC	**0.998**	0.991	0.997
WPBC	0.999	0.995	**1.000**

(d) Average number of rules obtained with different datasets

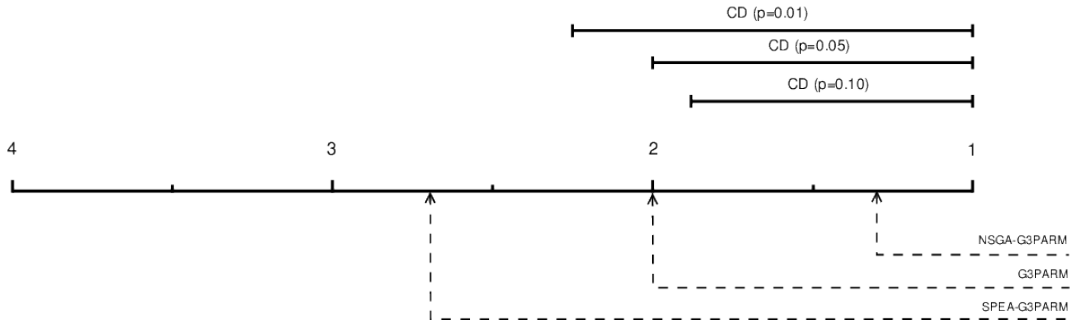
	Average *Average number of rules*	
G3PARM	**NSGA-G3PARM**	**SPEA-G3PARM**
19.8	18.5	20.0
19.9	3.9	20.0
19.3	17.5	20.0
20.0	34.4	20.0
20.0	49.9	20.0
20.0	31.9	20.0
20.0	2.3	20.0
19.0	12.7	20.0
19.4	4.8	20.0
20.0	8.5	20.0

Table 5.7: Average ranking of each algorithm for each measure

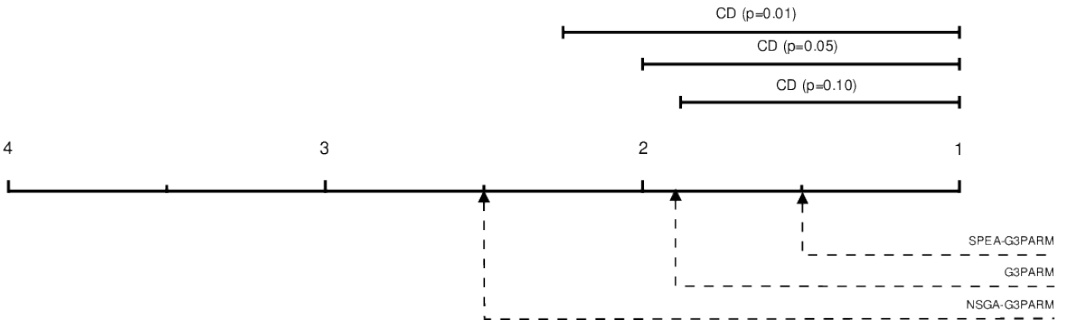
Measure	**G3PARM**	**NSGA-G3PARM**	**SPEA-G3PARM**
Support	2.8	**1.1**	2.1
Confidence	2.0	**1.3**	2.7
Coverage	1.9	2.5	**1.5**



(a) Critical difference obtained with the Bonferroni-Dunn test for the support measure



(b) Critical difference obtained with the Bonferroni-Dunn test for the confidence measure



(c) Critical difference obtained with the Bonferroni-Dunn test for the coverage measure

Figure 5.2: Critical differences obtained with the Bonferroni-Dunn test for different measures when using support and confidence as objectives to be optimized

the dataset. As mentioned above, this algorithm does not have a maximum POF size, so the number of rules mined may vary greatly.

The Friedman average ranking statistics for average support measure distributed according to F_F with $k-1$ and $(k-1)(N-1)$ degrees of freedom is 24.3; 8.6 for the average confidence measure; and 3.4 for the coverage measure. The support does not belong to the critical interval $[0, (F_F)_{0.01,2,18} = 6.0]$. Thus, we reject the null-hypothesis that all algorithms perform equally well for the support measure

using $\alpha = 0.01$. Using the same critical interval $[0, (F_F)_{0.01,2,18} = 6.0]$, the Friedman test rejects the null-hypothesis that all algorithms perform equally well for the confidence measure. Finally, using the critical interval $[0, (F_F)_{0.1,2,18} = 2.6]$, with $\alpha = 0.1$, the Friedman test rejects the null-hypothesis for the coverage measure. In order to analyse whether there are significant differences among the three algorithms using all the measures, the Bonferroni-Dunn test is used to reveal the difference in performance (See Figure 5.2), 0.8 being the critical difference (CD) value for a significance level of $p = 0.1$; 1.0 for $p = 0.05$; and 1.2 for $p = 0.01$.

With regard to the support measure (see Figure 5.2(a)), the results indicate that at a significance level of $p = 0.01$ (i.e., with a probability of 99%), there are significant differences between NSGA-G3PARM and G3PARM, the performance of the former being statistically better. Using a significance level of $p = 0.1$ (i.e., with a probability of 90%), there are significant differences between SPEA-G3PARM and NSGA-G3PARM, the performance of the latter being statistically better. Finally, it is not possible to assert that there are significant differences between G3PARM and SPEA-G3PARM, despite the fact that SPEA-G3PARM produces the best ranking.

If we focus on the confidence measure, as shown in Figure 5.2(b), with a probability of 99%, it is possible to state that there are significant differences between NSGA-G3PARM and SPEA-G3PARM, the former being statistically better. However, it is not possible to state that there are significant differences between G3PARM and both multi-objective proposals, NSGA-G3PARM being statistically better.

Finally, the coverage measure shown in Figure 5.2(c) establishes that when a significance level of $p = 0.1$ is used, there are significant differences between both multi-objective proposals, the performance of SPEA-G3PARM being statistically better. Moreover, it is not possible to state that, with a probability of 90%, there are significant differences between SPEA-G3PARM and G3PARM, but the former obtains the best ranking. Analysing NSGA-G3PARM and G3PARM, it is not possible to assert that, with a significance level of $p = 0.1$, there are any significant differences between the two algorithms but the latter obtains the best ranking.

To conclude this analysis, the G3P-based multi-objective proposals perform very well when using support and confidence as objectives to be maximized. In such

a situation, they offer a good alternative to G3PARM by mining more representative and reliable rules, especially the NSGA-G3PARM proposal. However, some situations require discovering rules with the lowest support but of high interest, as mentioned in Section 2.1. An example can be found when analysing the average results obtained using the *MAROB* dataset. Regardless of the algorithm used, the results obtained with this dataset is always equal to 1.0 for both measures, support and confidence. Therefore, all rules discovered using this dataset have a lift value of 1.0, i.e., the antecedent and consequent are statistically independent. Thus, it is valuable to perform a new analysis for discovering interesting rules, so lift and support measures are used as objectives to be maximized. In order to make a fair comparison, G3PARM was modified for mining only rules with a lift value greater than one, i.e., rules with interest.

The results presented in Table 5.8 show that multi-objective proposals discover rules with a high interest at the expense of a decrease in frequency (see Table 5.8(a)), as is to be expected. Focusing on the lift measure shown in Table 5.8(b), the best results are obtained with the SPEA-G3PARM algorithm, while G3PARM discovers more reliable rules (see Table 5.8(c)). It is worth mentioning that the confidence of the rules mined with the multi-objective G3P proposals decreases more than with G3PARM. Studying the number of rules discovered (see Table 5.8(e)), it is seen that the G3PARM and the SPEA-G3PARM algorithms mine a small, homogeneous set of rules. The latter obtains a set of rules equal to the size of its POF, while the former discovers a maximum of 20 rules, constrained by its external population size. If we focus on the NSGA-G3PARM algorithm, it obtains a large set of rules (between 32 and 49 rules depending on the dataset used). Finally, when analysing the coverage measure (see Table 5.8(d)), the NSGA-G3PARM algorithm is seen to mine the best results, covering all the instances in many datasets thanks to the high number of rules mined.

Different statistical tests [76, 77] were carried out based on the average ranking for each algorithm (see Table 5.9). The Friedman average ranking statistics for average support measure distributed according to F_F with $k-1$ and $(k-1)(N-1)$ degrees of freedom is 5.1E15, 90.9 for average lift, 5.1E15 for average confidence, and 8.1 for coverage (percentage of instances covered). None of them belong to the critical interval $[0, (F_F)_{0.01,2,18} = 3.5]$. Thus, we reject the null-hypothesis

Table 5.8: Results obtained (presented in a per unit basis) by different algorithms using support and lift as objectives to be maximized

(a) Average support values obtained with different datasets

	Average *support*		
Dataset	**G3PARM**	**NSGA-G3PARM**	**SPEA-G3PARM**
Autom	**0.79**	0.53	0.39
Credit	**0.87**	0.46	0.39
HH	**0.98**	0.44	0.30
MAR	**0.89**	0.50	0.42
MAROB	**0.93**	0.76	0.54
Mush	**0.97**	0.48	0.41
Soyb	**0.87**	0.45	0.40
WDBC	**0.99**	0.53	0.34
WDatBC	**0.75**	0.42	0.34
WPBC	**0.96**	0.36	0.35

(b) Average lift values obtained with different datasets

	Average *confidence*		
G3PARM	**NSGA-G3PARM**	**SPEA-G3PARM**	
1.04	2.11	**3.91**	
1.00	2.09	**3.65**	
1.00	**94.08**	80.39	
1.00	6.65	**11.37**	
1.01	1.99	**20.69**	
1.00	15.37	**26.95**	
1.02	3.59	**4.95**	
1.00	11.32	**49.38**	
1.05	2.30	**2.70**	
1.01	7.08	**12.08**	

(c) Average confidence values obtained with different datasets

	Average *confidence*		
Dataset	**G3PARM**	**NSGA-G3PARM**	**SPEA-G3PARM**
Autom	**0.99**	0.89	0.85
Credit	**0.99**	0.77	0.74
HH	**0.99**	0.80	0.75
MAR	**0.99**	0.82	0.74
MAROB	**0.99**	0.92	0.87
Mush	**0.99**	0.90	0.80
Soyb	**0.99**	0.91	0.90
WDBC	**1.00**	0.88	0.82
WDatBC	**0.96**	0.89	0.84
WPBC	**0.99**	0.84	0.82

(d) Average instances covered in different datasets

	Average Instances covered	
G3PARM	**NSGA-G3PARM**	**SPEA-G3PARM**
0.899	**1.000**	0.999
0.980	**0.998**	0.985
0.999	**0.999**	0.997
0.986	**1.000**	0.999
0.970	**1.000**	**1.000**
0.995	**0.999**	0.997
0.992	0.997	**0.999**
0.998	**1.000**	0.996
0.978	**1.000**	0.995
0.999	0.998	0.997

(e) Average number of rules obtained with different datasets

	Average number of rules		
Dataset	**G3PARM**	**NSGA-G3PARM**	**SPEA-G3PARM**
Autom	19.2	46.2	20.0
Credit	19.5	44.9	20.0
HH	19.9	48.9	20.0
MAR	15.6	42.8	20.0
MAROB	13.7	49.6	20.0
Mush	20.0	43.3	20.0
Soyb	20.0	32.2	20.0
WDBC	18.9	49.2	20.0
WDatBC	15.7	49.2	20.0
WPBC	19.8	46.2	20.0

Table 5.9: Average ranking for each algorithm and each measure

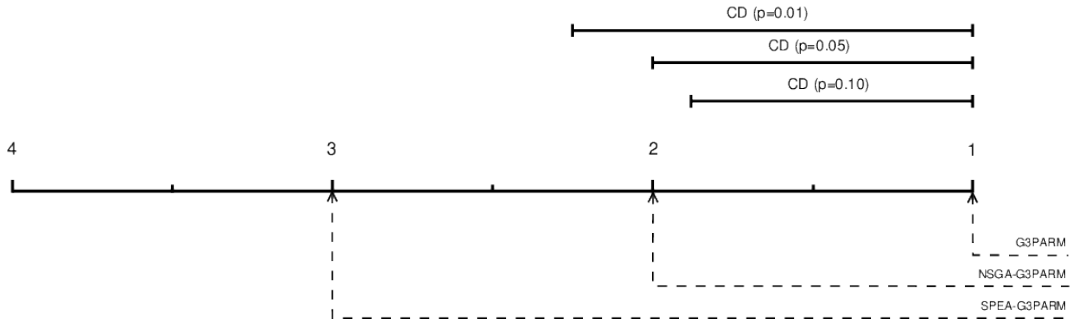
Measure	G3PARM	NSGA-G3PARM	SPEA-G3PARM
Support	**1.0**	2.0	3.0
Lift	3.0	1.9	**1.1**
Confidence	**1.0**	2.0	3.0
Coverage	2.6	**1.2**	2.1

that all algorithms perform equally well for these measures using $\alpha = 0.01$. In order to analyse whether there are any significant differences between the three algorithms, the Bonferroni-Dunn test is used to reveal the difference in performance (see Figure 5.3), where the critical difference (CD) value is 0.9 for $p = 0.1$; 1.0 for $p = 0.05$; and 1.2 for $p = 0.01$.

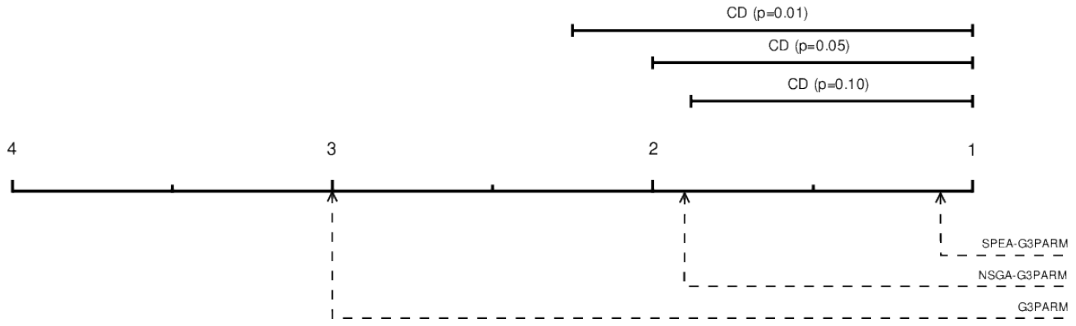
The results indicate that for support (see Figure 5.3(a)), at a probability of 99%, there are significant differences between G3PARM and SPEA-G3PARM, the performance of G3PARM being statistically better. On the other hand, at a significance level of $p = 0.05$, it is not possible to state that there are any significant differences between G3PARM and NSGA-G3PARM. However, there are significant differences between them at a probability of 90%, when the performance of the former is statistically better. Similarly, at a probability of 90%, there are significant differences between both multi-objective proposals and SPEA-G3PARM is found to be statistically better.

Focusing on lift (see Figure 5.3(b)), at a probability of 99%, there are significant differences between G3PARM and SPEA-G3PARM, the performance of the latter being statistically better. At a probability of 95%, there are significant differences between G3PARM and NSGA-G3PARM, the latter performing better in statistical terms. Finally, it is not possible to state that, at a significance level of $p = 0.1$, there are any significant differences between NSGA-G3PARM and SPEA-G3PARM, although the latter obtains the best ranking.

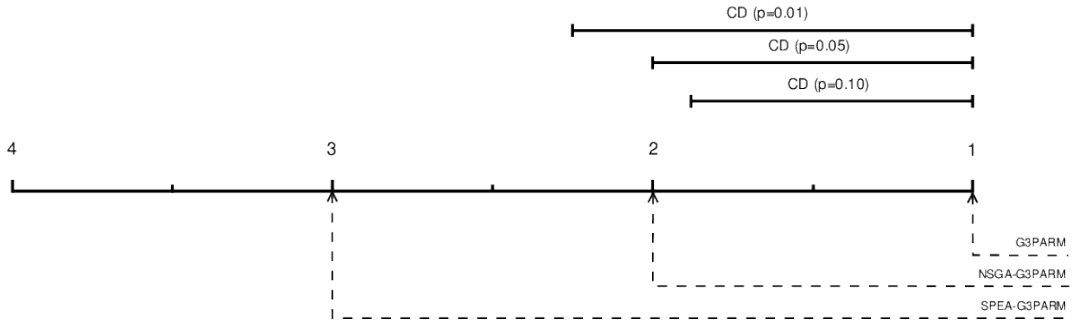
With regard to confidence (see Figure 5.3(c)), at a probability of 99%, there are significant differences between G3PARM and SPEA-G3PARM, the former performing statistically better. At a probability of 90%, there are significant differences between G3PARM and NSGA-G3PARM, and G3PARM continues to perform statistically better. Similarly, at a probability of 90%, there are significant differences



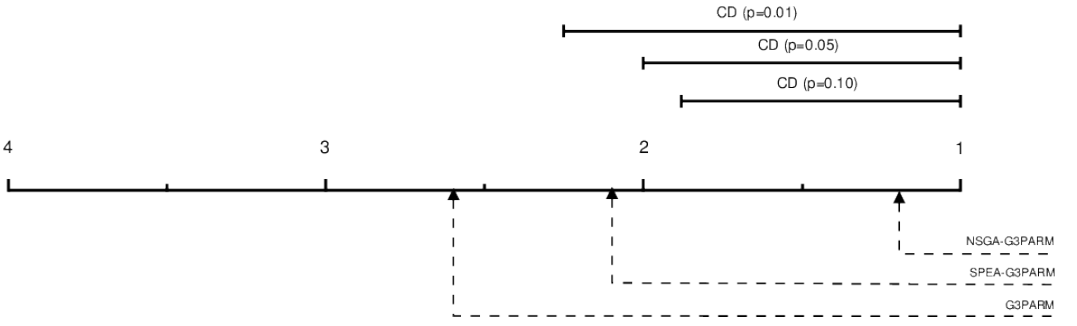
(a) Critical difference obtained with the Bonferroni-Dunn test for the support measure



(b) Critical difference obtained with the Bonferroni-Dunn test for the lift measure



(c) Critical difference obtained with the Bonferroni-Dunn test for the confidence measure



(d) Critical difference obtained with the Bonferroni-Dunn test for the coverage measure

Figure 5.3: Critical differences obtained with the Bonferroni-Dunn test for different measures when using support and lift as objectives to be optimized

between both multi-objective proposals but now NSGA-G3PARM performs better in statistical terms.

Finally, as shown in Figure 5.3(d) for the coverage measure, at a probability of 99%, there are significant differences between G3PARM and NSGA-G3PARM, the performance of the latter being statistically better. At a probability of 90%, there are significant differences between SPEA-G3PARM and NSGA-G3PARM and the latter continues to perform statistically better. However, it is not possible to state that, at a significance level of $p = 0.1$, there are any significant differences between G3PARM and SPEA-G3PARM, although the latter obtains the best ranking.

Concluding the analysis of using support–lift as objectives, multi-objective proposals are seen to perform very well for discovering rules of high interest. The discovery of interesting rules implies a decrease in the average support so G3PARM obtains a higher average support than multi-objective proposals when support and lift are used as objectives to be maximized. Despite the fact that the G3PARM algorithm generates very frequent and reliable rules, these rules are of slight interest.

In summary, the synergy of connecting G3P and multi-objective models for mining association rules provides important characteristics. The proposed multi-objective algorithms have demonstrate themselves to perform better than G3PARM when support and confidence measures are used together as the objectives to be optimized. For these two measures, the NSGA-G3PARM algorithm performs better than the others. As far as the coverage measure is concerned, SPEA-G3PARM obtains better results. On the contrary, if support and lift are jointly used as the objectives to be optimized, both multi-objective proposals behave better than G3PARM for the lift measure, although G3PARM obtains better results for both support and confidence. Finally, the NSGA-G3PARM algorithm always performs better than the others with regard to the coverage measure.

5.3 Conclusions

In this chapter, we have presented two novel G3P multi-objective approaches for mining ARs. These two models are based on two well-known multi-objective proposals: NSGA-2 and SPEA-2. The approaches have been properly analysed

and studied, optimizing both support-confidence and support-lift quality measures. These algorithms combine the strength of using G3P with the ability to search for good trade-offs between quality measures. The experimental study have demonstrated that the algorithm based on NSGA obtain better POFs than the one based on SPEA.

5.4 Publications Associated with this Chapter

- **International Journal:**

  Title: *Grammar-Based Multi-Objective Algorithms for Mining Association Rules* [84]

  Authors: J. M. Luna, J. R. Romero and S. Ventura

  Data Knowledge and Engineering, Vol. 86, pp. 19-37, 2013

  Ranking:

  Impact factor (JCR 2012): 1.519

  Knowledge area:

  Computer Science, Artificial Intelligence: 42/114

  Computer Science, Information Systems: 34/132

- **International Conference:**

  J.M. Luna, J.R. Romero and S. Ventura. *G3PARM: A Grammar Guided Genetic Programming Algorithm for Mining Association Rules*, Proceedings of the IEEE Congress on Evolutionary Computation, IEEE CEC 2010, pp. 2586-2593, ISBN: 978-1-4244-6910-8 [85]

6

Parameter-Free G3P Algorithms in Association Rule Mining

In this chapter, we present two parameter-free algorithms in the ARM field. First, a G3P algorithm that self-adapts its parameters along the generations is properly described. This algorithm adjusts its parameter values depending on the dataset under study, discovering not only frequent and strong ARs but also rules of interest based on the lift measure. Finally, an improved version that reduces the number of parameters and highly optimizes the numerical attributes is presented. This algorithm searches for rules whose instances distribution does not comprise blank spaces, since it is more interesting an AR having optimized quantitative attributes than an AR with a higher support value.

6.1 The G3PARM+ Algorithm

In this section, the G3PARM+ algorithm, a self-adaptive G3P algorithm, is proposed for mining ARs. This proposal also uses a CFG to define syntax constraints and extract rules in both numerical and categorical domains. During its evolutionary process, the proposal discovers ARs by applying two genetic operators over

those rules selected as parents by means of a niche-crowding [86, 87] model. Both genetic operators adjust their probabilities along the mining process. The proposed algorithm allows of obtaining ARs without any previous knowledge about the dataset, and does not require support or confidence thresholds. Moreover, a support–confidence framework is not purely accomplished, bringing the lift measure into play in order to extract frequent, reliable, and also interesting rules. The resulting set of rules comprises dissimilar rules, covering a high percentage of the dataset instances, which is measured by the coverage. It is noteworthy that for the sake of avoiding mismatched rules, the instances covered by each rule are analysed. The results clarify the good behaviour and efficiency of this proposal compared to G3PARM, which was previously contrasted to other exhaustive and genetic algorithms in Chapter 3. More specifically, the empirical comparison demonstrates that the new proposal obtains interesting rules with high support, high confidence, and high coverage of dataset instances.

In this way, this proposal mines ARs requiring only a few parameters and defines ARs by using derivation trees. Each derivation tree is evaluated by applying two criteria. One guides the search for solutions along the mining process. The other one allows of selecting the best solutions found. To mine new individuals in the evolutionary process, a parent selector using a niche-crowding methodology and two new genetic operators are presented.

6.1.1 Encoding

Similarly to any G3P proposal, each individual is defined by a genotype and a phenotype. The former is defined by means of a tree structure with different shapes and sizes. The latter represents the AR defined by the tree structure. This tree structure is obtained according to the definition of a grammar G, which is defined as shown in Figure 6.1.

The proposal here presented allows the use of both categorical and numerical attributes. Unlike previous proposals, which use $<$, $<=$, $>$ and $>=$ as logic operator for numerical attributes, this proposal applies the operators IN or OUT and randomly selecting two feasible values. For the sake of clarifying the individual representation, it is interesting to show a sample individual generated through the

G = $(\Sigma_N, \Sigma_T, P, S)$ with:

S = $\{Rule\}$

Σ_N = $\{Rule,\ Antecedent,\ Consequent,\ Condition,\ Categorical\ Operator,$
$Categorical\ Values,\ Numerical\ Values\}$

Σ_T = $\{$*'AND', '!=', '=', 'IN', 'OUT', 'CategoricalAttribute', 'CategoricalValue',*
'NumericalAttribute', 'NumericalValue' $\}$

P = $\{Rule = Antecedent,\ Consequent\ ;$
$Antecedent = Condition \mid$ *'AND'*, $Condition,\ Antecedent\ ;$
$Consequent = Condition \mid$ *'AND'*, $Condition,\ Consequent;$
$Condition = Categorical\ Operator,\ Categorical\ Value \mid$
$Numerical\ Operator,\ Numerical\ Value\ ;$
$Categorical\ Operator =$ *'!='* $\mid$ *'='* $;$
$Numerical\ Operator =$ *'IN'* $\mid$ *'OUT'* $;$
$Categorical\ Value =$ *'CategoricalAttribute' 'CategoricalValue';*
$Numerical\ Value =$ *'NumericalAttribute' 'NumericalValue' 'NumericalValue';* $\}$

Figure 6.1: Context-free grammar expressed in extended BNF notation

Table 6.1: A sample metadata

Attributes	**Values**
toy	*ball, teddy, doll*
toy's price	*[10, 50]*
sex	*male, female*

sample meta-data (see Table 6.1) and the application of a sequence of production rules from P, which would represent the following rule:

IF $toy'sprice\ IN\ [25, 43]\ \wedge\ toy = ball$ **THEN** $sex = male$

Notice that two random values are selected for the numerical conditions though the entire range of values could also be valid, e.g., the condition *'toy's price IN [10, 50]'*. Anyhow, this is not a problem since any individual having the antecedent or consequent with a maximum support produces a misleading rule, i.e., a lift value equal to unity, as explained in Section 2.1.

6.1.2 Evaluation

In this proposal, each individual I is evaluated according to two bounded range functions. The function F_{pool} uses a specific criterion to determine which individuals represent the best ARs. The other function, F, selects individuals whose structure is interesting for subsequent generations.

Notice that in the process of searching for the best rules, it is necessary to maximize the support, confidence, and lift. However, as shown in Figure 2.1 in Section 2.1.2, the confidence measure is maximized with the support measure, so we only need to maximize support and lift. Therefore, we propose a fitness function F_{pool} (see Equation 6.1) that maximizes support and lift and, in consequence, the three measures at once. F_{pool} is defined within the domain $[0, 1]$ and serves to determine the quality of the individuals in order to keep the best ones.

$$F_{pool}(I) = \begin{cases} support(I) \times lift(I) & \text{if } lift(I) > 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

On the other hand, it is essential to maintain the most promising rules in subsequent generations, so a different criterion is used for this. Firstly, it is interesting to obtain reliable rules, so those with high confidence values are desired. Secondly, it should be noted that very reliable rules could have low support values. Therefore, a function F to obtain reliable rules that do not appear infrequently is required (see Equation 6.2), allowing of evolving individuals during the mining process. The range of values of this new function is $[0, 1]$.

$$F(I) = confidence(I) \times F_{pool}(I) \tag{6.2}$$

6.1.3 Major Procedures

In this proposal, a new parent selector based on a niche-crowding method has been constructed. Moreover, two genetic operators that follow a new interesting methodology in ARM are detailed in depth.

Parents selector. The responsibility for obtaining a subset of individuals that act as parents in a given generation of the evolutionary process belongs to the selection operator, which is carried out in this proposal by implementing a niche-crowding [86, 87] model. This new model behaves like a clustering algorithm [88, 89], where individuals that satisfy similar instances belong to the same cluster. Thus, to determine the niche that each individual belongs to, the distance between two individuals (see Equation 6.3) is measured by comparing their instances covered, R_1 and R_2 being vectors having the instances covered by two rules, respectively.

$$distance(R_1, R_2) = 1 - \frac{R_1\ R_2}{||R_1||\ ||R_2||} \tag{6.3}$$

For a better understanding, consider the sample dataset shown in Table 6.2 and the instances covered by two sample rules, as depicted in Figure 6.2. In such a situation, the value of $R_1\ R_2$ is 2, whereas the values of $||R_1||$ and $||R_2||$ are $\sqrt{2}$ and $\sqrt{3}$, respectively. Once these values are obtained, the distance between these two individuals is defined by $distance(R_1, R_2) = 1 - (2/(\sqrt{2}\sqrt{3})) = 0.194$. In consequence, since the range of the values is $[0, 1]$, both individuals tend to be closer. In such a way, the closer are two individuals, the more similar these individuals are.

In an iterative way, this parent selector aims to form niches comprising those individuals that cover similar instances. In an initial step, a number of individuals are selected to act as centroids for each niche. Then, each of the remaining individuals

Table 6.2: Sample market customers

Customer	Bread	Milk	Coffee
Id-1	1	1	1
Id-2	0	1	0
Id-3	1	0	0
Id-4	1	0	0
Id-5	0	1	0
Id-6	1	1	1
Id-7	0	1	0
Id-8	1	0	0
Id-9	0	1	1
Id-10	0	0	1

Instance in the dataset:	1 2 3 4 5 6 7 8 9 10
R_1) Bread → Milk:	1 0 0 0 0 1 0 0 0 0
R_2) Milk → Coffee:	1 0 0 0 0 1 0 0 1 0

Figure 6.2: Sample association rules

is assigned to the niche whose centroid is closer. Once every individual is assigned to a specific niche, the sum of distances of every individual to its centroid within a niche is obtained ($Dist$). The goal is to minimize the $Dist$ value.

For each iteration, the first step consists in choosing new centroids that allow of obtaining the best results. The individual with the highest distance to its centroid is used as a new centroid, replacing the closer centroid. The process is carried out in an iterative way until the $Dist$ value can not be improved any more.

Once the iterations are finished, the parent selector operator randomly selects individuals from random niches each time. Thus, the aim is to cross individuals from different niches, which represent dissimilar rules.

Crossover operator. The crossover procedure requires a set of individuals to act as parents, returning a set of new individuals. During this procedure, this genetic operator gets iteratively two individuals and then, according to an initial crossover probability, a random value will determine if the operation will be executed or not. In G3PARM+, the crossover probability is updated in each generation. If the average lift obtained in the pool of individuals is greater than that calculated in the previous generation, then the crossover probability will be increased since a depth search or exploitation is required. On the other hand, if the average lift is less than or equal to that obtained in the previous generation, then the crossover probability should be decreased.

This operator obtains new individuals from parents previously selected by the parent selector. In this work, the crossover operator acts by swapping the most promising condition from each individual, i.e., the condition that allows further improvement of the support of the entire rule. Thereby, two offspring are obtained having the most promising conditions swapped. If two conditions are equally promising, the one with the lowest absolute support will be selected. Additionally, if two

conditions have the same absolute support, then one of them will be randomly selected.

For a better understanding, assume the rule $(Bread \wedge Milk) \rightarrow Coffee$ illustrated in Figure 6.3(a), which is based on the sample dataset depicted in Table 6.2. Once each condition is analysed, if the condition $Bread$ is changed (see Figure 6.3(b)), it would possible to obtain a new condition that allows of covering one more instance. On the other hand, no instance might be affected if $Milk$ (see Figure 6.3(c)) or $Coffee$ (see Figure 6.3(d)) are selected. Replacing one of these two conditions, the maximum support that it would possible to obtain is the same that the support obtained using the original conditions. Therefore, the most promising condition to be changed is $Bread$ since an absolute support of three could be obtained.

G3PARM addresses the issue of obtaining new individuals by changing the condition with the lowest support value within a rule. Sometimes, this methodology is not the best choice, as depicted in Figure 6.3. If the condition with the lowest support of the sample rule is changed, i.e., the condition $Coffee$, which has a support value of 0.4, then the entire rule could not be improved. Changing this condition, the support of the rule remains the same or even lower. However, using the crossover operator presented in this work, the best condition to be replaced is $Bread$, which is not the lowest support condition, but the most promising condition. If this condition is altered, the support of the rule could be improved in one instance, so a support value of 0.3 could be obtained at most.

Mutation operator. The mutation genetic operator is responsible for obtaining a new individual from another one previously selected. This operator also requires an initial mutation probability, which is responsible for determining if certain individual will be mutated or not. This genetic operator modifies the most promising condition from a parent to obtain a better one. Similarly to the crossover probability, the mutation probability is updated in each generation. If the average lift obtained in the pool of individuals is greater than that calculated in the previous generation, then the mutation probability decreases. On the other hand, if the average lift is less than or equal to that obtained in the previous generation, then the mutation probability increases since a breadth search or exploration is required.

Instances in the dataset:	1 2 3 4 5 6 7 8 9 10
Bread:	1 0 1 1 0 1 0 1 0 0
Milk:	1 1 0 0 1 1 1 0 1 0
Coffee:	1 0 0 0 0 1 0 0 1 1
(Bread ∧ Milk) → Coffee:	1 0 0 0 0 1 0 0 0 0

(a) Instances covered by a sample rule

Instances in the dataset:	1 2 3 4 5 6 7 8 9 10
Bread:	1 0 1 1 0 1 0 1 **0** 0
Milk:	1 1 0 0 1 1 1 0 1 0
Coffee:	1 0 0 0 0 1 0 0 1 1
Bread:	0 0 0 0 0 0 0 0 **1** 0

(b) Condition *Bread* could improve the rule at most in one instance

Instances in the dataset:	1 2 3 4 5 6 7 8 9 10
Bread:	1 0 1 1 0 1 0 1 0 0
Milk:	1 1 0 0 1 1 1 0 1 0
Coffee:	1 0 0 0 0 1 0 0 1 1
Milk:	0 0 0 0 0 0 0 0 0 0

(c) Condition *Milk* could not improve the rule

Instances in the dataset:	1 2 3 4 5 6 7 8 9 10
Bread:	1 0 1 1 0 1 0 1 0 0
Milk:	1 1 0 0 1 1 1 0 1 0
Coffee:	1 0 0 0 0 1 0 0 1 1
Coffee:	0 0 0 0 0 0 0 0 0 0

(d) Condition *Coffee* could not improve the rule

Figure 6.3: Instances covered by each condition

Additionally, a procedure to improve the reliability of the rules is also implemented in this operator. As mentioned in Section 2.1, the confidence of an AR is maximum if and only if the set of instances covered by the antecedent of a rule is a subset of that covered by the consequent. Bearing in mind this, and considering that

the support and the lift measures are symmetrical, then the procedure here implemented swaps the antecedent and consequent of a rule if its consequent support value is lower than its antecedent support value. In such a way, both the support and the lift remain the same, while the confidence will improve.

Update process. G3PARM+ keeps the best individuals in a pool, i.e., those with a high quality, maintaining a good spread of solutions. An updating of this pool is required in each generation, adding those new individuals that exceed a quality level —using the fitness F_{pool}— and removing the worst ones. Individuals are ranked in a list and then, the pool is filled with the best ones using an iteratively procedure. This procedure avoids to keep individuals that cover the same instances, so no equivalent individuals are mined.

The pseudo-code of the complete updating process is shown in Algorithm 8. In this procedure, the set $auxSet$ is initialized to the set $currentPopulation$, which comprises the individuals of the current population. Then, this set is filled with those new individuals within the sets $newPopulation$ and $pool$ (line 1). While the former is obtained using the crossover and mutation operators, the latter represents the pool of individuals belonging to the last generation. It should be noted that no repeated individual is added to the set $auxSet$. Finally, this set is ranked according to the function F_{pool} (line 2).

Once the set of individuals is obtained and ranked, the update of the new pool of individuals is carried out (lines 5 to 33). At the beginning, the best individual from the ranked set $auxSet$ is added to the pool $newPool$, which serves as the starting individual when the pool is empty. Then, for every individual $individualAux$ in the set $auxSet$, the algorithm checks whether there exists any rule in the set $newPool$ that already covers the same instances (line 10). If there is such an individual and, furthermore, their lift values are equal, then the one with the higher confidence value is kept in this pool (lines 11 to 15). However, if both have the same confidence value, the one with the lowest number of conditions is added for comprehensibility purposes (lines 17 to 20). In this way, the proposal searches for small rules with a high confidence. Finally, the set of individuals $newPool$ is returned.

It should be noted that, since G3PARM+ updates the pool of individuals by analysing the instances covered by each new rule, the resulting set of rules will

be very representative of the domain under study and, therefore, the resulting set obtains a high coverage value.

Algorithm 8 Update process

Require: *pool*, *currentPopulation*, *newPopulation*, *poolSize*
Ensure: *newPool*

```
1: auxSet ← auxSet ∪ obtainNotRepeatedIndividuals(
   currentPopulation, newPopulation, pool)
2: auxSet ← rankIndividuals(auxSet,F_pool)
3: i ← 0
4: newPool ← ∅
5: for all individualAux ∈ auxSet do
6:    if getSize(newPool) = 0 then
7:       newPool ← individualAux
8:    else
9:       for all individualPool ∈ newPool do
10:         if instancesCovered(individualAux) =
            instancesCovered(individualPool) then
11:            if calculateLift(individualAux) =
               calculateLift(individualPool) then
12:               if calculateConfidence(individualAux) >
                  calculateConfidence(individualPool) then
13:                  newPool ← newPool\{individualPool}
14:                  newPool ← newPool ∪ individualAux
15:                  break
16:               else
17:                  if calculateConfidence(individualAux) =
                     calculateConfidence(individualPool) AND
                     calculateConditions(individualAux) <
                     calculateConditions(individualPool) then
18:                     newPool ← newPool\{individualPool}
19:                     newPool ← newPool ∪ individualAux
20:                     break
21:                  end if
22:               end if
23:            end if
24:         else
25:            newPool ← newPool ∪ individualAux
26:            break
27:         end if
28:      end for
29:   end if
30:   if getSize(newPool) = poolSize then
31:      break
32:   end if
33: end for
34: return newPool
```

6.1.4 Algorithm

The algorithm proposed (see Figure 6.4 for a general sketch) uses two populations with a prefixed size: one to maintain the most promising individuals obtained throughout the generations, and the other to keep those individuals that should not be lost because of their high-quality. The algorithm starts by randomly generating individuals conformant to the CFG defined in Figure 6.1. Those individuals that cover at least one instance are added to the population, so it comprises only rules with a support greater than zero. It is an iteratively process, adding individuals until the pool size is reached. On the other hand, and in order to obtain new individuals in each generation, a set of parents is chosen from the current population and the pool of individuals. Since the pool of individuals is empty in the first generation, parents are chosen only from the current population. The parent selector procedure allows of selecting individuals from different niches, providing a set of individuals from the application of the mutation and crossover operators.

After applying these operators, a new set of individuals is obtained and added to a new population, also comprising the current population and the pool of individuals. It is worth mentioning that this approach avoids the repetition of individuals in this new population. At this point, the algorithm is divided into two paths, none being dependent on the other. The easiest path is responsible for updating the next population of the evolutionary process. With this purpose, the algorithm ranks the individuals within the new population previously obtained. This ranking procedure is carried out by using the F function described above, so the best individuals are selected to be part of the new population. On the contrary, the second path is responsible for updating the pool of individuals by running the procedure described in Listing 8. In this case, the procedure ranks the individuals according to the F_{pool} function, being selected the best ones as described in Section 3.4. Once this procedure is completed, the pool of individuals is checked. If the average lift of this set of individuals is greater than that calculated in the previous generation, it means that the population is improving, so a lower diversity is required. In such a situation, the crossover probability increases and the mutation probability decreases. On the other hand, if the average lift is less than or equal to that obtained in the previous generation, a higher diversity is necessary, so the crossover probability decreases and the mutation probability increases.

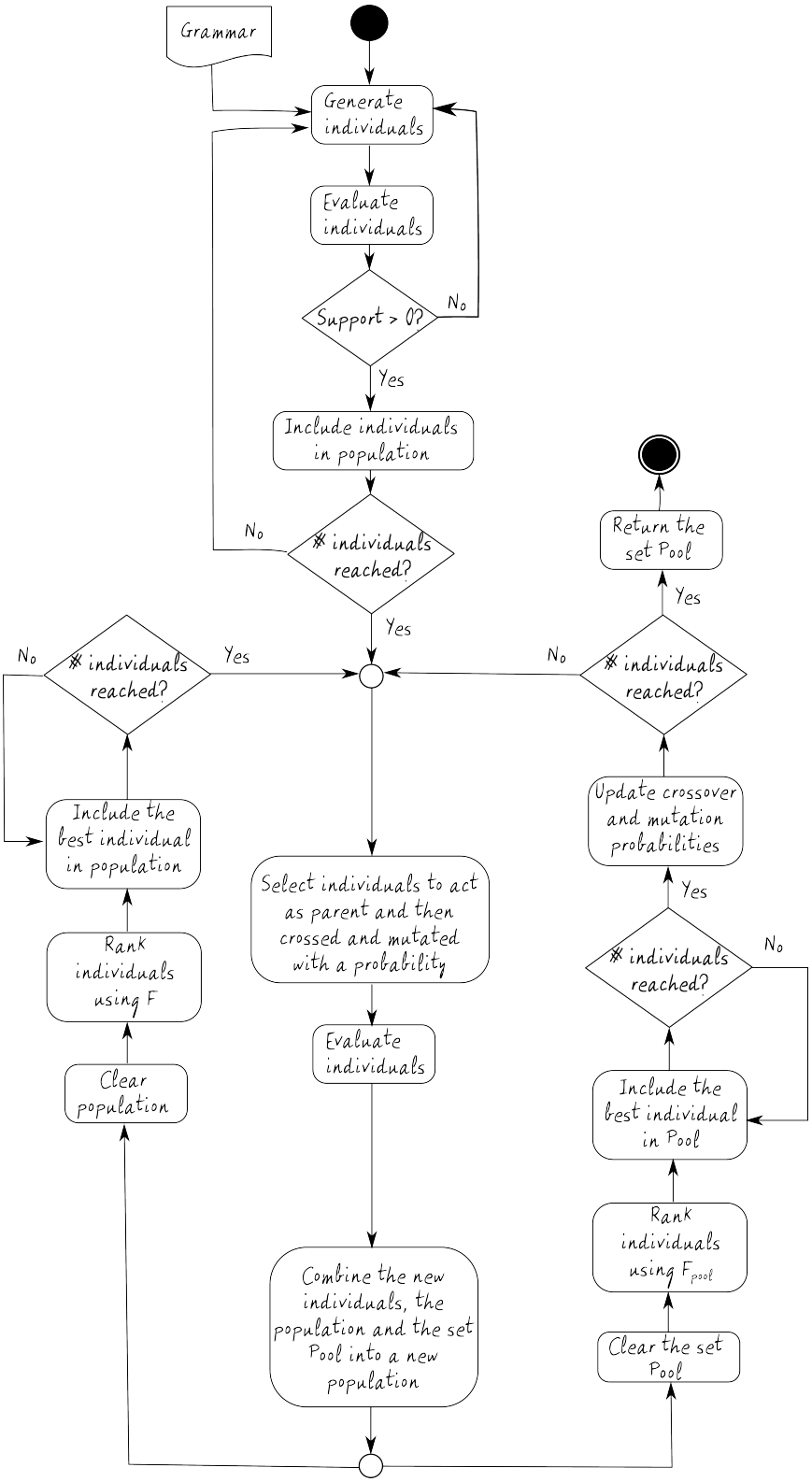


Figure 6.4: The flowchart for the proposed algorithm

Finally, unlike other existing evolutionary ARM algorithms, an important feature of this algorithm is that the number of parameters required is low. Specifically, the continuous updating of the crossover and mutation probabilities mitigates the difficulty of using this algorithm by non-expert users. Additionally, G3PARM+ does not require establishing minimum thresholds, hence it does not require a previous knowledge of the domain under application.

6.1.5 Datasets

In this work, several experiments were carried out, using a variety of 17 datasets, which were obtained from the UCI machine learning repository with different numbers of instances and attributes (see Table 6.3), to demonstrate the effectiveness of the proposal presented. G3PARM was originally compared with other exhaustive search (Apriori and FP-Growth) and evolutionary algorithms (ARMGA and QuantMiner) in Chapter 3, providing great results. At a significance level of $p = 0.05$, G3PARM was significantly different from exhaustive search algorithms in support and confidence measures. Focusing on evolutionary algorithms and the

Table 6.3: Datasets

Dataset	**Abbreviation**	**#Instances**	**#Attributes**	**Type**
Automobile	*Autom*	392	8	Num.
Ankara Weather	*Ankara*	1608	10	Num.
Chess	*Chess*	3196	37	Categ.
Connect-4	*Con4*	7557	43	Categ.
Credit German	*Credit*	1000	21	Categ., Num.
House_16H	*HH*	22784	17	Num.
Izmir Weather	*Izmir*	1461	10	Num.
Low Birth Weight	*LBW*	189	10	Categ., Num.
Mushroom	*Mush*	8124	23	Categ.
Nursery	*Nurs*	12960	9	Categ.
Primary Tumor	*Prim*	339	18	Categ.
Soybean	*Soyb*	683	36	Categ.
Splice	*Splice*	3190	61	Categ.
Treasury	*Treas*	1049	16	Categ.
Wisc. Diagn. Cancer	*WDBC*	569	31	Categ., Num.
Wisc. Progn. Cancer	*WPBC*	194	34	Categ., Num.
Zoo	*Zoo*	102	17	Categ.

support measure, at a significance level of $p = 0.05$, G3PARM was significantly different from the other algorithms. On the other hand, studying the confidence measure, G3PARM was significantly different from QuantMiner, but not from ARMGA. Thus, a comparison of the new proposal versus G3PARM is made in this experimental stage. To make a fair comparison the lift measure is not considered as measure to be compared since G3PARM does not optimize this measure.

All the experiments carried out were performed on an Intel Core i7 machine with 12GB main memory and running CentOS 5.4. Moreover, the proposal here presented was written by using JCLEC [74].

In the following subsections, first the experimental set-up is presented. Then, a study of the adaptiveness of the crossover and mutation probabilities is examined, and the results obtained are discussed. Next, an analysis of the influence of varying the number of generations is presented. Finally, a series of experiments serve to contrast the scalability of the algorithms as well as how they behave on increasing the number of attributes and the number of instances in the dataset.

6.1.6 Experimental set-up

The evolutionary proposal here presented only requires four parameters to be determined by the expert: the population size, the maximum number of generations, the maximum derivation size, and the pool size. To achieve a fair comparison with the G3PARM algorithm, their values remain the same as those provided in Chapter 3. These optimal parameters are shown in Table 6.4, where the best results for G3PARM were obtained using a population size of 50 individuals, 100 generations, 70% crossover probability, 14% mutation probability, and a maximum derivation size of 24.

The results shown for each algorithm in the following experimental study are the average values calculated running each one 30 times with different seeds.

6.1.7 Analysis of Genetic Operator Probabilities

A major feature of G3PARM+ is its ability to update both genetic operator probabilities. To show this, Figure 6.5 lists the crossover and mutation probability values

Table 6.4: Optimal parameters

Parameter	G3PARM+	G3PARM
Population size	50	50
Pool	20	20
Number of generations	100	100
Maximum derivation size	24	24
Crossover probability	-	70%
Mutation probability	-	14%
Confidence threshold	-	90%
Support threshold	-	70%

in each generation for a sample execution. Notice that in the first generation both measures have the initial value 0.5.

In subsequent generations, these probabilities are updated. It is interesting to note that in the earliest generations, both probabilities increase and decrease while the optimal average lift value is not found. Then, the mutation probability begins to increase whereas the crossover probability decreases. This means that the average lift is not improving and, therefore, the optimal value is obtained. Nevertheless, despite the average lift value not being improved, it is possible to obtain other

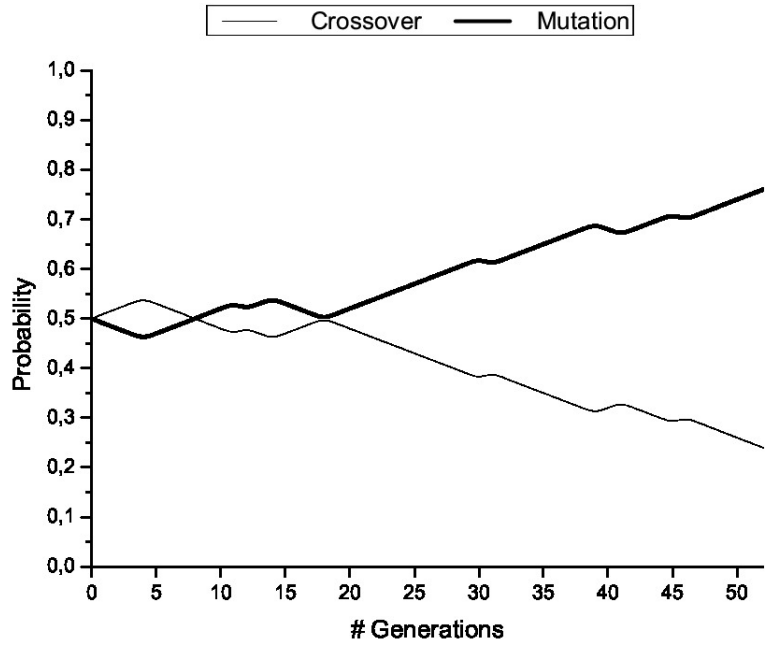


Figure 6.5: Crossover and mutation probabilities along the generations

Table 6.5: Average results obtained for mushroom and nursery datasets

	Mushroom dataset		Nursery dataset	
Measures	**G3PARM+**	**G3PARM**	**G3PARM+**	**G3PARM**
Support	0.991	1.000	0.663	0.759
Confidence	0.999	1.000	0.971	0.990
Lift	1.005	1.000	1.105	1.003
#Rules	20.000	20.000	20.000	2.967

different and interesting rules, so the evolutionary process must go on, at least for a certain number of generations.

6.1.8 Analysis of the Experiments

As discussed in Section 2.1, the use of a support–confidence framework could cause unexpected values in some domains, so different measures are required to mine interesting rules. To clarify this assertion, the mushroom and nursery datasets were used and the average results obtained are shown in Table 6.5, including the lift measure to demonstrate that other measures are required. Using the mushroom dataset, G3PARM mined a set of 20 rules and every rule had a maximal support value. As expected, its confidence value is maximal too. Furthermore, maximal support values provide misleading rules, so their lift values are equal to unity. This clarifies that the goal of ARM is not only the extraction of frequent and reliable rules, but interesting rules are required too.

On the other hand, attending to the nursery dataset, the average number of rules mined by executing the G3PARM algorithm 30 times is 2.967. Every execution was performed while setting the thresholds to 70% and 90% for the support and confidence measures, respectively. Moreover, the size of the population to be returned was set to 20. As mentioned above, the use of these thresholds requires an exhaustive knowledge of the dataset used, and the results obtained for the nursery dataset is a clear example.

Globally, the average results of the experiments performed over different datasets are shown in Table 6.6, where the best results for each measure are written in bold typeface. To make a fair comparison the lift measure is not considered as measure to be compared since G3PARM does not optimize this measure. Notice that the two

Table 6.6: Average results obtained for different datasets

(a) Average support values obtained with different datasets

Dataset	G3PARM+	G3PARM
Ankara	**0.987**	0.940
Autom	**0.958**	0.862
Chess	**0.947**	0.898
Con4	**0.952**	0.946
Credit	**0.966**	0.925
HH	**0.969**	0.958
Izmir	**0.989**	0.958
LBW	0.929	**0.944**
Prim	**0.917**	0.836
Soyb	0.901	**0.925**
Splice	0.984	**0.999**
Treas	**0.944**	0.894
WDBC	**0.983**	0.982
WPBC	**0.980**	0.952
Zoo	0.667	**0.780**

(b) Average confidence values obtained with different datasets

Dataset	G3PARM+	G3PARM
Ankara	0.995	**0.999**
Autom	**0.988**	**0.988**
Chess	0.993	**1.000**
Con4	0.998	**1.000**
Credit	0.987	**0.999**
HH	**1.000**	**1.000**
Izmir	0.996	**0.999**
LBW	0.981	**0.997**
Prim	0.976	**0.994**
Soyb	0.995	**0.999**
Splice	**1.000**	**1.000**
Treas	0.990	**1.000**
WDBC	0.997	**1.000**
WPBC	0.995	**1.000**
Zoo	**0.976**	0.973

(c) Average coverage values obtained with different datasets

Dataset	G3PARM+	G3PARM
Ankara	**1.000**	**1.000**
Autom	**1.000**	0.989
Chess	**1.000**	0.999
Con4	**1.000**	**1.000**
Credit	**1.000**	**1.000**
HH	**1.000**	**1.000**
Izmir	**1.000**	**1.000**
LBW	0.998	**1.000**
Prim	0.999	**1.000**
Soyb	**1.000**	0.999
Splice	**1.000**	**1.000**
Treas	**0.993**	0.961
WDBC	**1.000**	0.999
WPBC	0.998	**1.000**
Zoo	**1.000**	0.995

datasets mentioned above were not considered, due to their problems. Studying the support measure values, which are shown in Table 6.6(a), it is worth mentioning

that the new proposal mines more frequent rules than G3PARM, obtaining an average difference value of 0.023.

Focusing on the confidence measure (see Table 6.6(b)), the results show that using G3PARM, the average confidence values obtained for each dataset are greater than those acquired using the new proposal. Although G3PARM+ does not provide the best confidence values, the average difference between both algorithms is very low, just a value of 0.005, discovering very reliable rules too.

Finally, Table 6.6(c) shows the average coverage measure for each dataset and algorithm. This measure represents the percentage (in a per unit basis) of instances covered by the resulting set of rules. Focusing on this measure, note that while the new proposal wins in 12 out of 15 datasets, the G3PARM algorithm wins in 9 out of 15 datasets. The average results of all the datasets demonstrate that using this measure, G3PARM+ obtains better results than G3PARM, calculating a difference value of 0.003.

Table 6.7 shows the average number of attributes per rule for the resulting set and the average amplitude of the numerical attributes for this set, respectively.

Table 6.7: Average number of attributes and amplitude obtained for different datasets

(a) Average number of attributes

Dataset	G3PARM	G3PARM+
Ankara	**2.82**	2.91
Autom	2.78	**2.49**
Chess	**2.99**	3.01
Con4	3.63	**3.05**
Credit	**2.59**	2.81
HH	**2.80**	3.16
Izmir	**2.81**	3.14
LBW	**2.60**	3.01
Prim	2.75	**2.26**
Soyb	3.05	**2.65**
Splice	3.14	**2.42**
Treas	2.83	**2.72**
WDBC	2.95	**2.80**
WPBC	**2.87**	2.88
Zoo	**3.07**	3.11

(b) Average amplitude (in percentage)

Dataset	G3PARM+	G3PARM
Ankara	81.56	**76.06**
Autom	84.98	**75.26**
Chess	-	-
Con4	-	-
Credit	**77.95**	78.15
HH	81.48	**70.73**
Izmir	89.71	**73.19**
LBW	**79.15**	79.83
Prim	-	-
Soyb	-	-
Splice	-	-
Treas	-	-
WDBC	79.11	**59.27**
WPBC	84.55	**75.26**
Zoo	-	-

For the second value, only numerical datasets have been considered. Notice that there is no difference in the average number of attributes for both algorithms. This number of attributes is not constrained a priori. As for the amplitude, G3PARM divides the range of values into a number of equal-width points and uses these cut-points as values for the numerical attributes. On the contrary, G3PARM+ does not require any number of equal-width cut-points, so ranges of any amplitude could be obtained. For example, using 5 equal-width cut-points, G3PARM could obtain an amplitude of value 80% at most, whereas G3PARM+ is not restricted in this sense.

Although this analysis is based on the average values of several measures over datasets that have different features, a more accurate statistical analysis is required. With this purpose, the Wilcoxon signed rank test [76] (see Table 6.8 where the sum of all positive ranks (W+) and all negative ranks (W-) are depicted) has been performed, obtaining a p-value of 0.041 for the support measure, so it is possible to assert that there are significant differences between G3PARM and G3PARM+ using a significance level of $\alpha = 0.05$, the latter being statistically better. Obtaining the sum of all ranks, it is stated that the sum of all positive ranks is 96, whereas the sum of all negative ranks is 24. Continuing with the analysis, a p-value of 0.002 is calculated for the confidence measure. Here, the sum of all positive ranks is 3, whereas the sum of all negative ranks is 75. Using a significance level of $\alpha = 0.05$, we can state that G3PARM is statistically better for the confidence measure with significant differences.

Focusing on the coverage measure, it is not possible to assert that there are significant differences between both algorithms using this measure, obtaining a p-value of 0.301. Under these circumstances, it is necessary to determine which one is statistically better, so the sum of all positive and negative ranks are calculated. The

Table 6.8: Wilcoxon signed rank test results obtained for different measures

Measure	p-value	W+	W-
Support	0.041	96	24
Confidence	0.002	3	75
Coverage	0.301	31.5	13.5
#Attributes	0.248	72	48
Amplitude	0.018	3	33

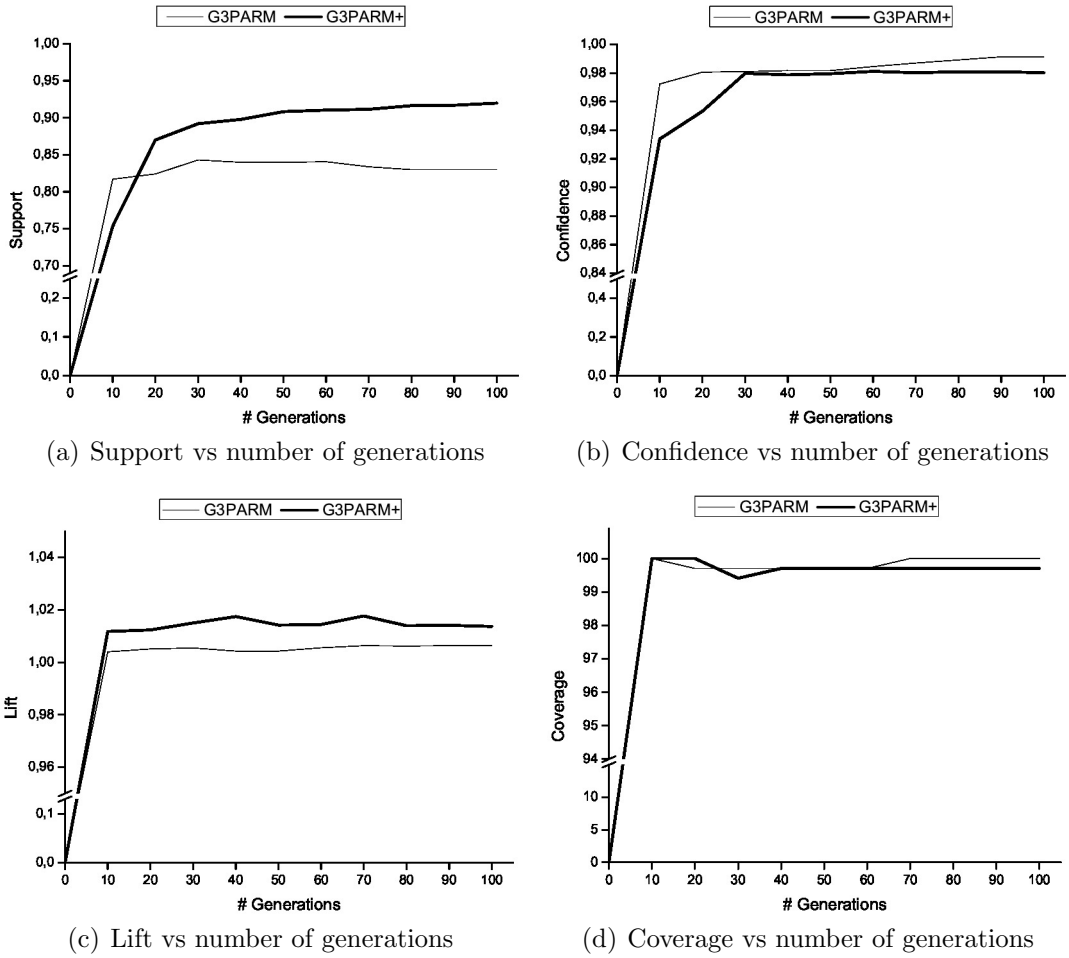
results show that the positive ranks sum to 31.5 whereas the negative ranks sum to 13.5. In consequence, it is possible to assert that, despite the fact that there are no significant differences between the proposals, G3PARM+ is statistically better. As for the number of attributes obtained, this test reveals a p-value of 0.248, so it is not possible to assert that there are significant differences between both algorithms, even when G3PARM+ behaves better considering that the sum of all positive ranks is 72. Finally, regarding the amplitude of the numerical intervals, a p-value of 0.018 is obtained, stating that there are significant differences when a significance level of $\alpha = 0.05$ is calculated, G3PARM being statistically better.

This statistical analysis shows that G3PARM+ behaves better in four out of six measures, obtaining differences in support value for a significance level of $\alpha = 0.05$. As for the coverage measure, G3PARM+ behaves better than G3PARM. Finally, G3PARM obtains the best confidence values. Noted that, unlike G3PARM+, G3PARM mines rules with only one attribute in the consequent, limiting the type of rules which can be discovered. Since the consequent is composed of only one condition, the support of the consequent can be higher easily and, therefore, the confidence values are likely to be higher. Finally, it is interesting to point out that the number of attributes obtained by G3PARM+ is slightly lower than by G3PARM, as expected after the analysis of the support values obtained. Notice that the higher is the support value of a rule, the lower is its number of attributes.

To sum up, the results demonstrate that the new proposal does not behave statistically worse than G3PARM, obtaining better results in some quality measures. Additionally, G3PARM+ does not require a previous step to determine the optimal parameter, so it is a high advantage for non-expert users. Finally, G3PARM+ considers the lift measure, only obtaining interesting rules unlike G3PARM, which maximizes support and confidence without consider the interest of the rules mined.

6.1.9 Analysis of the Number of Generations

In order to observe the effect of varying the number of generations on the proposed algorithm, a number of experiments were carried out over the *Primary Tumor* dataset. Here, the behaviour of G3PARM and G3PARM+ are described for support, confidence, lift and coverage measures (see Figure 6.6).



(a) Support vs number of generations

(b) Confidence vs number of generations

(c) Lift vs number of generations

(d) Coverage vs number of generations

Figure 6.6: Influence of the variation of the number of generations in different measures

Focusing on the support measure, whereas G3PARM+ increases its values when the number of generations grows, G3PARM comes to a standstill in early generations, as shown in Figure 6.6(a). In such a situation, the support can not be improved anymore, so the average confidence grows, as depicted in Figure 6.6(b). On the other hand, G3PARM+ allows of increasing the average support in every generation. Focusing on the average confidence value, the new model allows of obtaining almost the same average values as those obtained with G3PARM. Therefore, a global analysis of the average support and confidence measures reveals that the behaviour of the new model is better.

Figure 6.6(c) shows the average lift values obtained in different generations. Whereas G3PARM+ obtains the best average lift values, G3PARM discovers rules whose

lift values are close to unity. Finally, analysing the coverage measure (see Figure 6.6(d)), both G3PARM and the new algorithm obtained values close to 100% in terms of the percentage of instances covered by the resulting set.

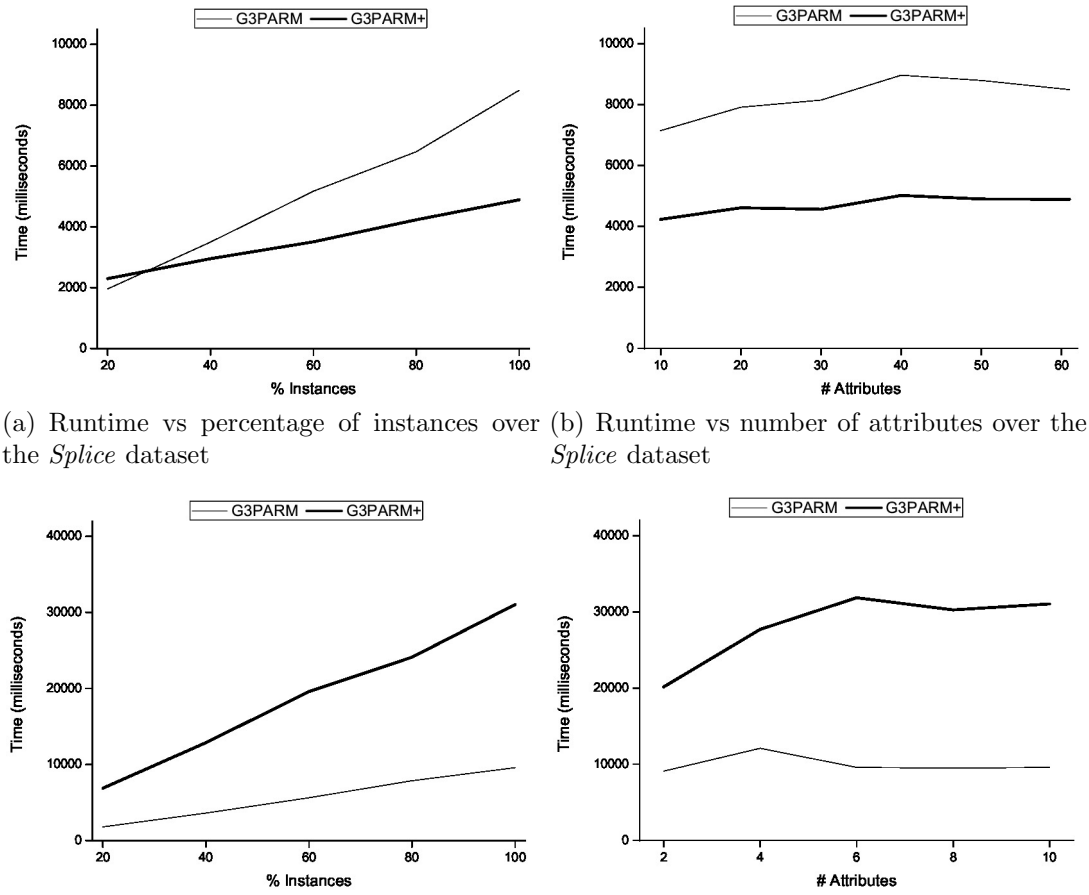
6.1.10 Analysis of Scalability

Since G3PARM+ uses two different functions to evaluate each measure and a specific analysis to group the individuals in niches, it is possible to think that the computation time has increased significantly in comparison to G3PARM. However, this situation does not occurr as shown in the following analysis.

A number of experiments were also carried out to analyse the computation time of G3PARM and G3PARM+ over two different datasets, as depicted in Figure 6.7. The Y-axis represents the time in milliseconds, whereas the X-axis stands for the percentage of instances (% Instances) using all attributes or the number of attributes (# Attributes) using all the instances.

Figures 6.7(a) and 6.7(b) show the runtimes for the *Splice* dataset, all its attributes being categorical. Focusing on Figure 6.7(a), it should be noted that G3PARM+ scales better than G3PARM, which seems to increase exponentially. On the other hand, analysing the results plotted in Figure 6.7(b), note that both algorithms behave similarly, G3PARM+ expending a lower computational time in this case. However, studying the behaviour of a different dataset, i.e., *Ankara*, which defines all of its attributes in a numerical domain, it is seen that G3PARM+ scales worse than G3PARM, as shown in Figures 6.7(c) and 6.7(d).

Apparently, G3PARM+ scales better over categorical than over numerical attributes, as expected. Nevertheless, the use of the operators IN and OUT requires a more complex evaluation since they require analysing whether a value is within a range or not. On the other hand, note that in G3PARM these operators are not defined and the evaluation process only requires checking whether a value is less than or greater than the other. An important advantage of the operators IN and OUT is their interpretability by the data miner, giving intervals that are difficult to be obtained using traditional numerical operators.



(a) Runtime vs percentage of instances over the *Splice* dataset

(b) Runtime vs number of attributes over the *Splice* dataset

(c) Runtime vs percentage of instances over the *Ankara* dataset

(d) Runtime vs number of attributes over the *Ankara* dataset

Figure 6.7: Relation between the runtime and the dataset size

6.2 Mining Highly Optimized Quantitative Attributes

In addition to the task of searching for the optimal parameter values, which could be a handicap for non-expert users in evolutionary computation and unsuccessful results could be obtained, there is a problem related to the continuous values themselves. Existing algorithms do not carry out an accurate optimisation of such values, meaning that some of the discovered quantitative association rules are lacking in interest. Without the optimisation of the continuous values, a huge number of extracted rules could comprise an unnecessary range of values. The discovery

of rules comprising patterns with a highly representative range of values is of high interest, despite the fact that the quality measures of these rules decrease slightly.

In this section, we propose a solution to this problem. To do so, we suggest a free-parameter algorithm that self-adapts to the required parameters (only requiring the number of rules to be extracted), meaning that it is highly useful for users with no experience in evolutionary computation. We also propose a heuristic approach to search for highly representative numerical patterns. Even when the problem could be tackled in many different ways, we suggest the use of a grammar-guided genetic programming (G3P) model [58], which has achieved excellent results in unsupervised learning tasks [53].

Finally, in order to demonstrate the performance and usefulness of the proposed model, a series of experiments were carried out, which are fully described in the experimental section. Both categorical and continuous domains were therefore used, demonstrating that the proposed model discovers highly representative rules.

6.2.1 Main idea behind the proposal

The aim of the proposed approach is twofold. First, the mining of optimised quantitative association rules in an efficient way. Second, designing a proposal which does not require as many parameters as other existing evolutionary proposals, and also presenting the advantages, both of using evolutionary methodology and having a grammar to represent solutions to the problem under study.

The proposed approach includes a process which accordingly evaluates the rules discovered by considering the optimisation of the continuous patterns. The support and confidence of each rule is not therefore the only point of interest, but also the distribution of instances satisfied by the rule. The aim of mining highly optimised quantitative association rules is to select the right amount of values to contain as few gaps as possible. We consider gaps to be spaces that do not comprise any instance. Let us take by way of an example the range $[A, B]$ of values for a sample numerical feature (see Figure 6.8) that comprises 15 instances distributed within this range of values. Two sample intervals could be obtained, for example, X and Y, comprising 10 and 13 instances, respectively. Analysing the support of each interval, Y seems to be more interesting than X. However, when analysing the

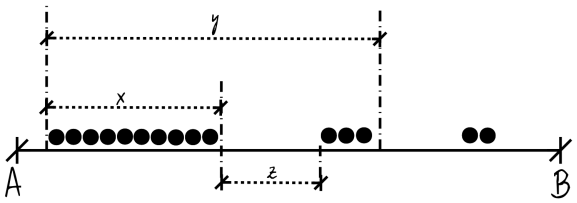


Figure 6.8: Sample range of values

distribution of instances within the interval, X is of high interest as its instances are uniformly distributed. On the contrary, Y comprises a gap Z, so its instances are not as well distributed, as in X.

This specific process, together with other processes that form the whole algorithm, starts by generating an initial set of individuals which conform to a context-free grammar. This initial set of individuals, or population, depends on the number of rules to be mined.

The aim of the proposed algorithm is to obtain the best n rules according to a number of quality measures. Therefore, a pool of individuals with a predefined size of n is used, this pool working as an elitist population to maintain the best n rules across the generations. In each generation, this elitist population is updated with the best individuals. This means that individuals included in the population are ranked according to their fitness function values, and the best individuals are kept for new generations. This fitness value determines how close a given solution comes to meeting the aim.

For the sake of generating new individuals in each generation of the evolutionary process, a genetic operator, described in subsequent sections, is applied. This genetic operator is used based on a probability. Most evolutionary algorithms require fixed values, meaning that the optimal probability values are determined by the data miner, which in turn in based on the dataset used. A major feature of the G3P free-parameter algorithm presented in this work is its ability to update the genetic operator probability, not requiring any previous study of the parameters to obtain optimal results, as most evolutionary algorithms do. In the first generation, the genetic operator is applied using an initial value, whereas in subsequent generations, this probability is updated according to the average fitness value obtained in the aforementioned elite population. The proposed algorithm continues its iterative

process without requiring a maximum number of generations. On the contrary, the evolutionary process is carried out while the elite population improves with the passing of the generations. This improvement is measured using the average fitness function values of the n best rules.

Finally, if the quality of the rules mined does not improve with the passing of the generations and the maximum genetic probability is reached, then the algorithm finishes and the best rules discovered in the evolutionary process are returned to the data miner.

6.2.2 Encoding

Similarly to the models described in previous chapters, the proposed model is based on the use of a CFG, which defines all of the possible solutions that could be obtained.

The properties of a CFG has been properly described in Chapters 2 and 3, where a grammar was formally defined as $G = (\Sigma_N, \Sigma_T, P, S)$. Once the grammar is defined (see Figure 6.9), either to describe valid expressions or to impose restrictions to the search space, it is necessary to validate it. Therefore, considering the grammar G defined in this problem and depicted in Figure 6.9, the following language is obtained: $L(G) = \{(AND\ Comparison)^n\ Comparison \rightarrow (AND\ Comparison)^m\ Comparison : n >= 0, m >= 0\}$. The grammar is therefore well-defined and structured, as any rule with at least one condition in the antecedent and consequent is obtained. Notice that the antecedent and consequent are disjoints sets, meaning

G = $(\Sigma_N, \Sigma_T, P, S)$ with:

S = {*Rule*}

Σ_N = {*Rule, Antecedent, Consequent, Comparison*}

Σ_T = {*'AND', '!=', '=', 'IN',* $attr_1$, ..., $attr_n$, $value_1$, ..., $value_n$ }

P = {*Rule = Antecedent, Consequent ;*
Antecedent = Comparison ;
Consequent = Comparison ;
Comparison = 'AND', Comparison, Comparison | '!=' attr, value |
'=' attr, value | 'IN' attr, value, value ; }

Figure 6.9: Context-free grammar used in the proposed algorithm

that they have no items in common. Using this grammar, it is possible to mine any association rule containing either numerical or nominal features. Numerical attributes are used applying the operator IN and randomly selecting two feasible values within the feasible range of values imposed by the metadata. As for categorical attributes, they could be considered as expressions in both of the following ways: $X = u$ or $X! = u$, where X is a categorical attribute and u is a value in the domain $\mathcal{D}$ of X.

6.2.3 Genetic operator

In order to obtain new individuals in each generation of the evolutionary process, the proposal described in this work uses a genetic operator to introduce diversity to the population, avoiding entrapment in non-optimal solutions.

It randomly chooses a sub-tree from the tree structure of one individual, generating a new sub-tree. A major restriction of the application of a random genetic operator to tree structures is preservation of the grammar. Therefore, the selection of the sub-tree is carefully supervised by this operator, avoiding the construction of invalid individuals that do not satisfy the language derived from the grammar. The proposed genetic operator (see Algorithm 9) restricts sub-tree selection to those sub-trees that form a condition of the resulting association rule. This means that only sub-trees that start from the *Comparison* non-terminal symbol could be selected to be replaced. This operator selects an individual from a set of parents (line

Algorithm 9 Genetic operator

Require: *parents*
Ensure: $offspring$
1: $offspring \leftarrow \emptyset$
2: **for all** individuals in *parents* **do**
3: $individual \leftarrow$ getIndividual(*parents*)
4: **if** random() < operatorProbability **then**
5: $condition \leftarrow$ getRandomCondition(*individual*)
6: $newCondition \leftarrow$ generateNewCondition()
7: $newIndividual \leftarrow$ exchange(*individual*, *condition*, *newCondition*)
8: $offspring \leftarrow offspring \cup newIndividual$
9: **end if**
10: **end for**
11: **return** $offspring$

3), and modifies the individual, based on a probability (line 4). If the probability value is satisfied, then the process of replacing a sub-tree from the individualâÄôs tree structure is carried out (lines 5 to 7), and the new individual obtained from this process is included in the set of offspring (line 8).

A major feature of this G3P algorithm is its ability to autonomously update the genetic operator probability. This updating process is based on the fact that a higher exploration is required in situations where the average fitness value is not improving along the generations. In this process, the proposed algorithm calculates the average fitness value from the elite population in each generation of the evolutionary process. The resulting average fitness value is compared to the prior value, that is, the average fitness value obtained in the last generation. In situations where the evolutionary process is behaving well and the average fitness value obtained is improving, modification of the parameter values would not be required. On the other hand, a higher genetic probability value is of interest if no better solutions are being found.

For a better understanding, Figure 6.10 illustrates a synthetic updating process, demonstrating how genetic probability (dashed line) changes in relation to the fitness function (solid line). In the initial generation of this example, the algorithm generates new individuals based on a specific starting probability, improving the average fitness value so that the genetic operator probability remains the same. In early generations, the average fitness value comes to a standstill, but after some generations, the genetic probability begins to increase, and while this occurs, the

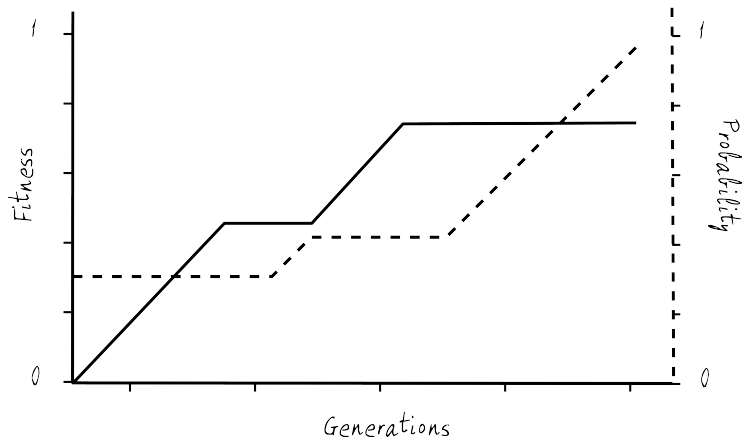


Figure 6.10: Updating process sample based on the fitness value

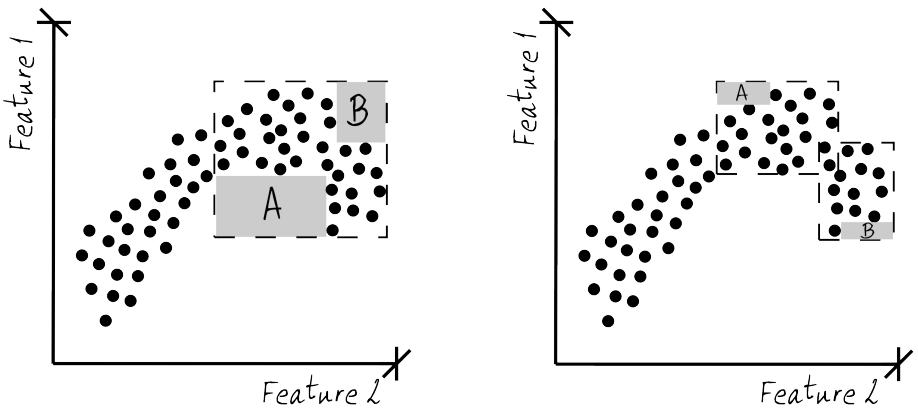
fitness value does not improve. At the instant in which the fitness value begins to improve, the algorithm puts a halt to the probability increment.

The updating process continues until the genetic probability value reaches the maximum value allowed and the average fitness value does not improve any more. The algorithm then finishes, and the best solutions are returned to the data mining expert. In this situation, there would be no sense in continuing the evolutionary process, as no higher probability to generate new individuals could be reached.

6.2.4 Evaluation

As mentioned in previous sections, a major feature of the proposed algorithm is its ability to optimise quantitative association rules. To this end, the search process is guided to look for the right width of values, that is, values containing as few spaces which do not comprise any instances as possible. This procedure seeks the biggest gap within each rule condition, so the size of this gap plays an important role in determining the quality of the.

In order to better understand this process, let us consider a synthetic association rule which comprises two quantitative features (features 1 and 2), whose instance distribution is depicted in Figure 6.11. In situations where the algorithm is designed to discover as many frequent rules as possible, a sample rule could comprise the instances within the dashed line rectangle (Figure 6.11(a)). In analysing the



(a) Instances covered by maximising the support

(b) Instances covered by minimising the gaps

Figure 6.11: Instances covered by sample association rules

rectangle formed by this association rule, we discover that there are two main gaps, represented by A and B. Therefore, despite the fact that this association rule covers a high percentage of instances, huge gaps are also comprised by this rule, meaning it would not be as promising as it seemed to be initially. This means that it is not only the discovery of frequent instances that is of interest, but the distribution of these instances also plays an important role.

On the other hand, in analysing Figure 6.11(b), two different synthetic association rules could be determined to satisfy the same set of instances as the rule depicted in Figure 6.11(a). However, the gaps included in these rules are smaller, so these rules are of high interest to the user. Discovering these gaps is not a trivial issue, especially for rules whose instances are not accordingly distributed or even for rules with a high number of features. Notice that the search space is smaller for minimum rules, that is, rules that comprise only two features (one in the antecedent and one in the consequent), as shown in Figure 6.11(b).

To address the problem of searching for the biggest gap, we propose the use of a real-coded genetic algorithm in the evaluation process. This way, the genotype of each individual within this algorithm is represented by $2 \times n$ genes, n being the number of features comprised by the rule to be analysed. The goal of this genetic algorithm is to discover the best combination of values for each feature in such a way that they represent the biggest gap. In this sense, we are reducing the problem to optimise the biggest blank space.

Let us consider that the evaluation process evaluates the rule $Feature1[2.5, 3.7] \rightarrow Feature2[5.7, 8.1]$. In this way, the real-coded genetic algorithm included in the evaluation process searches for sub-ranges with a maximum blank space, a sample individual of which is $[2.7, 3.10, 5.73, 6.02]$. The individual comprises 4 genes, as two features are to be optimised in this example. The first two genes represent $Feature1$, whereas the last two genes represent $Feature2$. None of the individuals could discover a range of values not comprised within the range determined by the rule.

The genetic algorithm included into the evaluation procedure follows an elitist methodology and uses two well-known genetic operators, the BLX-Alpha crossover and a random mutation. In each generation, the best individual is kept, and this individual is returned if the best result does not improve after 50 generations

(this number has been experimentally obtained). The specific number of generations, was obtained in a study, determines that a higher value does not provide better results. Once the best gap is found, the fitness function for the specific association rule is calculated. This fitness function comprises three functions (see Equation 6.4), which are described in detail below.

$$fitness(A \rightarrow C) = F_3 + F_2 \times F_1 \tag{6.4}$$

F_1 is responsible for searching for a set of instances with small gaps (see Equation 6.5). Therefore, the biggest blank space discovered using the real-coded genetic algorithm is used to determine the interest of the rule, considering its interval width to this end.

$$F_1 = x^2 = \left(1 - \prod_{i=1}^{n} \frac{BlankWidth_i}{IntervalWidth_i}\right)^2 \tag{6.5}$$

It should also be noted that this function is applied in a quadratic way within the fitness function (see Figure 6.12), meaning that the smaller the gap within a rule, the better the rule is.

As the goal of any algorithm for mining association rules is the discovery of frequent and reliable association rules, the support and confidence measures must also be considered. Regarding the support measure, it is of interest to discover the most

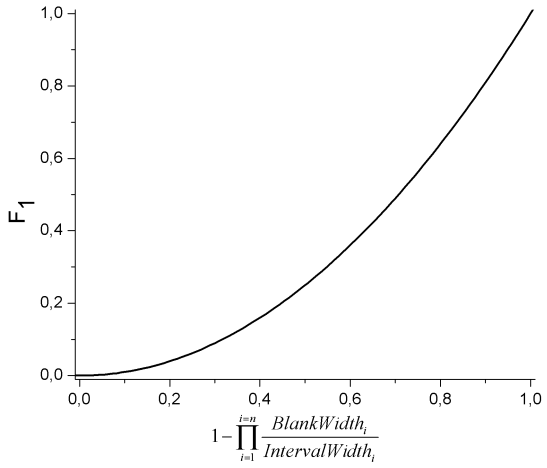


Figure 6.12: Representation of the F_1 function

frequent rules possible. However, the higher the support of a rule, the lower is the degree of interest for the user (see Section 2.1.1). For instance, maximum support values imply misleading rules as stated by the lift, leverage and conviction measures.

Additionally, since the aim of the proposed model is to discover frequent association rules, rules having low support values are not appealing, so a value of zero is assigned to rules that satisfy a low set of instances. In most of the proposals for mining association rules, support values lower than 50% are not of interest, and the higher the value, the better it is. We have therefore considered a function (see Figure 6.13) that reaches the maximum function value with a support close to 50% and decreases the function value when the support is close to the maximum.

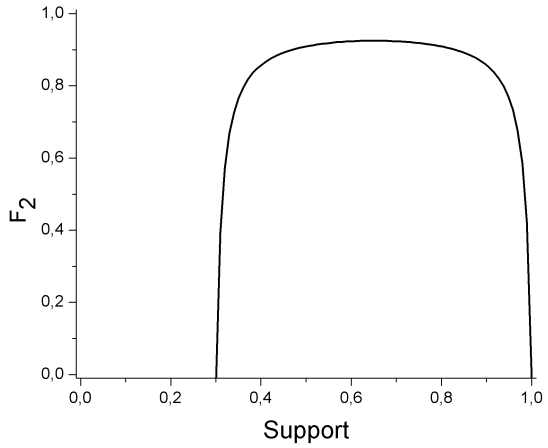


Figure 6.13: Representation of the F_2 function

The mathematical expression of function depicted in Figure 6.13 is proposed in Equation 6.6, "x" stating for the support value in this function. Notice that this function was properly obtained according to desired behaviour, that is, rules with a support value between 50% and 90% are desired. Values above 90% could imply misleading rules, and rules with a support value lower than 50% could not be considered as frequent.

$$F_2 = \frac{x(130 - 100x) - 30}{x(130 - 100x) - 29} \tag{6.6}$$

Finally, the third function included in the fitness function is related to the confidence measure, an important quality measure in determining the reliability of the

rules. Therefore, the higher the confidence value of the rule, the most accurate the rule is. Generally speaking, rules with low confidence values are not of interest to the user. This issue is reinforced by the fact that ARM algorithms usually seek frequent rules, and the confidence value is always greater than the support value, as depicted in Section 2.1.1.

$$F_3 = x^2 \tag{6.7}$$

In this sense, we have defined the F_3 function (see Equation 6.7), which states that the higher the confidence of a specific rule, the higher the F_3 value ("x" representing confidence values). Low confidence values imply low function rates, and these values get higher and higher with the increment of confidence values (see Figure 6.14).

As mentioned above, the goal is to maximise the resulting fitness function (Equation 6.4), which determines values within the range [0, 2]. In situations where rules comprising only discrete features are discovered, this fitness function discards the F_1 function by using its unity value. Consequently, the evaluation process should be designed with this issue in mind, as no searching for gaps is required. Therefore, the real-coded genetic algorithm is simply carried out in such situations where at least one numerical feature is considered within the rule.

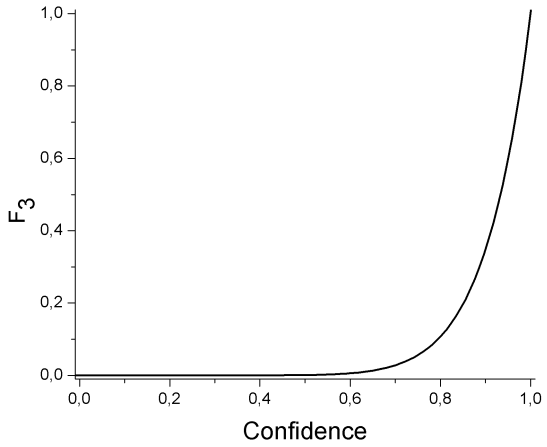


Figure 6.14: Representation of the F_3 function

6.2.5 Datasets

To analyse the effectiveness of the proposed algorithm, a varied set of data from the well-known UCI machine learning repository were considered. The main features of this dataset are depicted in Table 6.9, where data are arranged from the lowest to highest number of instances. Despite the fact that some of the datasets are widely used for supervised learning and a class attribute is therefore considered, we can use these data in unsupervised learning by considering the class attribute in the same way as any other.

6.2.6 Experimental Set-up

For a fair comparison to be drawn, the experiments were split into two separate parts, depending on whether the algorithms worked on discrete features or any type of feature. The algorithms that worked on any type of feature were G3PARM (properly described in Chapter 3) and QuantMiner [11]. As for nominal features, exhaustive search algorithms such as Apriori [5] and FP-Growth [7], together with the evolutionary algorithm GBAP-ARM [90] were considered in the experimental

Table 6.9: Dataset characteristics

		Type of Attributes	
Dataset	**#Inst.**	**#Attr. Continuous**	**#Attr. Nominales**
Lymphography	148	3	16
Sonar	208	60	1
Primary-tumor	339	0	18
Automobile	392	8	0
Soybean	683	0	36
Australian	690	6	10
Vowel	990	10	4
Credit-g	100	6	15
Contraceptive	1473	2	8
Segment	2310	19	1
Chess	3196	0	37
Connect-4	67557	0	43

stage. The algorithms used in this experimental stage are the original algorithms provided by the authors. Finally, it is worth noting that we have considered the optimal parameter provided by the authors (see Table 6.10).

For the G3PARM algorithm, the optimal values were used (see Chapter 3): a population of 50 individuals, a maximum number of 100 generations, probabilities of 70% and 14% for the crossover and mutation operators respectively, a maximum derivation number of 24, an external population size of 20, a 90% confidence threshold, and a 70% support threshold.

The optimal parameters of the QuantMiner algorithm are a population size of 250 individuals, 100 generations, 40% mutation probability, and 50% crossover probability. The support and confidence thresholds considered in this algorithm are 90% and 70%, for confidence and support, respectively.

For the GBAP-ARM algorithm, the parameter configuration corresponds to a population size of 20 ants, 100 generations, a maximum number of 10 derivations, an initial and maximum amount of pheromone of 1.0, a minimum amount of

Table 6.10: Parameters established for each algorithm

Dataset	**Apriori**	**FP-Growth**	**QuantMiner**	**GBAP-ARM**	**G3PARM**	**Proposal**
Pop. size	-	-	250	20	50	-
Pool size	-	-	20	20	20	20
# Generations	-	-	100	100	100	-
Max. derivations	-	-	-	10	24	-
Crossover prob.	-	-	0.50	-	0.70	-
Mutation prob.	-	-	0.40	-	0.14	-
Max. pheromone	-	-	-	1.00	-	-
Min. pheromone	-	-	-	0.10	-	-
Evaporation rate	-	-	-	0.05	-	-
α *exponent*	-	-	-	1.00	-	-
Max. rules	$2E^6$	$2E^6$	-	-	-	-
Min. support	0.70	0.70	0.70	0.70	0.70	-
Min. confidence	0.90	0.90	0.90	0.90	0.90	-

pheromone equal to 0.1, an evaporation rate of 0.05, a value of 1.0 for the α exponent, a 70% support threshold, and a maximum size for the set of rules returned by 20. It is worth noting that this algorithm does not require a confidence threshold.

Apriori and FP-Growth used the same support and confidence thresholds as the other algorithms so that a fair comparison could be drawn. Moreover, since both algorithms are exhaustive search methods, they discover any rule that satisfies the aforementioned thresholds. We have therefore determined the maximum number of rules they can extract, limiting this number to 2,000,000, so it is large enough to analyse how these algorithms behave.

The proposed G3P algorithm reduces the number of parameters significantly (see Table 6.10), especially in relation to evolutionary algorithms. The algorithm here proposed only requires the number of rules to be mined. In order to make a fair comparison, this number of rules is set to 20 as the other algorithms. The remaining parameters self-adapt along the evolutionary process.

6.2.7 Analysis of the Updating Process

In Section 6.2.3, the process for updating the genetic operator probability was described in detail, denoting that this probability is related to the average fitness function value.

The aim of this section is to demonstrate how the algorithm behaves when the process is applied to real data, paying special attention to the probability updating process. In this sense, a number of experiments are carried out with different probability values and different numbers of rules to be discovered. The aforementioned probability value denotes the starting value for the genetic operator, which self-adapts this value based on the algorithm's behaviour.

Figure 6.15 relates the average fitness value (solid line) and the probability value of the genetic operator (dashed line). As shown, the increment in the number of rules to be discovered softens the trend of the fitness value along the generations, meaning that a lower number of rules gives rise to a sharp increase regardless of the starting probability value. This behaviour makes sense, as improvements in the fitness value of one rule in a set of 5 gives rise to a higher average fitness value than that obtained in a set of 20 rules. Also, an increment in the number of rules to

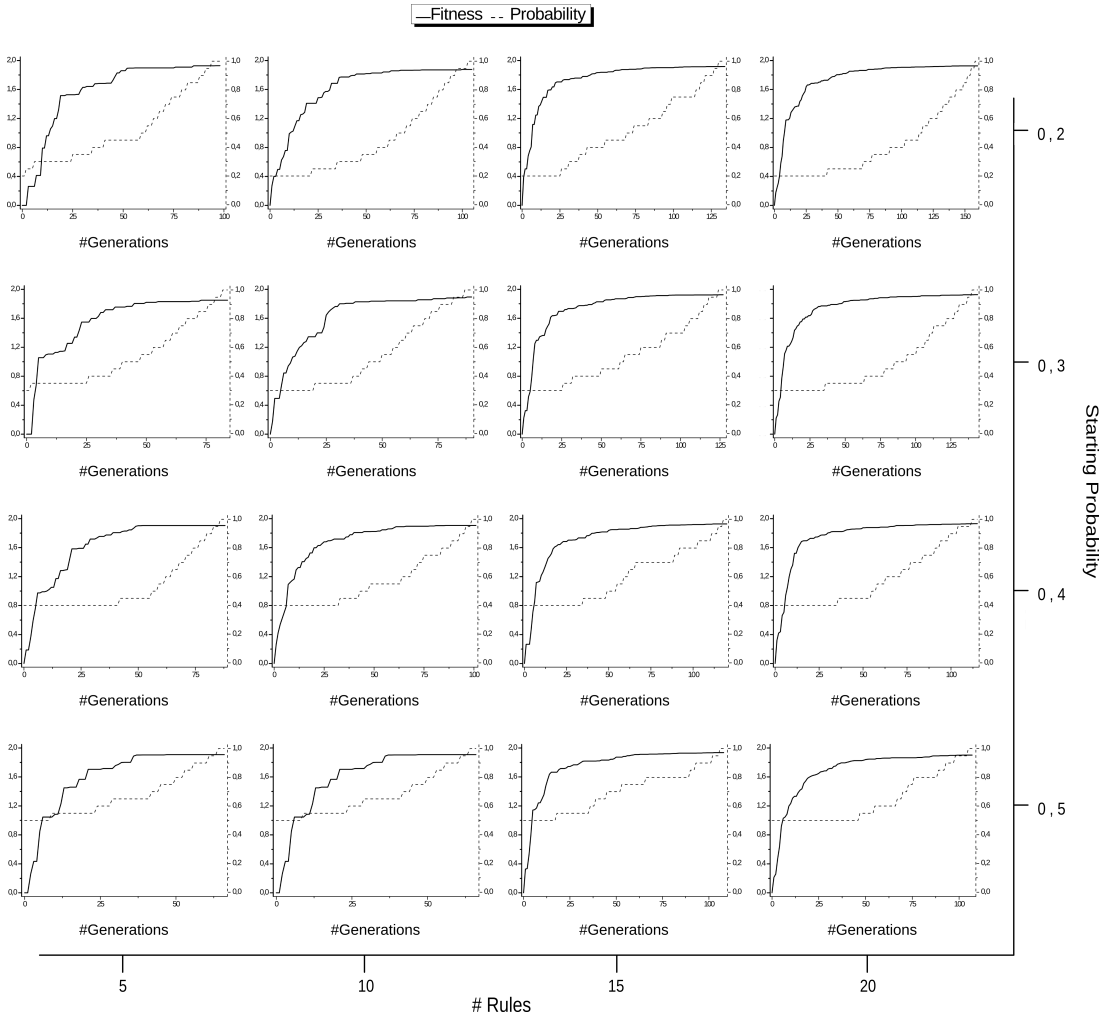


Figure 6.15: Probability updating process for different number of rules and starting probability values

be discovered requires a higher number of generations to reach the optimum value. The most interesting part of this study is the analysis of the algorithm's behaviour for different starting probability values. As shown in Figure 6.15, there was no particular improvement. The algorithm automatically adapts its probability value along the generations, obtaining similar fitness values regardless of the starting probability value. However, it is worth noting that despite the resulting average fitness value remaining the same, the number of generations required to achieve this value increases while the starting probability decreases.

This study explains how the algorithm behaves similarly, regardless of the used parameter value, meaning that its self-adaptation is excellent, providing the same

results for different starting probability values. Therefore, it could state that any starting probability value could be suitable for obtaining the optimal solutions, but a probability of 0.5 enables both the number of generations and, consequently, the execution time required by the algorithm to be reduced. However, this value is not mandatory, as the algorithm behaves similarly when other values are used.

6.2.8 Analysis of the Interval Optimisation

This algorithm, together with QuantMiner and G3PARM, is executed on data with continuous attributes and the resulting association rules are analysed in order to demonstrate the ability of the proposed algorithm to mine optimised quantitative association rules. In order to draw a fair comparison between the three algorithms, only those association rules that comprise the same two features (horsepower and mpg) from the Automobile dataset are considered.

The G3PARM algorithm discovers quantitative association rules using the following four logic operators: $<, <=, >$ and $>=$. It aims to maximise the support quality measure, so that analysis of the distribution of the instances is not carried out in the mining process. Once this algorithm is applied to the *Automobile* dataset, the following rule is obtained: IF ($horsepower$ $<=$ 190) THEN (mpg $<$ 39.08). As for the quality measures, the support value is 0.939, the confidence measure determines a value of 0.973, a 0.999 value is obtained for lift and, finally, both conviction and leverage obtain a value of 0. Following the analysis of the quality measures described in Section 2.1.1, it could be stated that this rule is not of interest, obtaining a lift value close to the unity, implying independence between the antecedent and consequent. Also, a value of zero for conviction and leverage determine a misleading rule. Finally, the distribution of the instances satisfied by this rule is depicted in Figure 6.16. As we can see, the resulting rule satisfies oversized gaps, meaning that it does not accurately represent the covered instances.

As for the QuantMiner algorithm, two quantitative rules that comprise the aforementioned attributes are considered. First, the rule IF $horsepower$ IN $[49.0; 125.0]$ THEN mpg IN $[16.0; 41.5]$ is analysed and depicted in Figure 6.17. This association rule satisfies 71.7% of the instances, with a confidence value of 0.972, a lift value of 1.221, a leverage value of 0.130, and a value of 10.216 for the conviction measure.

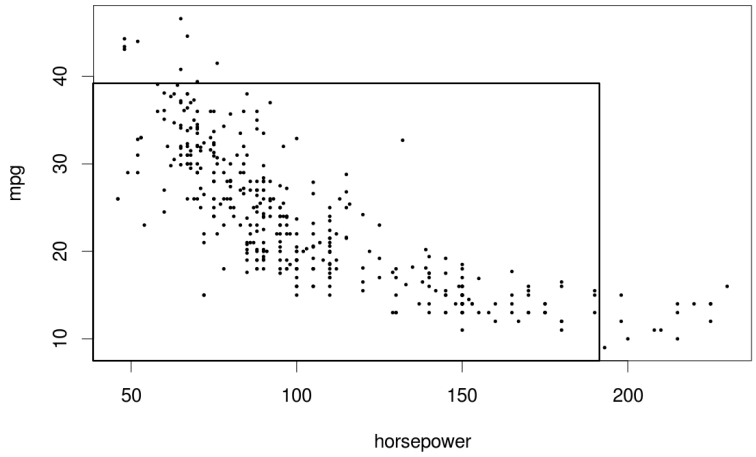


Figure 6.16: Representation of the instances covered by the rule IF $(horsepower <= 190)$ THEN $(mpg < 39.08)$

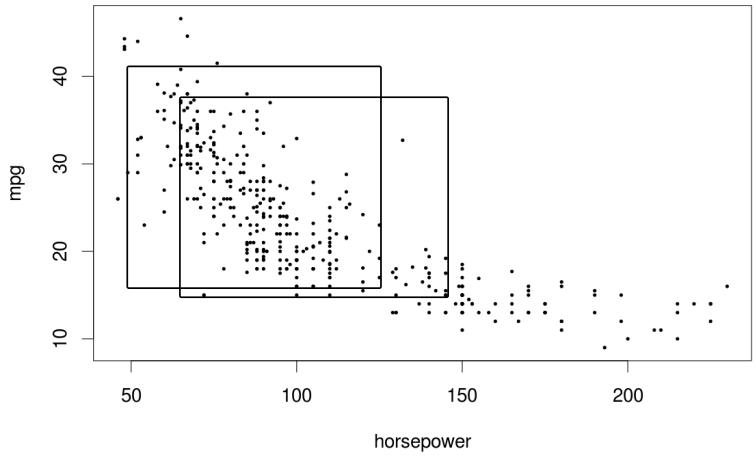


Figure 6.17: Representation of the instances covered by the rules IF *horsepower* IN $[49.0; 125.0]$ THEN *mpg* IN $[16.0; 41.5]$ and IF *horsepower* IN $[65.0; 145.0]$ THEN *mpg* IN $[15.0; 37.3]$

Secondly, the rule IF *horsepower* IN $[65.0; 145.0]$ THEN *mpg* IN $[15.0; 37.3]$ is also obtained. This rule produces a support value of 0.714, a confidence value of 0.948, a lift value of 1.155, and the values of 0.095 and 5.586 for leverage and conviction, respectively.

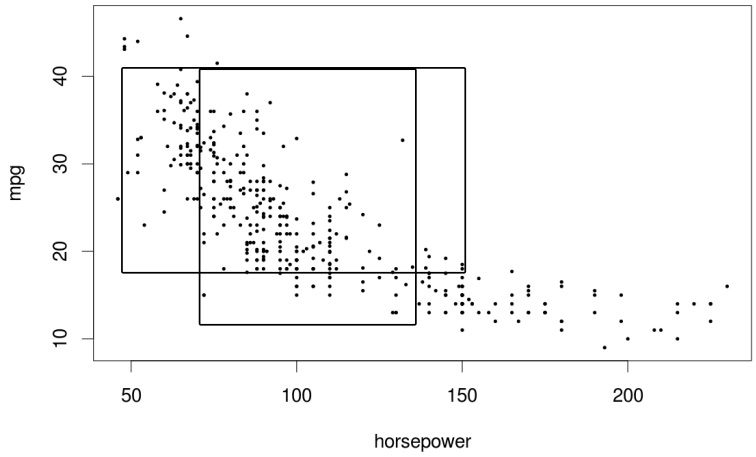


Figure 6.18: Representation of the instances covered by the rules IF *horsepower* IN [71.8; 136.5] THEN *mpg* IN [12.6; 40.8] and IF *mpg* IN [18.5; 40.9] THEN *horsepower* IN [46.2; 150.9]

Finally, the proposed model discovers two association rules comprising both the horsepower and the mpg attributes (see Figure 6.18). The first rule, IF *horsepower* IN [71.8; 136.5] THEN *mpg* IN [12.6; 40.8], calculates a support value of 0.579, a confidence of 0.996, a lift of 1.719, a leverage of 0.242 and a conviction value of 105.25. In a similar way, the second rule obtained with the proposed algorithm is analysed. This rule is defined as IF *mpg* IN [18.5; 40.9] THEN *horsepower* IN [46.2; 150.9], obtaining a support value of 0.645, a confidence value of 0.992, and the values 1.531, 0.224 and 44.38 for lift, leverage and conviction, respectively.

In comparing the resulting values of the quality measures for the proposed model to the values obtained with G3PARM and QuantMiner, we can see that the rules mined with the proposed model are of higher interest, obtaining better values for measures of confidence, lift, leverage, and conviction.

As for the space optimisation for the instances, QuantMiner and the proposed model are the algorithms that best optimise these spaces. These two algorithms obtain numerical intervals that properly represent the set of instances and their rules are of higher interest, as illustrated by analysis of the quality measure values. Regarding the proposed algorithm, the mined rules have better values for the confidence, lift, leverage and conviction measures, obtaining values close to the

maximum for these quality measures. To clarify this issue, a larger study was carried out, considering 12 datasets and comparing the resulting average values (see Table 6.11) for the five quality measures described in Section 2.1.1.

Note that, first, the average results per dataset are computed, and the last row of each table is the ranking obtained by each algorithm. Bold type values indicate the algorithm that attains the best result for a specific dataset. Results marked with "—" point out that no rules were obtained. For instance, QuantMiner is not able to discover any rule situations where no numerical attribute is considered. However, this algorithm does enable nominal features to be discovered in situations where at least one numerical attribute is considered. Table 6.11 shows that the proposed model discovers highly frequent association rules for nominal datasets, that is, those datasets that do not comprise any numerical feature. This issue was aforementioned described in detail, where the fitness function was defined. The results demonstrate, then, that the algorithm here proposed behaves well in both numerical and categorical domains.

In order to analyse these results statistically, the Iman-Davenport test was performed, considering all problems to be equal in terms of importance. The computed value for the Iman-Davenport statistic for the average support distributed according to a F distribution is equal to 3.666, for the confidence measure, its value is equal to 21.162, 5.673 for the lift measure, 2.118 for leverage, and 3.498 for the conviction measure. Except for the leverage measure, none of the values fall within the critical interval at the $\alpha = 0.05$ significance level, which is 3.443. Therefore, the null-hypothesis set out by the Iman-Davenport test that all algorithms perform equally well is rejected for four out of the five measures considered. The rejection of this hypothesis implies the existence of differences between the performances of all of the algorithms studied.

Following this, a post-hoc test could be used in order to find out whether the control or proposed algorithm presents statistical differences with regard to the remaining methods in the comparison. In this study, we therefore proceed with the Bonferroni-Dunn test. This method assumes that the performance of two algorithms is significantly different if their average ranks differ by at least the CD.

The Bonferroni-Dunn CD considering the level of significance $\alpha = 0.05$ is 0.915. At this level of significance, there is no difference for the support measure (see

Table 6.11: Results obtained (presented in a per unit basis)

	Support			Confidence		
Dataset	G3PARM	QuantMiner	Proposal	G3PARM	QuantMiner	Proposal
Lymphography	0.961	0.818	0.605	1.000	0.954	0.973
Sonar	1.000	0.712	0.618	1.000	0.931	0.949
Primary-tumor	0.872	—	0.980	0.993	—	0.989
Automobile	0.858	0.703	0.594	0.993	0.982	0.993
Soybean	0.934	—	0.974	0.987	—	0.993
Australian	0.984	0.809	0.653	0.998	0.960	0.997
Vowel	0.944	0.700	0.713	0.981	0.949	0.994
Credit-g	0.923	0.740	0.999	0.995	0.982	0.999
Contraceptive	0.889	0.921	0.657	0.991	0.979	0.928
Segment	0.998	0.700	0.734	1.000	0.975	1.000
Chess	0.919	—	0.993	0.998	—	0.998
Connect-4	0.926	—	0.999	1.000	—	0.999
Ranking	**1.500**	2.500	2.000	**1.375**	2.917	1.708

	Lift			Leverage		
Dataset	G3PARM	QuantMiner	Proposal	G3PARM	QuantMiner	Proposal
Lymphography	1.020	1.121	1.751	0.012	0.104	0.189
Sonar	1.000	1.753	2.079	0.000	0.103	0.234
Primary-tumor	1.008	—	1.009	0.012	—	0.009
Automobile	1.013	1.568	1.766	0.015	0.183	0.219
Soybean	1.005	—	1.019	0.007	—	0.018
Australian	1.001	1.252	1.677	0.002	0.121	0.181
Vowel	1.002	1.587	1.517	0.008	0.189	0.162
Credit-g	1.001	1.487	1.001	0.001	0.162	0.001
Contraceptive	1.002	1.001	1.497	0.005	0.003	0.166
Segment	1.000	1.473	1.399	0.000	0.201	0.181
Chess	1.001	—	1.005	0.026	—	0.005
Connect-4	1.001	—	1.001	0.013	—	0.001
Ranking	2.417	2.250	**1.333**	2.291	2.167	**1.541**

	Conviction		
Dataset	G3PARM	QuantMiner	Proposal
Lymphography	Infinity	5.278	20.994
Sonar	Infinity	6.874	10.852
Primary-tumor	20.455	—	2.140
Automobile	33.811	37.771	45.542
Soybean	4.991	—	5.209
Australian	1.102	65.110	79.639
Vowel	2.313	66.442	65.460
Credit-g	3.982	18.942	4.107
Contraceptive	1.374	1.199	5.083
Segment	Infinity	32.653	Infinity
Chess	29.661	—	8.699
Connect-4	Infinity	—	4.107
Ranking	2.417	2.125	**1.458**

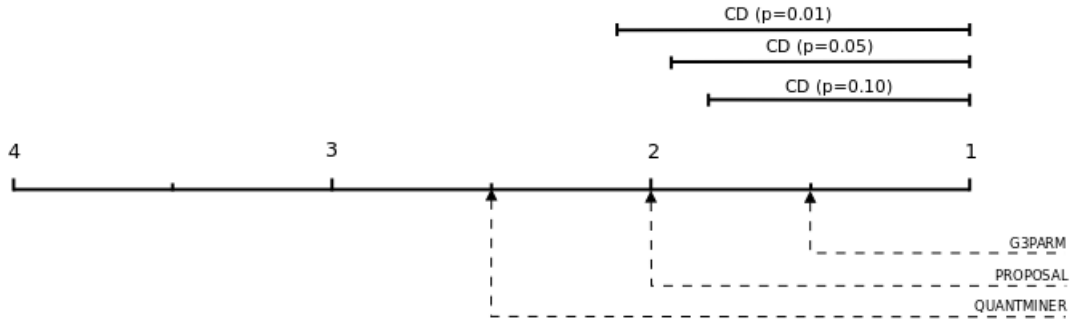


Figure 6.19: Critical difference obtained with the Bonferroni-Dunn test for the support measure

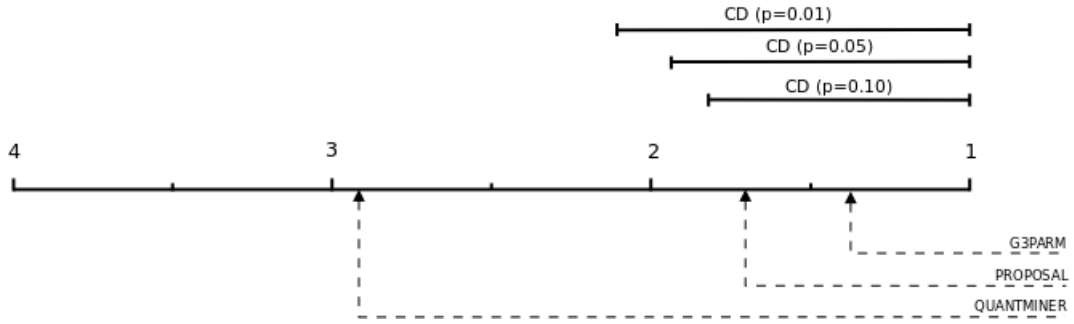


Figure 6.20: Critical difference obtained with the Bonferroni-Dunn test for the confidence measure

Figure 6.19), meaning our proposal does not behave better for this quality measure. As for the confidence measure (see Figure 6.20), there is no difference between G3PARM and the proposed algorithm. However, the proposal presented here behaves statistically better than QuantMiner for this measure at this level of significance.

Regarding the lift, leverage and conviction quality measures, the proposed algorithm obtains the best rankings. Analysing the CDs, and beginning with the lift measure (see Figure 6.21), the proposed algorithm behaves statistically better than G3PARM, and there is no difference with QuantMiner. As for the leverage value (see Figure 6.22), no statistical difference is obtained, but our proposal obtains the best ranking. Finally, the Bonferroni-Dunn test reveals that the proposed algorithm and QuantMiner behave statistically better than G3PARM for the conviction measure (see Figure 6.23).

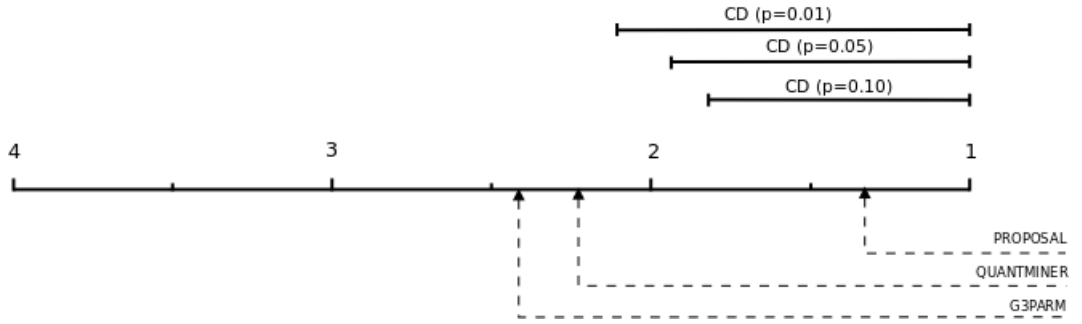


Figure 6.21: Critical difference obtained with the Bonferroni-Dunn test for the lift measure

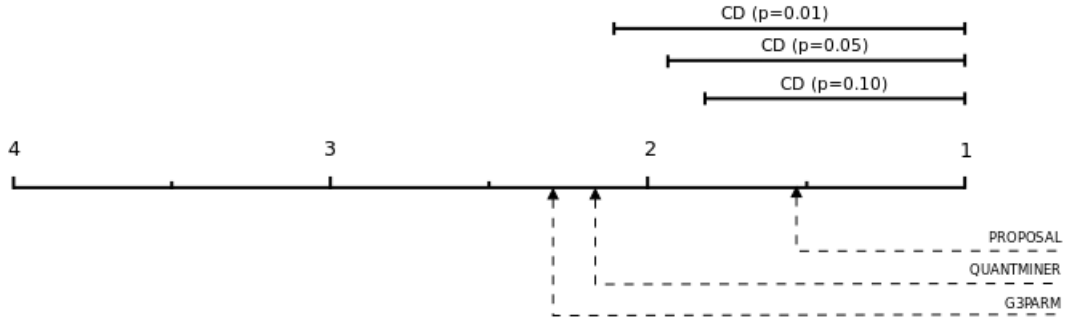


Figure 6.22: Critical difference obtained with the Bonferroni-Dunn test for the leverage measure

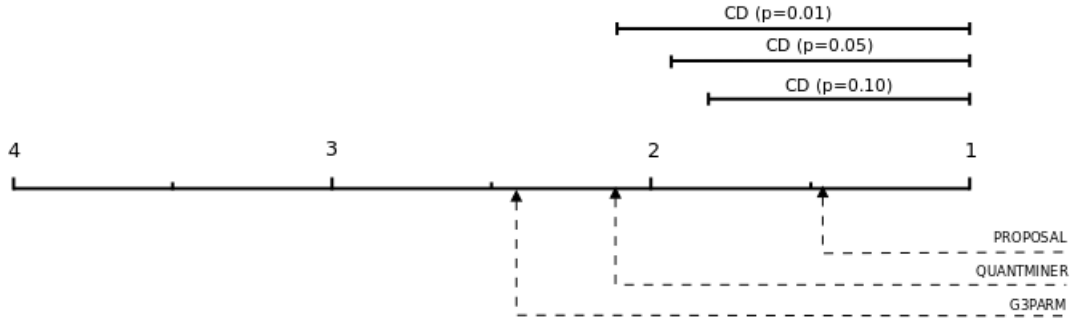


Figure 6.23: Critical difference obtained with the Bonferroni-Dunn test for the conviction measure

6.2.9 Analysis of the Nominal Datasets

In order to compare the behaviour of the proposed algorithm with respect to algorithms that only discover rules in discrete domains, a study over the four nominal

Table 6.12: Average values obtained for the support and confidence quality measures

Support

Dataset	FP-Growth	Apriori	GBAP-ARM	Proposal
Primary-tumor	0.757	0.757	0.973	0.980
Soybean	0.730	0.730	0.968	0.974
Chess	0.759	0.759	0.988	0.933
Connect-4	0.917	0.917	0.988	0.999

Confidence

Dataset	FP-Growth	Apriori	GBAP-ARM	Proposal
Primary-tumor	0.942	0.942	0.989	0.989
Soybean	0.886	0.886	0.989	0.993
Chess	0.939	0.939	0.995	0.998
Connect-4	0.976	0.976	0.999	0.999

datasets is carried out (see Table 6.12). Since FP-Growth and Apriori are exhaustive search approaches, a statistical test here is meaningless, and only the average results have sufficient levels of significance.

The results of FP-Growth are the average values from 209, 112,650, 2,000,000 and 2,000,000 rules that correspond to the datasets Primary-tumour, Soybean, Chess and Connect-4 respectively. As for the Apriori algorithm, the results are the average values from 209, 47,304, 2,000,000 and 2,000,000 rules for the same datasets. It is also worth noting that only the support and confidence measures are considered in this study. These algorithms do not take additional quality measures into account.

6.3 Conclusions

We have proposed two approaches for mining ARs without requiring a high number of parameters as evolutionary proposals do. Both approaches self-adapt their parameter values along the generations and depending on the dataset features. One of the proposed approaches was mainly designed for optimizing numerical conditions. The experimental stage has revealed that the rules mined comprise conditions with

a high density of transactions, minimizing the blank spaces and avoiding misleading rules. Additionally, a series of quality measures have been analysed, the rules mined being of great interest according to five different quality measures.

6.4 Publications Associated with this Chapter

- **International Journal:**

  Title: *An Evolutionary Self-Adaptive Algorithm for Mining Association Rules*

  Authors: J. M. Luna, J. R. Romero and S. Ventura

  IEEE Transactions on Cybernetics, Submitted June, 2013

  Ranking:

  Impact factor (JCR 2012): 3.236

  Knowledge area:

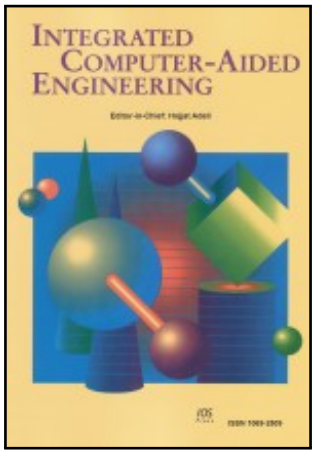
  Computer Science, Artificial Intelligence: 12/114

  Computer Science, Cybernetics: 1/21

- **International Journal:**

  Title: *Mining optimised quantitative association rules using a genetic programming free-parameter algorithm*

  Authors: J. M. Luna, J. R. Romero, C. Romero and S. Ventura



Integrated Computer-Aided Engineering, Submitted June, 2013

Ranking:

Impact factor (JCR 2012): 3.370

Knowledge area:

Computer Science, Artificial Intelligence: 10/114

Computer Science, Interdisciplinary Applications: 10/99

- **International Conference:**

J.M. Luna, J.R. Romero, C. Romero and S. Ventura. *A Genetic Programming Free-Parameter Algorithm for Mining Association Rules*, Proceedings of the 12th International Conference on Intelligent Systems Design and Applications, ISDA 2012, pp. 64-69, ISBN: 978-1-4673-5118-8 [91]

7

Applications of G3P and ARM in Educational Environments

In this chapter, we present two sample applications of G3P and ARM to the educational field. First, we apply the G3PARM algorithm to obtain relations of interest in quiz data. The quiz's answers are used to discover interesting relationships that enable the instructors to improve future courses. Finally, we propose the use of a modified version of Rare-G3PARM to discover students with specific needs. To do so, we use Moodle data and discover relations between the use of resources provided by the instructors and the final mark.

7.1 Providing Feedback to Instructors from Quiz Data

Computer-based testing, also known as quiz systems, is one of the most widely used and well-developed tools in education [92]. They allows to administer tests in which the responses are electronically recorded, assessed, or both. Among the

existing types of quiz systems, multiple-choice questions is one of the most popular, where students are required to select the best answer or group of answers from a set of choices provided on a list. This kind of systems have the following advantages: rapid feedback, automatic evaluation, perceived objectivity, reuse of questions as required, easily computed statistical analysis of test results, and the possibility of generating data that can provide a better understanding of their learning process [93].

The goal of chapter is the discovery of relations of interest to aid the instructor in decision making about how to improve both the quizzes and the courses that contain the concepts evaluated by the quizzes. Thereby, the ARM concepts provided in previous chapters are used to discover specific ARs that are of interest for the instructor.

The objective is not only to show a set of discovered rules but it also goes one step further in that the results obtained are applied in a real course where the improvements achieved are evaluated. Thus, the discovered information is firstly applied to introduce a list of updates in a specific quiz and course. Then, the proposed changes are properly evaluated in order to determine whether they really improve the results achieved by the students.

7.1.1 Proposed Process for Providing Feedback

The task of designing, developing, and evaluating a quiz could be arduous and laborious, and instructors or authors usually have to take important decisions such as: how many questions must a quiz contains? What are the most appropriate questions for evaluating each concept of the course? How much time should students be allowed in order to do the quiz? What are the most discriminatory questions? What behaviour may explain failing or passing the quiz? Are the current contents properly evaluated by the questions? Owing to all these issues, it is quite difficult to determine the most suitable quiz for evaluating a specific course. In fact, it is very likely that different authors would propose different quizzes for the same course.

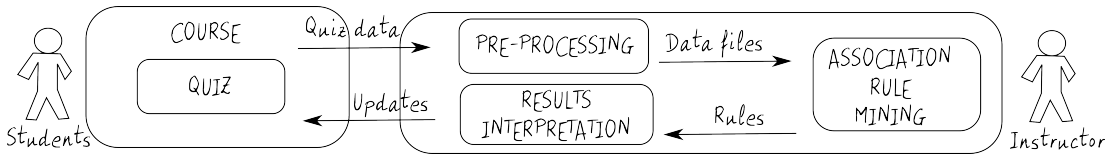


Figure 7.1: Data mining process for providing feedback

We propose the use of ARM to provide feedback to instructors and educational designers. The proposed process could be considered as a quiz and course evaluation cycle (see Figure 7.1) in which the quiz's answers are used for discovering subsequent updates in the quiz and also in the course. In this process, there are two main actors or users: students and instructors. Students use courses for learning concepts and then they do quizzes to evaluate their knowledge about the concepts. Instructors pre-process the quiz data and they run an ARM algorithm to discover ARs of interest. The rules mined are then post-processed in order to detect the most interesting rules for helping in decision making about how to update and improve subsequent quizzes and courses.

Quiz Data. Quiz data can be gathered from different sources, so different data type could be obtained: a score matrix provided directly by the used quiz system, a relationships matrix provided by instructors, and a knowledge matrix automatically generated from the two previous matrices.

- **Score matrix**. This is a traditional student-rating matrix in which each row represents a student and each column represents an item or question (see top left of Figure 7.2). A_{ij} represents the score of $Item_j$ obtained by $Student_i$. Additionally, two columns determine the time used (T_i) and the final score (S_i) of each student. First, the answer or score of one particular item can be *CORRECT* or *INCORRECT*. Second, the total time taken by each student is determined by the *ALL* label if the value is higher than or equal to 34 minutes, and by the *NOT-ALL* label if the value is lower than 34 minutes. Finally, the final score obtained by each student is automatically calculated by adding all the individual answers for each question. The *FAIL* label is used if the number of *CORRECT* answers is lower than 5, *PASS* determines that the number of *CORRECT* answers is higher than or equal to 5 and lower than 7, and the *EXCELLENT* label if the number of *CORRECT* answers is higher than 7. This specific cut-off is based on the Spanish grading scale.

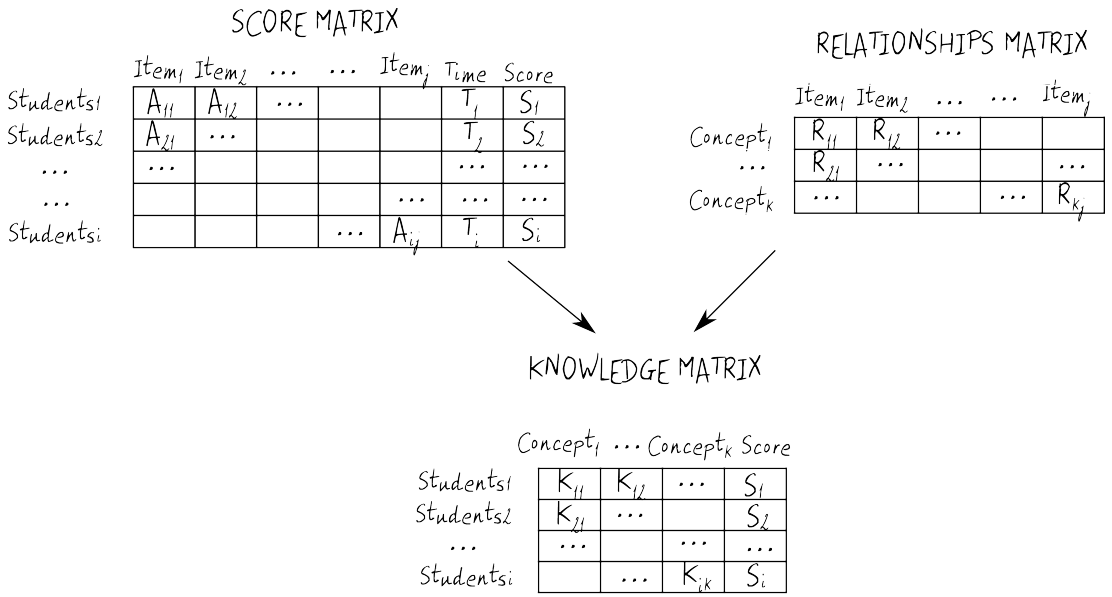


Figure 7.2: Score, relationship, and knowledge matrices

- **Relationships matrix**. This matrix shows the degree of association between the questions or items, and the concepts evaluated by the quiz whereby one item could be related to one or several concepts (see the top of the right-hand-side of Figure 7.2). R_{kj} represents the probability that $Item_i$ is related to $Concept_k$. This matrix is designed by hand by the instructor, specifying the list of concepts that compose the domain taught by the course and that are evaluated by the quiz. Each question and concept is marked as 0 to identify no relation, 1 to show full relation, and 0.5 to state for half relation between question and concept.

- **Knowledge matrix**. This matrix shows the level of knowledge that students have about each concept evaluated by the quiz. It has been created automatically from the two previous matrices (see bottom of Figure 7.2). K_{ik} represents the addition of the answers to each one of the items related to $Concept_k$ by each $Student_i$, that is, $K_{ik} = \Sigma_j A_{i,j} \times R_{k,j}$. Finally, a column with the final score obtained by each student (S_i) is also included. Each value is an integer value between 0 and 5, since in our case each concept has a maximum of five full related questions. Then, these values have been normalized to a real value between 0 and 10. The LOW label is set if the value is lower than 5, the $NORMAL$ label if the value is higher than or equal

to 5 and less than or equal to 7, and the *HIGH* label if the value is higher than 7.

Data has been gathered from a Moodle quiz. All the concepts evaluated by the quiz were explained in an artificial intelligence course during the second year of the Computer Science degree at the University of Córdoba in Spain. This course had traditional in-class lectures and also used Moodle to provide learning content, additional resources, and online activities such as a multiple-choice quiz. 104 students took the quiz at the end of the course, consisting of 40 questions with three possible answers, only one of which was correct. The maximum total time for doing the test was set to 35 minutes, although students could finish their quiz ahead of time.

Algorithm to be Used. To solve the aforementioned problem, we apply the G3PARM algorithm (see Chapter 3), where each individual is evaluated by a fitness function based on the support measure. Nevertheless, a new fitness function is applied in order to maximize both support and confidence measures in a different grade: $F = w_1 \times support + w_2 \times confidence$, where w_1 and w_2 are parameters to weight the support and confidence metrics, respectively. Notice that $w_1 + w_2 = 1$, and both parameters should be established based on the importance of support and confidence in the resultant rule. A high value of w_1 allows to discover rules having high support and, therefore, having high confidence. On the other hand, a high value of w_2 allows to discover rules having high confidence. However, the mere fact of having a high confidence does not guarantee a high support (see Section 2.1.1). To study the behaviour of these parameters, a series of experiments was carried out by using values from 0.1 to 0.9 because values 0 and 1 are meaningless. Since our goal is the discovery of both frequent and reliable association rules, the expert in the field (according to the results obtained during the experimental stage) determined that support and confidence must be equally weighted so $w_1 = w_2 = 0.5$.

7.1.2 Experiments and Results

The objective of the result interpretation is to analyse the set of discovered rules in order to provide feedback for improving quiz and course. ARM algorithms normally discover a huge number of rules but not all are interesting for the user.

End users have a crucial role since they should guide the search for the best set of rules. In our case, three experts in the domain have identified general patterns and types of interesting relations. This post-mining procedure has been carried out as described: (a) three experts identified the types of general patterns considering the information provided by the matrices; (b) they selected the types of interesting relations analysing all the rules discovered by the G3PARM algorithm; (c) they chose the best obtained rules for each type of relation.

Next, some examples of the rules previously discovered by G3PARM are classified by type of pattern and relation together with information about the feedback provided by each one. Rules obtained by means of the score matrix show relationships between items, times, and scores, so three types of patterns of rules can be distinguished.

Item-item pattern. This pattern shows relationships between several items or questions. Experts have distinguished the following two types of relations:

- Relations between to get several right questions. They show that if students get one right item then they also get another one or more right items. An example of an obtained rule of this type is:

  IF Item-Num.12 = CORRECT **AND** Item-Num.29 = CORRECT
  THEN Item-Num.38 = CORRECT
  (Support = 0.631, Confidence = 0.902, Lift = 1.177)

- Relations between to get several wrong questions. They show that if students get one wrong item then they also get another one or more wrong items. An example of an obtained rule of this type is:

  IF Item-Num.24 = INCORRECT **AND** Item-Num.35 = INCORRECT
  THEN Item-Num.8 = INCORRECT
  (Support = 0.640, Confidence = 0.985, Lift = 1.193)

In general, these two types of relationship show questions that could evaluate the same or closely related concepts. The first rule could identify easy questions, so the instructor should check the content of these questions in order to increase their level of difficulty, if necessary. In a similar way, the second rule could identify questions that could be quite difficult or could contain any typo. In this sense,

the instructor should check the content of these questions to decrease their level of difficulty if necessary or to correct the possible error.

Item-score pattern. This pattern shows relationships between items and scores. The experts have distinguished two very interesting subtypes:

- Relations between to get wrong questions and fail the exam. For example, the next rule shows that if students make a mistake in questions 1 and 29, then they obtain a FAIL score.

  IF Item-Num.1 = INCORRECT **AND** Item-Num.29 = INCORRECT
  THEN Score = FAIL
  (Support = 0.631, Confidence = 0.984, Lift = 1.102)

- Relations between to get right questions and pass the exam with a high score. For example, the next rule shows the opposite relation to the previous one, showing that getting correct answers is related to an *EXCELLENT* score.

  IF Item-Num.23 = CORRECT **AND** Item-Num.31 = CORRECT
  THEN Score = EXCELLENT
  (Support = 0.621, Confidence = 0.941, Lift = 1.227)

The instructor could select the specific questions that appear in these rules as good discriminatory items of *FAIL* and *EXCELLENT* scores, respectively. For example, the instructor could create a new and shorter version of the quiz made up of only this type of items because they are the most discriminatory ones.

Item-time-score pattern. The following type of relations have been selected by the experts:

- Relation between wrong questions, the score obtained, and the time spent to do the quiz. For instance, the following rule shows that students who get wrong the question number 40 and also obtain a *FAIL* score, then they use *ALL* the time provided.

  IF Item-Num.40 = INCORRECT **AND** Score = FAIL
  THEN Time = ALL
  (Support = 0.611, Confidence = 0.926, Lift = 1.084)

- Relation between right questions, the score obtained, and the time used to do the quiz. For example, the following rule shows that students who get right the item number 26 and do not use all the time provided, then they obtain a *GOOD* score.

  IF Item-Num.26 = CORRECT **AND** Time = LESS
  THEN Score = GOOD
  (Support = 0.660, Confidence = 1.0, Lift = 1.051)

Again, the instructor could use questions that appear in these rules as good discriminatory items of the final score obtained. The instructor could also consider providing less or more time to execute the quiz because of the relationships between to use (or not) all the time provided and the score obtained.

With respect to rules obtained using the knowledge matrix, they show relationships between concepts and scores, so two different types of patterns can be distinguished.

Concept-concept pattern. It shows relationships between different concepts, and two types of relations have been distinguished by experts:

- Relations that include low levels of knowledge in several concepts. For example, the following rule shows that if students have a *LOW* knowledge level in the *Rules Definition* and *Rule Conditional Element* concepts, then they also have a *LOW* knowledge level in the *Rules Execution* concept:

  IF Rules-Definition = LOW **AND** Rule-Conditional Element = LOW
  THEN Rules Execution = LOW
  (Support = 0.466, Confidence = 0.960, Lift = 1.177)

- Relations between to obtain a high level of knowledge in several concepts. For example, the following rule shows that students who obtain a *HIGH* knowledge level in *Initial Facts*, *Rule Conditional Element*, and *Rules Execution* concepts also obtain a *HIGH* knowledge level in the *Functions* and *Actions* concept.

  IF Initial-Facts = HIGH **AND** Rule-Conditional-Element = HIGH
  AND Rules-Execution = HIGH **THEN** Functions-And-Actions = HIGH
  (Support = 0.388, Confidence = 0.952, Lift = 1.167)

These two relations show two concepts that are closely related. The first relation, however, can be used to detect concepts that could have brief or unsuitable contents in the course. The instructor should check the contents of these chapters in the course in order to decide if they should be modified or extended for improvement. The second relation can be used to detect good concepts that are related. The instructor must check if these chapters are located together in the course in order to decide if they could be placed closer together or even combine in only one concept.

Concept-score pattern. It includes relations between concepts and scores, and the following relations of interest have been selected by experts:

- Relations between to obtain a high level of knowledge in one or several concepts and a high score in the exam. For example, the following rule shows that if students have a *HIGH* knowledge level in *Rule Conditional Element* and *Rules Execution* concepts, then they obtain an *EXCELLENT* score.

  IF Rule-Conditional-Element = HIGH **AND** Rules-Execution = HIGH
  THEN Score = EXCELLENT
  (Support = 0.320, Confidence = 1.0, Lift = 1.907)

- Relations between to obtain a low level of knowledge in one or several concepts and to obtain a *LOW* score in the exam. For example, the following rule shows that students who have a *LOW* knowledge level in *Initial Facts* and *Variables and Wildcard* concepts also obtain a *FAIL* score.

  IF Initial-Facts = LOW **AND** Variables-And-Wildcards = LOW
  THEN Score = FAIL
  (Support = 0.310, Confidence = 1.0, Lift = 1.226)

These relations show the instructor the most influential concepts for obtaining a good or a bad score. The instructor should check the content of these chapters in order to decide if they should be modified or extended for improvement.

7.1.3 Evaluating Updates done in Quiz and Course

A pilot experiment was conducted to evaluate the effect of applying updates in the aforementioned quiz and course starting with the feedback provided by the rules previously discovered.

The hypothesis was that the updated quiz and course would have a beneficial effect on student performance. The list of specific updates in both the quiz and course is as follows:

- The contents of some questions have been modified starting with information provided by mined relations. The instructor checked and realized that two questions (numbers 24 and 35) were really very difficult questions and four questions (numbers 12, 14, 17, and 38) were really very easy questions. The instructor therefore decided to modify these questions briefly in order to decrease or increase their level of difficulty. The instructor also checked and realized that question number 9 had an error in the answer and so it was modified to correct it.

- Some questions have been removed from the quiz starting with the information provided by some relations. In fact, some questions did not appear in any of the rules obtained and it could indicate that they did not affect significantly to the final score. The instructor decided, after reviewing the content of these questions, to remove five of these questions since they were not of interest.

- The time available to respond to each question has been increased from the information discovered. Although the maximum total time provided to students was the same (35 min), the average time to respond to each question was increased (from 52.5 to 60 s) because the number of questions was reduced (from 40 to 35).

- The contents of some concepts in the course have been improved. The instructor decided, after reviewing the content of concepts referenced by the discovered rules that the following concepts should be extended: *Variables and Wildcards*, *Rules Definition*, *Rules Execution*, and *Rule Conditional Element*.

It is interesting to notice that the post-mining procedure used to select the best rules is quite subjective since it is based on three experts. Thus, the list of updates in the quiz and in the course could also be modified depending upon the experts.

After the application of all these specific updates, two new groups of students took the updated course and quiz. The effectiveness of these updates was evaluated in the light of the performance by the students, that is, by comparing the score obtained with this updated quiz and course versus the original quiz and course. Therefore, there were three groups of students: one control and two experimental groups. On the one hand, a total of 104 students (control group) took the original Moodle course and quiz during the 2008–2009 academic year. On the other hand, 98 students (experimental group one) and 102 students (experimental group two) took the updated course and quiz during the 2009–2011 academic years. All these groups of students had similar features (age, previous experience, and knowledge in Computer Science, etc.), and all of them were students on the same course (second year) of the Computer Science degree at the University of Córdoba in Spain.

Two experimental studies were carried out to analyse whether there are differences between the scores obtained by the different groups of students. On the one hand, the first study evaluates both the changes on the quiz and the course. On the other hand, the second study evaluates only the changes on the course. For both studies a statistical analysis was considered. First of all, it was checked that the values of the obtained scores in all the groups were normally distributed in order to decrease the risk of error. Histograms of the scores obtained showed a bell-shaped curve for all the groups. Then, descriptive and comparative statistics were calculated (see Table 7.1) such as: the number of questions in each quiz (Q), the number of students in each group (N), the score average value (Mean), the mean difference between two groups (Mean Differences), the standard deviation (SD) that quantifies score variability, standard error of the mean (SEM) that gives an idea of the accuracy of the mean, and the Student t-Test p value (p) that compares the means of two groups.

In the first study (see comparison 1 and 2 in Table 7.1), the three groups were compared in pairs (control versus experimental), that is, the scores obtained in one year were compared versus the scores obtained in the other years. As we have commented, the objective was to test whether the updates done in both the quiz

Table 7.1: Descriptive statistics of each group and pairwise comparison between control and experimental groups (* only the common items in both quizzes are used)

Number Comp.	Group	Q	N	Mean	Mean difference	SD	SEM	p
1	2008–2009	40	104	5.9469	0.6844	1.8074	0.1772	0.0060
	2009–2010	35	98	6.6313		1.6758	0.1702	
2	2008–2009	40	104	5.9469	0.7113	1.8074	0.1772	0.0041
	2010–2011	35	102	6.6583		1.7093	0.1692	
3	2008–2009*	29	104	5.9904	0.6502	1.8607	0.1825	0.0109
	2009–2010*	29	98	6.6406		1.7156	0.1742	
4	2008–2009*	29	104	5.9904	0.6804	1.8607	0.1825	0.0103
	2010–2011*	29	102	6.6708		1.9112	0.1892	

and course had an effect (positive or negative) on the student scores. Table 7.1 shows that the differences between the control group and the two experimental groups can be considered statistically significant with a confidence level of 99% ($p < 0.01$). In fact, there is a mean difference of 0.68 points (in a scale from 0 to 10 points), with 2009–2010 students scoring better than 2008–2009 students, and 0.71 points with 2010–2011 students scoring better than 2008–2009. Thereby, updates done in the quiz and the course can be considered as very positive to obtain better scores.

In the second study (see comparison 3 and 4 in Table 7.1), the three groups were compared in pairs (control versus experimental), that is, the student scores were obtained starting on only the common 29 items in both quizzes. In this case, the six modified questions and the five deleted questions were not used to obtain the student scores. The objective was to test whether the updates done only in the course had an effect (positive or negative) on the student scores. Table 7.1 shows that the differences can be considered statistically significant with a confidence level of 95% ($p < 0.05$). Again, 2009–2010 and 2010–2011 students scoring better than 2008–2009 students, with a similar but less mean difference of 0.65 and 0.68 points, respectively. Thus, updates done only in the course can also be considered as positive on the performance of the students.

7.2 Discovering Rare Association Rules in Learning Management Systems

In previous sections, we have described ARM problems and the importance of applying it for mining not only frequent ARs but also rare or infrequent ARs. More specifically, this type of relation could be very appropriate in e-learning domains due to its intrinsic imbalanced nature. In educational problems, the aim is to discover a small but interesting and useful set of rules, discovering useful information about students' unusual behavior regarding the achievement of bad or good marks.

In this work, we propose to use the Rare-G3PARM algorithm, which was properly described in Chapter 4. This interesting algorithm for mining ARs that do not frequently occur in data has been briefly modified to the problem under study. Thus, the CFG has been modified for the sake of mining only ARs having the final mark of the course in the consequent, and not requiring numerical attributes to be handled. Finally, an analysis of the rules mined is presented, analysing the importance of these rules to understand or discover unusual behavior in the students' learning process.

7.2.1 Adjusting Rare-G3PARM

Rare-G3PARM enables different types of data to be handled without producing invalid individuals by using a grammar. As mentioned in its encoding criterion definition (see Chapter 4), Rare-G3PARM uses a CFG defined as a four-tuple (Σ_N, Σ_T, P, S) where Σ_T represents the alphabet of terminal symbols and Σ_N the alphabet of non-terminal symbols. Notice that they have no common elements, i.e., $\Sigma_N \cap \Sigma_T = \emptyset$.

We propose a modification of the aforementioned grammar to the problem under study. In this new grammar (see Figure 7.3), no numerical attributes are required since data were categorized by the instructors according to their own criterion. Additionally, the experts in the domain under study determined that no negative attribute or condition is required. They established that this kind of attribute could be hardly understandable and it is of interest to remove them. Finally,

G = $(\Sigma_N, \Sigma_T, P, S)$ with:

S = $\{Rule\}$
Σ_N = $\{Rule, Antecedent, Consequent, Comparison\}$
Σ_T = $\{$*'AND', '=', 'name', 'mark', 'value'*$\}$
P = $\{Rule$ = *Antecedent, Consequent ;*
Antecedent = Comparison | 'AND', Comparison, Antecedent ;
Consequent = '=', 'mark', 'value' ;
Comparison = '=', 'name', 'value' ;$\}$

Figure 7.3: Context-free grammar expressed in extended BNF notation

the proposed study is related to the students' final mark, so the consequent only comprises this kind of attribute.

As for the genetic operator proposed in the Rare-G3PARM algorithm, it was briefly modified in order to not change the consequent of the rule. Thus, it remains the same and only conditions within the antecedent could be considered to be changed.

7.2.2 Experiments and Results

Dataset. The experiments were performed using data collected from 230 students on 5 Moodle courses on Computer Science at the University of Córdoba. Moodle keeps detailed logs of all the activities performed by these students (e.g., assignments, forums, or quizzes). All this information was properly preprocessed, including the transformation of every continuous attribute into a discrete domain, so they can be treated as categorical attributes.

The following list of attributes summarises the most important information about the activities monitored by Moodle from students during the life of the course:

- *course*: identifies the course. Its available values are: C218, C94, C110, C111 and C46.
- *n_assigment*: determines the number of assignments done. Its available values are: ZERO, LOW, MEDIUM, HIGH.
- *n_quiz*: establishes the number of quizzes taken. Its available values are: ZERO, LOW, MEDIUM, HIGH.

- *n_quiz_pass*: determines the number of quizzes passed. Its available values are: ZERO, LOW, MEDIUM, HIGH.
- *n_quiz_fail*: identifies the number of quizzes failed. Its available values are: ZERO, LOW, MEDIUM, HIGH.
- *n_posts*: determines the number of messages sent to the forum. Its available values are: ZERO, LOW, MEDIUM, HIGH.
- *n_read*: identifies the number of messages read on the forum. Its available values are: ZERO, LOW, MEDIUM, HIGH.
- *total_time_assignment*: establishes the total time spent on assignments. Its available values are: ZERO, LOW, MEDIUM, HIGH.
- *total_time_quiz*: determines the total time spent on quizzes. Its available values are: ZERO, LOW, MEDIUM, HIGH.
- t*otal_time_forum*: determines the total time spent on the forum. Its available values are: ZERO, LOW, MEDIUM, HIGH.
- *mark*: establishes the final mark obtained by the student on the course. Its available values are: ZERO, LOW, MEDIUM, HIGH.

It is worth mentioning that the values of two of these attributes (course and mark) are clearly distributed in an imbalanced way. So, from 230 students, 116 students obtained a PASS in the final exam with a normal/medium score, 87 students obtained a FAIL in the exam, 15 students obtained an EXCELLENT or a very good/high score in the exam and 12 students were ABSENT from the exam. Thus, there are two predominant marks (PASS and FAIL) and two minority marks (EXCELLENT and ABSENT).

On the other hand, concerning the course attribute, from a total of 230 students, 80 took course 218, 66 students did course 94, 62 students did course 110, 13 students took course 111 and 9 students took course 46. Thus, there are three predominant courses (C218, C94 and C110) and two minority courses (C111 and C46).

Experimental results. Next, we illustrate how the information is provided to the instructor after the mining task execution. Some rules discovered by RareG3PARM are shown and described. This analysis allows a demonstration of their

Table 7.2: Descriptive statistics of each group and pairwise comparison between control and experimental groups (only the common items in both quizzes are used)

#Rule	Antecedent	Consequent (Mark)	Support	Confidence
1	total_time_quiz=ZERO AND n_assignment=LOW	FAIL	0.021	1.00
2	n_quiz_fail = ZERO AND n_assignment=LOW	FAIL	0.021	1.00
3	n_read=HIGH AND total_time_quiz=LOW	PASS	0.039	1.00
4	n_quiz_fail=LOW AND n_read=HIGH	PASS	0.052	1.00
5	total_time_forum=ZERO AND n_quiz_pass=ZERO	ABSENT	0.047	1.00
6	total_time_assignment=ZERO AND n_assignment=ZERO	ABSENT	0.047	1.00
7	total_time_assignment=HIGH AND n_assignment=HIGH	EXCELLENT	0.017	1.00
8	course=94 AND total_time_assignment=HIGH	EXCELLENT	0.013	1.00

usefulness in making decisions about how to detect in time successful and failed students starting on their activities in the Moodle environment. For every rule, the antecedent and the consequent, as well as their support and confidence values are shown.

Focusing on Table 7.2, the algorithm proposed in this chapter discovers rules with any type of mark and course, that is, it discovers infrequent rules that contain not only infrequent but also frequent patterns, favouring the diversity of study cases.

Rule #1 shows that if students do not spend any time on quizzes and the number of assignments is low, then they fail the final exam. So, it is an expected rule that shows the instructor the importance of using quizzes and assignments to pass the exam. Rule #2 is a very similar rule. This rule states that if students do not fail any quizzes but their number of assignments is low, then these students also fail the final exam. This rule is very interesting because it shows how the fact of passing the quizzes may not be condition enough for passing the final exam. Rule #3 is an

interesting rule, since it shows that students that only spend a short time on quizzes could also pass the exam if they read a lot in the forum. So, this rule states that reading messages in the forum could significantly help student pass the exam. Rule #4 is very similar to Rule #3. Here, students that fail very few quizzes and read a lot in the forum pass the exam. Rule #5 shows an interesting rule. It states that students that do not spend any time on the forum and do not take any quizzes do not take the final exam. Similarly to Rule #5, Rule #6 shows that if the students do not do assignments, then they will be absent from the exam. Finally, Rules #7 and #8 identify students with an excellent mark. More specifically, these rules show that if students have a high number of assignments and spend a lot of time on them, or they submit a large number of assignments and they are subscribed to course 94, then these students obtain an excellent mark. Observe that this kind of rules help the instructor predicts the final performance of the students (both pass, fail, absent or excellent) before the exam.

The application of this approach has shown to be an interesting research line in the context of educational DM, where most real-world data are usually imbalanced. Infrequent ARs are more difficult to be mined by using traditional ARM algorithms since they do not usually consider class-imbalance and tend to be overwhelmed by the major class, whilst ignoring the minor class.

7.3 Conclusions

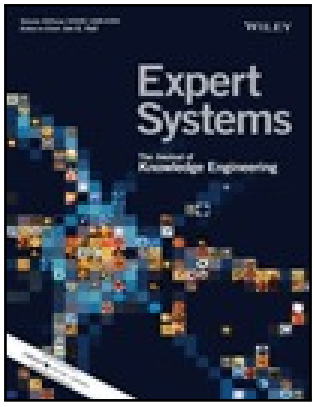
In this chapter, we have presented the application of the proposed models for mining ARs to the educational field. First, G3PARM has been applied to quiz data, discovering interesting relationships among answers of the quiz. These relationships are of great interest since they help instructors to improve future courses. Additionally, we have dealt with the mining of RARs in the educational field, discovering reliable relations that do not frequently occur. This is highly interesting since it allows to discover students with specific needs that are hardly discover by searching for frequent relations.

7.4 Publications Associated with this Chapter

- **International Journal:**

  Title: *Association rule mining using genetic programming to provide feedback to instructors from multiple-choice quiz data* [94]

  Authors: C. Romero, A. Zafra, J. M. Luna and S. Ventura



  Expert Systems, Volume 30, Issue 2 (May 2013), pp. 162-172

  Ranking:

  Impact factor (JCR 2012): 0.769

  Knowledge area:

  Computer Science, Artificial Intelligence: 82/114

  Computer Science, Theory & Methods: 53/100

- **International Journal:**

  Title: *An Evolutionary Algorithm for the Discovery of Rare Class Association Rules in Learning Management Systems*

  Authors: J. M. Luna, C. Romero, J. R. Romero and S. Ventura

  Applied Soft Computing, Submitted May, 2012

  Ranking:

  Impact factor (JCR 2012): 2.140

  Knowledge area:

Computer Science, Artificial Intelligence: 23/114

Computer Science, Interdisciplinary Applications: 18/99

- **International Conference:**

  C. Romero, J. M. Luna, J. R. Romero and S. Ventura. *Mining Rare Association Rules from e-Learning Data.* Proceedings of the 3rd International Conference on Educational Data Mining, EDM 2010, pp. 171-180, ISBN: 978-0-615-37529-8 [95]

8

Conclusions and Future Work

The work developed along the execution of this Doctoral Thesis cannot be thought of as concluded but the starting point to a more extensive research in the field. Therefore, although a brief summary of the results obtained and the main conclusions around the work done are presented in this chapter, a series of research lines are also described, which could be built up taking the present work as a basis. Finally, any of the scientific publications associated to this Thesis are enumerated.

8.1 Concluding Remarks

In this PhD Thesis we have explored the use of G3P in mining ARs. The results obtained are highly promising and show that G3P is an appropriate technique to undertake the ARM task.

A number of G3P-based algorithms for mining ARs from real-world and large data sources are presented. The proposed algorithms use a CFG that allows both categorical and numerical attributes to be defined. The use of any type of dataset without the need of a pre-processing step is one of the main advantages of using G3P. One of the most remarkable features of these algorithms that differentiates

them from other existing algorithms in this field is the use of some logical operators that can be changed beforehand according to the expert's needs.

The use of G3P for the discovery of ARs allows us to obtain understandable and closer relations between items. Moreover, according to the definition of an individual prescribed by the grammar, the proposed algorithms may produce any kind of ARs, that is, rules that comprise not only positive and categorical attributes but also quantitative and negative attributes.

An additional feature of the algorithms proposed is the low number of rules discovered. Since the number of rules is restricted by the population size and guided by the fitness function, the proposed algorithms always obtain a reduced set of ARs but covering a high percentage of instances in the dataset used. Hence, the proposed algorithms do not provide a huge set of rules hardly understandable as exhaustive search algorithms mostly do. On the contrary, they only provide the best rules discovered along the evolutionary process.

The results obtained during the experimentation phase outline some conclusions concerning the effectiveness of our proposals. The scalability of G3P for the extraction of ARs is quite linear when the dataset size or the number of attributes are increased. Moreover, G3P for mining ARs requires low computational time for both numerical and categorical attributes.

The following points summarize the most relevant results presented in the different chapters of this PhD Thesis and the conclusions obtained.

- A thorough bibliographical revision covering the topics treated in this Thesis has been discussed. As main topics it includes a study of the proposals by using both exhaustive search and evolutionary methodologies that exist in literature. Notice that G3P has not been previously used in ARM.

- Computational and memory requirements of ARM have been overcome. All the proposed algorithms follow an evolutionary methodology and do not require the same two steps for mining ARs as exhaustive search algorithms do. Hence, the proposed G3P algorithms are able to discover frequent and also infrequent ARs without requiring the previous discovery of frequent or infrequent patterns.

- All the proposed algorithms are based on a CFG that increases the flexibility and interpretability of the extracted knowledge. This grammar also enables ARs that comprise positive, quantitative and negative conditions to be mined, and the logical operators used to mine this kind of attributes could be predefined or modified depending on the domain under study.

- The mining of reliable and infrequent ARs is an easy task, requiring a higher computational time because of the high number of rules that could appear from data. We present the Rare-G3PARM algorithm, which was specifically designed to mine this sort of infrequent ARs, and it has provided pretty promising results.

- The ARM problem could be considered as a multi-objective problem when optimizing more than one measure at time is required. For this purpose, we have presented two multi-objective approaches for mining ARs by means of G3P. The results have revealed that both algorithms behave very well for optimizing both support–confidence and support–lift at time.

- An important drawback in evolutionary computation is the high number of parameters required to execute algorithms. Besides, they are often required to be previously optimized to obtain the best results. Sometimes, the number of parameters is too high to be accordingly optimized in a reasonable time. To address this issue, we have proposed two new G3P models that do not require as many parameters as existing EAs do, self-adapting their parameters to the data under study as the evolutionary process evolves.

- Finally, as a real case study, some of the proposed algorithms in PhD Thesis has been applied to the educational field, discovering interesting relations that are highly interesting for instructors in order to improve their teaching work.

8.2 Future Research

In the following sections, some possible paths of extensions and future research lines, which are based on this PhD Thesis, are presented.

8.2.1 Multi-Relational ARM

With the growing interest in the storage of information, databases have become essential [96]. Whereas the extraction of association rules in a single relation or dataset is a well–studied topic, only a few proposals have been made for mining association rules mined from relational databases [97–99]. Relational databases have a more complex structure and store more information than raw datasets. However, the existing ARM proposals for mining rules in single relations cannot be directly applied. Instead, data have to be transformed by joining all the relations into a single relation [100]. Then, classical ARM algorithms could be successfully applied to this relation. Nevertheless, this transformation technique suffers from serious disadvantages: (1) a high computational time, and (2) a total lack of preservation of the support.

We plan to explore the problems of mining ARs in relational databases by means of G3P, where each individual could be encoded in the form of a tree through the use of a CFG. By using trees, each node could represent a relation and the edges might represent relationships between keys in two relations. In such a way, it may not required to join relations into a single table and a direct mining could be carried out over the original database.

8.2.2 Context-Aware Association Rules

In some learning processes, contextual features are explicitly provided by the problem under study, and these features could be used as a preprocessing step of classification algorithms in order to achieve more specialized and accurate classifiers [101]. Contextual features are those whose behavior depends on contextual information without interfering in a problem explicitly. Contextual features could be considered as adjustable filters for giving the right meaning in the current context, for instance, the concept of water for a thirsty person is completely different to this concept for a plumber, a builder or even a weather forecaster. However, a person could be any of these at different times

Unfortunately, contextual information is often omitted and efficient algorithms are required to detect hidden contexts [102] or even switch from context to context without being explicitly informed about it. The problem of mining hidden contexts

is still under study. Existing proposals require a fixed number of contexts and features to be mined and thus some previous knowledge of the dataset is required.

We plan to propose a G3P approach for mining contextual features. The algorithm should provide the possibility of satisfying the context mining requirements (discovering hidden contexts and irrelevant features) with the features of G3P.

8.2.3 Subgroup Discovery

In this PhD Thesis we have proposed G3P models for mining ARs, which is a descriptive task. Nowadays, there are also tasks that lie in the intersection between predictive and descriptive models, such as the so-called subgroup discovery (SD).

The SD task was originally proposed by *Wrobel et al.* [103], who defined it as: "*given a population of individuals (customers, objects, etc.) and a property of those individuals that we are interested in, the task of subgroup discovery is to find population subgroups that are statistically most interesting for the user, e.g., subgroups that are as large as possible and have the most unusual statistical characteristics with respect to a target attribute of interest*".

Since its definition, many researchers have studied the SD task and they have presented several algorithms [104–107]. Some of these algorithms were designed as adaptations of existing ARM algorithms, being comprehensibility a major requirement for this type of algorithms. Thus, the use of grammars to represents solutions could be a good starting point for SD. Therefore, we find it interesting to develop a G3P-based algorithm for SD that could be able to discover interesting subgroups for the user. The result would consist of comprehensible rules in a similar way that we have done in this PhD Thesis for mining ARs.

8.2.4 Exceptional Models in ARM

Sometimes, it is interesting to discover small subsets of data, also including those subsets whose distribution is exceptionally different from that of the entire data. Hence, the concept of exceptional model mining (EMM) was introduced by *Leman et al.* [108] as a framework for the SD task.

EMM enables the extraction of exceptional behaviour in data by identifying subgroups where a model fitted to a subgroup is somehow exceptional. Actually, this technique is mainly considered as a supervised learning task, ascertaining exceptional subgroups within predefined models but not providing any descriptive information. Nevertheless, the description of the discovered knowledge is a duty, specially for the analysis of quality measures. At this point, we consider to deal with the EMM problem under an unsupervised learning task, not requiring labeled data to induce exceptional subgroups and enabling the discovery of reliable, abnormal and exceptional relations between item-sets. Thus, the applicability of EMM to the ARM field could give rise to a new way of considering relations between items, specially in RAR mining where the searching for the abnormality is a dare.

We propose to deal RAR and EMM as a whole by means of G3P. To do so, we suggest the discovery of unusual and reliable ARs whose distribution is exceptional. This synergy brings about an interesting way of dealing with exceptional and rare relations, considering EMM as an unsupervised learning task.

8.3 Related Publications

In this section, a compilation of the most relevant publications obtained during the execution of this PhD Thesis are listed next.

8.3.1 International Journals

1. J. M. Luna, J. R. Romero and S. Ventura. *Design and Behaviour Study of a Grammar Guided Genetic Programming Algorithm for Mining Association Rules*. Knowledge and Information Systems, Vol. 32, Issue 1 (2012), pp. 53-76. Impact factor (JCR 2011): 2.225

2. J. M. Luna, J. R. Romero and S. Ventura. *On the Adaptability of G3PARM to the Extraction of Rare Association Rules*. Knowledge and Information Systems, Accepted (2012). DOI: 10.1007/s10115-012-0591-9. Impact factor (JCR 2011): 2.225

3. J. M. Luna, J. R. Romero and S. Ventura. *Grammar-Based Multi-Objective Algorithms for Mining Association Rules*. Knowledge and Information Systems, Data Knowledge and Engineering, Vol. 86, pp. 19-37, 2013. Impact factor (JCR 2012): 1.519

4. J. M. Luna, J. R. Romero and S. Ventura. *An Evolutionary Self-Adaptive Algorithm for Mining Association Rules*. IEEE Transactions on Cybernetics, Submitted June, 2013. Impact factor (JCR 2012): 3.236

5. J. M. Luna, J. R. Romero, C. Romero and S. Ventura. *Mining optimised quantitative association rules using a genetic programming free-parameter algorithm*. Integrated Computer-Aided Engineering, Submitted June, 2013. Impact factor (JCR 2012): 3.370

6. C. Romero, A. Zafra, J. M. Luna and S. Ventura. *Association rule mining using genetic programming to provide feedback to instructors from multiple-choice quiz data*. Expert Systems, Volume 30, Issue 2 (May 2013), pp. 162-172. Impact factor (JCR 2012): 0.769

7. J. M. Luna, C. Romero, J. R. Romero and S. Ventura. *An Evolutionary Algorithm for the Discovery of Rare Class Association Rules in Learning Management Systems*. Applied Soft Computing, Submitted May, 2012. Impact factor (JCR 2012): 2.140

8.3.2 International Conferences

1. J. M. Luna, J. R. Romero and S. Ventura. *Analysis of the Effectiveness of G3PARM Algorithm*. Proceedings of the 5th International Conference on Hybrid Artificial Intelligence Systems, HAIS 2010. pp. 27-34, ISBN: 978-3-642-13802-7

2. J. M. Luna, J. R. Romero and S. Ventura. *Mining and Representing Rare Association Rules through the Use of Genetic Programming*. Proceedings of the 3rd World Congress on Nature and Biologically Inspired Computing, NaBIC 2011, pp. 86-91, ISBN: 978-145771123-7

3. J.M. Luna, J.R. Romero and S. Ventura. *G3PARM: A Grammar Guided Genetic Programming Algorithm for Mining Association Rules*. Proceedings of the IEEE Congress on Evolutionary Computation, IEEE CEC 2010, pp. 2586-2593, ISBN: 978-1-4244-6910-8

4. J.M. Luna, J.R. Romero, C. Romero and S. Ventura. *A Genetic Programming Free-Parameter Algorithm for Mining Association Rules*. Proceedings of the 12th International Conference on Intelligent Systems Design and Applications, ISDA 2012, pp. 64-69, ISBN: 978-1-4673-5118-8

5. C. Romero, J.M. Luna, J.R. Romero and S. Ventura. *Mining Rare Association Rules from e-Learning Data*. Proceedings of the 3rd International Conference on Educational Data Mining, EDM 2010, pp. 171-180, ISBN: 978-0-615-37529-8

8.3.3 National Conferences

1. J.M. Luna, J.L. Olmo, J.R. Romero y S. Ventura. *Minería de reglas de asociación poco frecuentes con programación genética*. VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, MAEB 2012, pp. 181-188. ISBN: 978-84-615-6931-1

2. J.M. Luna, J.L. Olmo, J.R. Romero y S. Ventura. *Minería de reglas de asociación con programación genética gramatical*. XV Congreso Español sobre Tecnologías y Lógica Fuzzy, pp. 241-248 ISBN: 978-84-92944-02-6

Bibliography

[1] U. Usama Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From Data Mining to Knowledge Discovery in Databases," *AI Magazine*, vol. 17, pp. 37–54, 1996.

[2] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus, "Knowledge Discovery in Databases: An Overview," *AI Magazine*, vol. 13, no. 3, pp. 57–70, 1992.

[3] C. Romero, J. M. Luna, J. R. Romero, and S. Ventura, "RM-Tool: A framework for discovering and evaluating association rules," *Advances in Engineering Software*, vol. 42, no. 8, pp. 566–576, 2011.

[4] J. L. Olmo, J. R. Romero, and S. Ventura, "Using Ant Programming Guided by Grammar for Building Rule-Based Classifiers," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 6, pp. 1585–1599, 2011.

[5] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD Conference, Washington, DC, 1993, pp. 207–216.

[6] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[7] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach," *Data Mining and Knowledge Discovery*, vol. 8, pp. 53–87, 2004.

[8] X. Yan, C. Zhang, and S. Zhang, "ARMGA: Identifying Interesting Association Rules with Genetic Algorithms," *Applied Artificial Intelligence*, vol. 19, no. 7, pp. 677–689, 2005.

[9] J. Alcalá-Fdez, N. Flugy-Papé, A. Bonarini, and F. Herrera, "Analysis of the Effectiveness of the Genetic Algorithms based on Extraction of Association Rules," *Fundamenta Informaticae*, vol. 98, no. 1, pp. 1001–1014, 2010.

[10] A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag Berlin Heidelberg, 2002.

[11] A. Salleb-Aouissi, C. Vrain, and C. Nortet, "QuantMiner: A Genetic Algorithm for Mining Quantitative Association Rules," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, ser. IJCAI'97, Hyberadad, India, 2007, pp. 1035–1040.

[12] B. Alatas and E. Akin, "An Efficient Genetic Algorithm for Automated Mining of Both Positive and Negative Quantitative Association Rules," *Soft Computing*, vol. 10, pp. 230–237, 2006.

[13] Y. S. Koh and N. Rountree, *Rare Association Rule Mining and Knowledge Discovery: Technologies for Infrequent and Critical Event Detection*. Hershey, New York: Information Science Reference, 2010.

[14] L. Geng and H. J. Hamilton, "Interestingness Measures for Data Mining: A Survey," *ACM Computing Surveys*, vol. 38, 2006.

[15] F. Berzal, I. Blanco, D. Sánchez, and M. A. Vila, "Measuring the Accuracy and Interest of Association Rules: A new Framework," *Intelligent Data Analysis*, vol. 6, no. 3, pp. 221–235, 2002.

[16] J. Mata, J. L. Álvarez, and J. C. Riquelme, "Discovering numeric association rules via evolutionary algorithm," in *Proceeding of the 6th International Conference on Knowledge Discovery and Data Mining*, ser. PAKDD '02, 2002, pp. 40–51.

[17] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[18] C. Zhang and S. Zhang, *Association rule mining: models and algorithms*. Springer Berlin / Heidelberg, 2002.

[19] Y. S. Koh and N. Rountree, "Finding Sporadic Rules using Apriori-Inverse," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3518, pp. 97–106, 2005.

[20] G. Piatetsky-Shapiro, "Discovery, analysis and presentation of strong rules," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. Frawley, Eds. AAAI Press, 1991, pp. 229–248.

[21] N. Lavrac, P. A. Flach, and B. Zupan, "Rule evaluation measures: A unifying view," in *Proceedings of the 9th International Workshop on Inductive Logic Programming*, ser. ILP '99. London, UK: Springer-Verlag, 1999, pp. 174–185.

[22] P. Tan and V. Kumar, "Interestingness Measures for Association Patterns: A Perspective," in *Proceedings of the Workshop on Postprocessing in Machine Learning and Data Mining*, ser. KDD '00, New York, USA.

[23] C. Ordoñez, N. Ezquerra, and C. Santana, "Constraining and Summarizing Association Rules in Medical Data," *Knowledge and Information Systems*, vol. 9, no. 3, pp. 259–283, 2006.

[24] D. Malerba, F. A. Lisi, and F. Sblendorio, "Mining spatial association rules in census data: a relational approach," in *Proceeding of the ECML/PKDDâĂô02 workshop on mining official data*. University Printing House, Helsinki: Springer-Verlag, 2002, pp. 80–93.

[25] A. Rahman, C. I. Ezeife, and A. K. Aggarwal, "Wifi miner: An online apriori-infrequent based wireless intrusion system," in *Proceedings of the 2nd International Workshop in Knowledge Discovery from Sensor Data*, ser. Sensor-KDD '08, Las Vegas, USA, 2008, pp. 76–93.

[26] T. Li and X. Li, "Novel Alarm Correlation Analysis System based on Association Rules Mining in Telecommunication Networks," *Information Sciences*, vol. 180, no. 16, pp. 2960–2978, 2010.

[27] D. Sánchez, J. M. Serrano, L. Cerda, and M. A. Vila, "Association Rules Applied to Credit Card Fraud Detection," *Expert systems with applications*, no. 36, pp. 3630–3640, 2008.

[28] C. Romero and S. Ventura, "Educational data mining: A survey from 1995 to 2005," *Expert Systems with Applications*, vol. 33, no. 1, pp. 135–146, 2007.

[29] ——, "Educational data mining: a review of the state of the art," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 40, no. 6, pp. 601–618, 2010.

[30] S. Ha, S. Bae, and S. Park, "Web mining for distance education," in *Proceedings of the 2000 IEEE International Conference on Management of Innovation and Technology*, ser. ICMIT 2000, Singapore, 2000, pp. 715–719.

[31] J. Li and O. Zaïane, "Combining usage, content, and structure data to improve web site recommendation," in *Proceedings of the 5th International Conference on e-Commerce and Web Technologies*, ser. EC-Web 2004, Zaragoza, Spain, 2004, pp. 305–315.

[32] Y. Psaromiligkos, M. Orfanidou, C. Kytagias, and E. Zafiri, "Mining log data for the analysis of learners' behaviour in web-based learning management systems," *Operational Research Journal*, vol. 11, pp. 1–14, 2009.

[33] A. Merceron and K. Yacef, "Interestingness measures for association rules in educational data," in *Proceedings of the 1st International Conference on Educational Data Mining*, ser. EDM 2008, Montreal, Canada, 2008, pp. 57–66.

[34] C. J. Tsai, S. S. Tseng, and C. Y. Lin, "A two-phase fuzzy mining and learning algorithm for adaptive learning environment," in *Proceedings of the International Conference on Computational Science*, ser. ICCS 2001. San Francisco, USA: Springer-Verlag, 2001, pp. 429–438.

[35] A. A. Ramli, "Web usage mining using a priori algorithm: UUM learning care portal case," in *Proceedings of the International Conference on Knowledge Management*, Malaysia, 2005, pp. 1–19.

[36] D. Zhenguo, W. Qinqin, and D. Xianhua, "An improved fp-growth algorithm based on compound single linked list," in *Proceedings of the 2009 Second International Conference on Information and Computing Science*, ser. ICIC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 351–353.

[37] C. Borgelt, "Efficient Implementations of Apriori and Eclat," in *Proceeding of the 1st Workshop on Frequent Itemset Mining Implementations*, ser. FIMI'03, Melbourne, Florida, USA, 2003.

[38] R. Srikant and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables," in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD'96, Montreal, Quebec, Canada, 1996.

[39] J. Li, H. Shen, and R. Topor, "Mining the optimal class association rule set," *Knowledge-Based Systems*, vol. 15, no. 7, pp. 399 – 405, 2002.

[40] H. Zhang, Y. Zhao, L. Cao, and C. Zhang, "Class association rule mining with multiple imbalanced attributes," in *Proceedings of the 20th Australian Joint Conference on Artificial Intelligence*, Gold Coast, Australia, 2007, pp. 827–831.

[41] T. Scheffer, "Finding association rules that trade support optimally against confidence," in *Proceedings of the 5th European Conference of Principles and Practice of Knowledge Discovery in Databases*, ser. PKDD 2001, Freiburg, Germany, 2001, pp. 424–435.

[42] W. Lin, C. Ruiz, and S. A. Alvarez, "A new adaptive-support algorithm for association rule mining," Tech. Rep., 2000.

[43] L. Szathmary, A. Napoli, and P. Valtchev, "Towards rare itemset mining," in *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, ser. ICTAI '07, Patras, Greece, 2007, pp. 305–312.

[44] L. Szathmary, P. Valtchev, and A. Napoli, "Generating Rare Association Rules Using the Minimal Rare Itemsets Family," *International Journal of Software and Informatics*, vol. 4, no. 3, pp. 219–238, 2010.

[45] M. Martínez-Ballesteros, F. Martínez-Álvarez, A. Troncoso, and J. C. Riquelme, "An evolutionary algorithm to discover quantitative association rules in multidimensional time series," *Soft Computing*, vol. 15, pp. 2065–2084, 2011.

[46] J. L. Olmo, J. M. Luna, J. R. Romero, and S. Ventura, "Mining association rules with single and multi-objective grammar guided ant programming," *Integrated Computer-Aided Engineering*, vol. 20, no. 3, pp. 217–234, 2013.

[47] R. J. Kuo, C. M. Chao, and Y. T. Chiu, "Application of particle swarm optimization to association rule mining," *Applied Soft Computing*, vol. 11, no. 1, pp. 326–336, 2011.

[48] J. Mata, J. L. Alvarez, and J. C. Riquelme, "Mining Numeric Association Rules via Evolutionary Algorithm," in *ICANNGA'01, Proceedings of the 5th International Conference on Artificial Neural Networks and Genetic Algorithms, Prague, Czech Republic*, April 2001, pp. 264–267.

[49] M. Delgado, N. Marín, D. Sánchez, and M. A. Vila, "Fuzzy association rules: general model and applications," *IEEE Transactions on Fuzzy Systems*, vol. 11, pp. 214–225, 2003.

[50] R. Alcalá, J. Alcala-Fdez, M. J. Gacto, and F. Herrera, "Genetic learning of membership functions for mining fuzzy association rules," in *Proceedings of the 16th IEEE International Conference on Fuzzy Systems*, ser. FUZZ-IEEE 2007, London, UK, 2007, pp. 1538–1543.

[51] L. J. Eshelman, "The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination," in *Foundations of Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann, 1991, pp. 265–283.

[52] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin, *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[53] P. González-Espejo, S. Ventura, and F. Herrera, "A Survey on the Application of Genetic Programming to Classification," *IEEE Transactions on Systems, Man and Cybernetics: Part C*, vol. 40, no. 2, pp. 121–144, 2010.

[54] A. Tsakonas, G. Dounias, J. Jantzen, H. Axer, B. Bjerregaard, and D. G. von Keyserlingk, "Evolving rule-based systems in two medical domains using genetic programming," *Artificial Intelligence in Medicine*, vol. 32, no. 3, pp. 195–216, 2004.

[55] A. Ratle and M. Sebag, "Genetic programming and domain knowledge: Beyond the limitations of grammar-guided machine discovery," in *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, ser. PPSN VI, Paris, France, September 2000, pp. 211–220.

[56] M. L. Wong and K. S. Leung, *Data Mining Using Grammar-Based Genetic Programming and Applications*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[57] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*, 1st ed. A Bradford Book, 1992.

[58] R. McKay, N. Hoai, P. Whigham, Y. Shan, and M. O'Neill, "Grammar-based Genetic Programming: a Survey," *Genetic Programming and Evolvable Machines*, vol. 11, pp. 365–396, 2010.

[59] P. A. Whigham, "Grammatically-based genetic programming," in *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, Tahoe City, California, USA, 1995, pp. 33–41.

[60] A. Geyer-Schulz, *Fuzzy Rule-Based Expert Systems and Genetic Machine Learning*, ser. Studies in Fuzziness. Heidelberg: Physica-Verlag, 1995, vol. 3.

[61] M. L. Wong and K. S. Leung, "Evolutionary program induction directed by logic grammars," *Evolutionary Computation*, vol. 5, no. 2, pp. 143–180, 1997.

[62] A. Zafra, C. Romero, S. Ventura, and E. Herrera-Viedma, "Multi-instance genetic programming for web index recommendation," *Expert Systems with Applications*, vol. 36, no. 9, pp. 11 470–11 479, 2009.

[63] A. Zafra and S. Ventura, "G3P-MI: A genetic programming algorithm for multiple instance learning," *Information Sciences*, vol. 180, no. 23, pp. 4496–4513, 2010.

[64] A. Cano, A. Zafra, E. L. Gibaja, and S. Ventura, "A grammar-guided genetic programming algorithm for multi-label classification," in *Proceedings of the 16th European Conference on Genetic Programming*, ser. EuroGP 2013, Vienna, Austria, 2013, pp. 217–228.

[65] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, 1st ed. Wiley, June 2001.

[66] C. A. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed. Berlin: Springer-Verlag, 2007.

[67] B. Alatas, E. Akin, and A. Karci, "MODENAR: Multi-objective Differencial Evolution Algorithm for Mining Numeric Association Rules," *Applied Soft Computing*, vol. 8, pp. 646–656, 2008.

[68] H. R. Qodmanan, M. Nasiri, and B. Minaei-Bidgoli, "Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence," *Expert Systems with Applications*, vol. 38, no. 1, pp. 288–298, 2011.

[69] A. Ghosh and B. Nath, "Multi-objective Rule Mining Using Genetic Algorithms," *Information Science*, vol. 163, no. 1-3, pp. 123–133, 2004.

[70] R. Price and K. Storn, "Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[71] R. Anand, A. Vaid, and P. K. Singh, "Association rule mining using multi-objective evolutionary algorithms: Strengths and challenges," in *Proceedings of the 2009 World Congress on Nature and Biologically Inspired Computing*, Coimbatore, India, 2009, pp. 385–390.

[72] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2000.

[73] x. Yan, D. Zhang, and S. Zhang, "Genetic Algorithm-based Strategy for Identifying Association Rules without Specifying Actual Minimum Support," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3066–3076, 2009.

[74] S. Ventura, C. Romero, A. Zafra, J. A. Delgado, and C. Hervás, "JCLEC: A Framework for Evolutionary Computation," *Soft Computing*, vol. 12, pp. 381–392, 2007.

[75] H. Liu, F. Hussain, C. L. Tan, and M. Dash, "Discretization: An Enabling Technique," *Data Mining and Knowledge Discovery*, vol. 6, pp. 2393–423, 2002.

[76] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[77] S. García, D. Molina, M. Lozano, and F. Herrera, "A Study on the Use of Non-parametric Tests for Analyzing the Evolutionary Algorithms' Behaviour: a Case Study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

[78] L. Martin, R. Leblanc, and N. K. Toan, "Tables for the Friedman Rank Test," *The Canadian Journal of Statistics*, vol. 21, no. 1, pp. 39–43, 1993.

[79] J. M. Luna, J. R. Romero, and S. Ventura, "Design and behavior study of a grammar-guided genetic programming algorithm for mining association rules," *Knowledge and Information Systems*, vol. 32, no. 1, pp. 53–76, 2012.

[80] ——, "Analysis of the Effectiveness of G3PARM Algorithm," in *Proceedings of the 5th International Conference on Hybrid Artificial Intelligent Systems, HAIS'10*, 2010, pp. 27–34.

[81] ——, "On the adaptability of g3parm to the extraction of rare association rules," *Knowledge and Information Systems*, 2012.

[82] ——, "Mining and Representing Rare Association Rules through the Use of Genetic Programming," in *Proceedings of the 3rd World Congress on Nature and Biologically Inspired Computing, NaBIC 2011*, 2011, pp. 86–91.

[83] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *Proceedings of the 2001 conference on Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems*, 2002, pp. 95–100.

[84] J. M. Luna, J. R. Romero, and S. Ventura, "Grammar-based multi-objective algorithms for mining association rules," *Data & Knowledge Engineering*, vol. 86, pp. 19–37, 2013.

[85] ——, "G3PARM: A Grammar Guided Genetic Programming Algorithm for Mining Association Rules," in *Proceedings of the IEEE Congress on Evolutionary Computation, IEEE CEC 2010*, 2011, pp. 2586–2593.

[86] G. Dick and P. Whigham, "Spatially-Structured Evolutionary Algorithms and Sharing: Do They Mix?" in *Simulated Evolution and Learning*, ser. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2006, vol. 4247, pp. 457–464.

[87] ——, "Spatially-Structured Sharing Technique for Multimodal Problems," *Journal of Computer Science and Technology*, vol. 23, pp. 64–76, 2008.

[88] J. Z. C. Laia and T. J. Huang, "An agglomerative clustering algorithm using a dynamic k-nearest-neighbor list," *Information Sciences*, vol. 181, no. 9, pp. 1722–1734, 2011.

[89] W. De Mulder, "Optimal clustering in the context of overlapping cluster analysis," *Information Sciences*, vol. 223, no. 0, pp. 56–74, 2013.

[90] J. L. Olmo, J. M. Luna, J. R. Romero, and S. Ventura, "Mining association rules with single and multi-objective grammar guided ant programming," *Integrated Computer Aided Engineering*, vol. 20, no. 3, 2013.

[91] J. M. Luna, J. R. Romero, C. Romero, and S. Ventura, "A Genetic Programming Free-Parameter Algorithm for Mining Association Rules," in *roceedings*

of the 12th International Conference on Intelligent Systems Design and Applications, ser. ISDA 2012, Kochi, India, 2012, pp. 64–69.

[92] P. Brusilovsky and P. Miller, "Web-based testing for distance education," in *Proceedings of the World Conference of WWW and Internet*, Orlando, Florida, 1998, pp. 149–154.

[93] W. L. Kuechler and M. G. Simkin, "How well do multiple choice tests evaluate student understanding in computer programming classes?" *Journal Information System Educaction*, vol. 14, pp. 389–399, 2003.

[94] C. Romero, A. Zafra, J. M. Luna, and S. Ventura, "Association rule mining using genetic programming to provide feedback to instructors from multiple-choice quiz data," *Expert Systems*, vol. in press, doi: 10.1111/j.1468-0394.2012.00627.x, 2012.

[95] R. Romero, J. M. Luna, J. R. Romero, and S. Ventura, "ining rare association rules from e-learning data," in *Proceedings of the 3rd International Conference on Educational Data Mining*, ser. EDM 2010, Pittsburgh, USA, 2010, pp. 171–180.

[96] A. R. Konan, T. I. Gundem, and M. E. Kaya, "Assignment query and its implementation in moving object databases," *International Journal of Information Technology and Decision Making*, vol. 9, no. 3, pp. 349–372, 2010.

[97] A. Alashqur, "RDB-MINER: A SQL-Based Algorithm for Mining Rrue Relational Databases," *Journal of Software*, vol. 5, no. 9, pp. 998–1005, 2010.

[98] B. Goethals, W. Le Page, and M. Mampaey, "Mining Interesting Sets and Rules in Relational Databases," in *Proceedings of the ACM Symposium on Applied Computing*, Sierre, Switzerland, 2010, pp. 997–1001.

[99] A. Jiménez, F. Berzal, and J. C. Cubero, "Using Trees to Mine Multirelational Databases," *Data Mining and Knowledge Discovery*, pp. 1–39, 2011.

[100] E. Ng, A. Fu, and K. Wang, "Mining association rules from stars," in *Proceedings of the 2002 IEEE International Conference on Data Mining*, ser. ICDM 2002, Maebashi City, Japan, 2002.

[101] P. Brézillon, "Context in problem solving: a survey," *The Knowledge Engineering Review*, vol. 14, no. 01, pp. 47–80, 1999.

[102] G. Widmer, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.

[103] S. Wrobel, "An algorithm for multi-relational discovery of subgroups," in *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, ser. PKDD '97. London, UK, UK: Springer-Verlag, 1997, pp. 78–87.

[104] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski, "Subgroup discovery with cn2-sd," *Journal of Machine Learning Research*, vol. 5, pp. 153–188, Dec. 2004.

[105] B. Kavsek and N. Lavrac, "APRIORI-SD: Adapting association rule learning to subgroup discovery," *Applied Artificial Intelligence*, vol. 20, no. 7, pp. 543–583, 2006.

[106] M. J. del Jesus, P. González, F. Herrera, and M. Mesonero, "Evolutionary fuzzy rule induction process for subgroup discovery: A case study in marketing," *Fuzzy Systems, IEEE Transactions on*, vol. 15, no. 4, pp. 578–592, aug. 2007.

[107] C. J. Carmona, P. González, M. J. del Jesus, and F. Herrera, "NMEEF-SD: Non-dominated multiobjective evolutionary algorithm for extracting fuzzy rules in subgroup discovery," *Fuzzy Systems, IEEE Transactions on*, vol. 18, no. 5, pp. 958–970, oct. 2010.

[108] D. Leman, A. Feelders, and A. J. Knobbe, "Exceptional model mining," in *Proceedings of the European Conference in Machine Learning and Knowledge Discovery in Databases*, ser. ECML/PKDD 2008, vol. 5212. Antwerp, Belgium: Springer, 2008, pp. 1–16.