

UNIVERSIDAD DE GRANADA



Departamento de Ciencias de la Computación
e Inteligencia Artificial

*Hybrid and Constructive Metaheuristics:
Methods and Applications*

Tesis Doctoral

Francisco Javier Rodríguez Díaz

Granada, Noviembre de 2012

Editor: Editorial de la Universidad de Granada
Autor: Francisco Javier Rodríguez Díaz
D.L.: GR 856-2013
ISBN: 978-84-9028-453-7

UNIVERSIDAD DE GRANADA



*Hybrid and Constructive Metaheuristics:
Methods and Applications*

MEMORIA QUE PRESENTA

Francisco Javier Rodríguez Díaz

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

Noviembre de 2012

DIRECTORES

Dr. Manuel Lozano Márquez

Dr. Carlos García Martínez

Departamento de Ciencias de la Computación
e Inteligencia Artificial

La memoria titulada "*Hybrid and Constructive Metaheuristics: Methods and Applications*", que presenta D. Francisco Javier Rodríguez Díaz para optar al grado de doctor, ha sido realizada dentro del Máster Oficial de Doctorado "*Soft Computing y Sistemas Inteligentes*" del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección de los doctores D. Manuel Lozano Márquez y D. Carlos García Martínez.

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

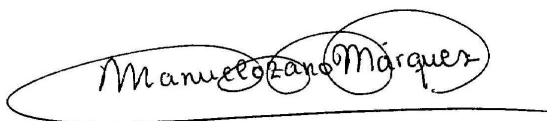
Granada, Noviembre de 2012

El Doctorando



Fdo: Francisco Javier Rodríguez Díaz

Los Directores



Fdo: Manuel Lozano Márquez



Fdo: Carlos García Martínez

Esta tesis doctoral ha sido desarrollada bajo la financiación de los fondos asociados al proyecto P08-TIC-4173 de la Junta de Andalucía y los proyectos TIN2007-66523 y TIN2011-24124 del Ministerio de Economía y Competitividad.

Table of Contents

- I. PhD dissertation** **1**
- 1. Introducción 1
- 1. Introduction 5
- 2. Problem statement 9
 - 2.1. Optimisation 9
 - 2.2. Black-box problems 10
 - 2.3. Parallel Machines Scheduling Problem 10
 - 2.4. Quadratic Minimum Spanning Tree Problem 13
 - 2.5. Metaheuristics 13
 - 2.5.1. Simulated Annealing 15
 - 2.5.2. Evolutionary Algorithms 16
 - 2.5.3. Greedy Randomised Adaptive Search 17
 - 2.5.4. Iterated Greedy 18
 - 2.6. Hybrid Metaheuristics 19
- 3. Justification 21
- 4. Objectives 23
- 5. Joint Discussion of Results 24
 - 5.1. A Simulated Annealing Method Based on a Specialised Evolutionary Algorithm 24
 - 5.2. Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test 26
 - 5.3. GRASP with Path-Relinking for the Non-Identical Parallel Machine Scheduling Problem with Minimising Total Weighted Completion Times 29
 - 5.4. An Iterated Greedy Algorithm for the Large-Scale Unrelated Parallel Machines Scheduling Problem 31
 - 5.5. Tabu Search with Strategic Oscillation for the Quadratic Minimum Spanning Tree 33
- 6. Conclusiones 35
- 6. Conclusions 37

7.	Future Work	39
II.	Publications: Published, Accepted and Submitted Papers	41
1.	A simulated annealing method based on a specialised evolutionary algorithm	41
2.	Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test	59
3.	GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times	75
4.	An Iterated Greedy Algorithm for the Large-Scale Unrelated Parallel Machines Scheduling Problem	95
5.	Tabu Search with Strategic Oscillation for the Quadratic Minimum Spanning Tree .	131
	Bibliografia	151

Table of Figures

- 1. Space of I&D [BR03] 14
- 2. SA model 15
- 3. Basic scheme of an EA 17
- 4. Basic scheme of GRASP 17
- 5. Basic scheme of GRASP construction phase 18
- 6. Basic scheme of IG 19
- 7. Number of publications and citations per year for HMs-EA/SA (Web of Science) . . 22
- 8. SASEA model 24
- 9. Average rankings of the HMs-EA/SA versions 28
- 10. Synergy study: HMs-EA/SA vs. Standalone SA, Canonical GAs, and CHC 29

Part I. PhD dissertation

1. Introducción

Un problema de optimización consiste en la selección de la mejor configuración para un conjunto de variables de acuerdo a unos determinados criterios. Los problemas de optimización se pueden clasificar básicamente en dos categorías dependiendo de si las variables son reales o discretas. Dentro de los problemas de optimización con variables discretas, encontramos los denominados *problemas de optimización combinatoria*. De acuerdo a [PS82], en los problemas de optimización combinatoria buscamos un determinado objeto como un entero, una permutación o un grafo de entre un conjunto finito (o posiblemente infinito numerable).

Debido a la importancia de los problemas de optimización en la industria y en la ciencia, la comunidad científica ha realizado grandes esfuerzos para desarrollar nuevos algoritmos para afrontar esta clase de problemas. Estos algoritmos se han aplicado para resolver incontables *aplicaciones reales*. Solo por nombrar algunas de estas aplicaciones, podemos destacar que estos algoritmos son usados por las compañías aéreas para planificar las rutas aéreas y las políticas de precios, por las grandes compañías para planificar la localización de sus almacenes, por los procesadores de texto para decidir donde introducir espacios en blanco para justificar un determinado párrafo y por las tiendas de comercio electrónico para recomendar productos a sus clientes.

Los métodos existentes para optimización combinatoria se pueden clasificar en dos categorías: *métodos exactos* y *algoritmos aproximados*. Los métodos exactos garantizan que se va a encontrar una solución óptima para cada instancia finita del problema en un tiempo limitado [PS82, NW88]. Sin embargo, muchos de los problemas que aparecen son NP duros, lo que hace difícil que se puedan diseñar algoritmos exactos eficientes para dichos problemas. Por lo tanto, los algoritmos aproximados, que se centran en encontrar buenas soluciones en un tiempo reducido, han incrementado su importancia en los últimos años.

Entre los algoritmos aproximados, las *metaheurísticas* [BR03, GK03, SM08] se han establecido como una de las alternativas más prometedoras para afrontar los problemas de optimización combinatoria. Las metaheurísticas son una familia de algoritmos que constan de un proceso iterativo que guía una heurística subordinada de forma que se combinan de forma inteligente conceptos para la exploración y explotación del espacio de búsqueda asociado al problema de optimización combinatorio. Las metaheurísticas se han aplicado para resolver problemas dentro de diversos campos, mostrando su habilidad para proporcionar buenas soluciones (no necesariamente óptimas) en un tiempo razonable. Existe un grupo de metaheurísticas que siguen paradigmas diferenciados y son conocidas como metaheurísticas clásicas, ya que tienen un amplio bagaje histórico. Este gru-

po está formados por métodos como *enfriamiento simulado* [KGV83] (SA), *búsqueda tabú* [GL97], *búsqueda local iterativa* [LMS03], *búsqueda en entornos variables* [MH97], *procedimiento de búsqueda voraz aleatorizado* [FR95] (GRASP), *algoritmos evolutivos* [BFM97] (EAs) y *búsqueda dispersa* [Lag03].

Además, durante los últimos años, ha aparecido un grupo de algoritmos que se caracteriza porque no se limita a seguir los conceptos de una única metaheurística clásica sino que combina lo mejor de distintas metaheurísticas (e incluso otros métodos de optimización) de forma que la combinación de las mismas se comporte mejor y dicha combinación produzca una sinergia positiva. Estas aproximaciones se conocen como *metaheurísticas híbridas* (HMs) [Tal02, Rai06a, Blu10]. El éxito y la importancia de esta línea de investigación se ve reflejado en el número creciente de aplicaciones de las metaheurísticas híbridas y en la existencia de eventos científicos centrados en esta temática como la serie *Workshops on Hybrid Metaheuristics*.

SA fue presentado en 1983, siendo una de las primeras propuestas de lo que conocemos como metaheurísticas clásicas. Sin embargo, SA es todavía objeto de estudio por parte de muchos trabajos y se aplica para resolver problemas de optimización bien por si solo o como componente de otros algoritmos de búsqueda [HJJ03, SSF02, GML09b]. SA es conocido por ser el primer algoritmo que extendía los métodos de búsqueda local añadiendo una estrategia para escapar de óptimos locales. Los algoritmos de búsqueda locales realizan un proceso de búsqueda iterativo que trata de mejorar la solución actual mediante la exploración de soluciones cercanas a la misma. La idea principal de SA es permitir movimientos hacia soluciones de peor calidad que la solución actual. La probabilidad de realizar estos movimientos hacia peores soluciones va decreciendo a lo largo del proceso de búsqueda.

Los EAs [BFM97, ES03] son actualmente una de las metaheurísticas más populares y son usados por muchos investigadores para resolver problemas complejos. Son métodos estocásticos de búsqueda que tratan de imitar el proceso de evolución natural y están basados en el concepto de una población de individuos (los cuales representan puntos del espacio de búsqueda de un determinado problema), utilizando operadores probabilísticos tales como la mutación, selección, y (a veces) cruce para evolucionar hacia individuos con mejores valores de adaptación. Existe una gran variedad de EAs ligeramente diferentes como la *programación evolutiva* [Fog95], las *estrategias de evolución* [BS02] y los *algoritmos genéticos* [Gol89]. Los EAs presentan grandes ventajas cuando se afrontan problemas de optimización complejos, ya que pueden localizar prometedoras regiones dentro de espacios de búsqueda muy grandes y complejos. Otras ventajas son su simplicidad, flexibilidad y robustez para responder a situaciones cambiantes.

La hibridación de los EAs está adquiriendo gran popularidad por su habilidad para tratar con problemas reales caracterizados por su complejidad, presencia de ruido, imprecisión, incertidumbre y vaguedad [GA07a, PT99, MS10, VRH09]. En particular, la *combinación de SA y EAs para diseñar HMs* destaca por su importancia dentro del campo de las hibridaciones de los EAs [TB05, LZXM10, WWR05, CWYH09, TW10].

Aparte de las metaheurísticas basadas en la búsqueda de nuevas soluciones cercanas a la actual, como SA, o en la combinación de varias soluciones, como los EAs, encontramos otro grupo de metaheurísticas que se basan en la construcción de nuevas soluciones añadiendo en cada paso un nuevo elemento a una solución parcial. Este tipo de metaheurísticas son conocidas como metaheurísticas constructivas y podemos encontrar en la literatura diversos ejemplos como la *búsqueda voraz iterativa* (IG) [JB95], *GRASP* [FR95] y la *optimización basada en colonias de hormigas* [DS04]. Los métodos constructivos son usualmente los más rápidos de entre los métodos aproximados, sin embargo, a menudo la calidad de las soluciones encontradas nos es tan buena como en los métodos de búsqueda local [CHS02]. Por lo tanto, la mayoría de las metaheurísticas constructivas incluyen

métodos de búsqueda local para mejorar la calidad de las soluciones obtenidas después de la etapa de construcción.

GRASP [FR89, FR95] es una metaheurística multiarranque para optimización combinatoria que consta de dos fases: construcción y fase de refinamiento. El mecanismo de construcción de soluciones proporciona una solución inicial usando una heurística aleatorizada, permitiendo de esta manera que se obtengan soluciones en diferentes áreas del espacio de búsqueda. Cada solución se obtiene añadiendo, paso a paso, a una solución parcial un nuevo elemento seleccionado de entre un conjunto restringido de candidatos. La fase de refinamiento parte de la solución obtenida anteriormente y realiza perturbaciones locales para obtener una solución localmente óptima con respecto a un vecindario predefinido. Este proceso formado por estas dos fases es iterativo y se realiza hasta que se alcanza una condición de parada definida por el usuario. Al final de este proceso iterativo, el algoritmo devuelve la mejor solución encontrada a lo largo de todo el proceso. GRASP es usado muy a menudo para afrontar problemas reales [RR03] debido a su simplicidad.

IG [CL96, JB95, RS07] es una metaheurística muy efectiva, desarrollada recientemente para afrontar problemas de optimización combinatoria y que sigue un sencillo principio. Además, es fácil de implementar y muestra un rendimiento excelente, de hecho, es considerada como estado del arte para un considerable número de problemas [FPR10, FL08, LLYL11, URS10]. IG, iterativamente, trata de mejorar la solución actual eliminando elementos de la solución actual y completando después la solución parcial obtenida mediante un procedimiento constructivo. Además, IG suele utilizar procedimientos de refinamiento local para mejorar la calidad de las soluciones obtenidas después de las fases de destrucción y reconstrucción.

Al mismo tiempo, existen en la actualidad múltiples trabajos que se centran en estudiar las hibridaciones de las metaheurísticas constructivas. En la literatura encontramos, entre otros ejemplos, hibridaciones entre algoritmos de colonias de hormigas y algoritmos genéticos [CK12] o lógica difusa [YLJ12], GRASP y encadenamiento de trayectorias [SCD⁺12] o optimización basada en nubes de partículas [MMD10] e IG y búsqueda en entornos variables [LB10].

Para nuestro estudio, hemos considerado problemas en dos escenarios diferentes. En el primero, no disponemos de conocimiento específico acerca del problema para usarlo a lo largo del proceso de búsqueda, excepto el proporcionado por la propia representación y las restricciones del problema. Estos problemas son conocidos como *problemas de caja negra* [CLM05, GDLM08]. En segundo lugar, afrontamos problemas en entornos en los que disponemos de conocimiento específico, tales como la estructura de la función objetivo. En particular, hemos considerado dos problemas diferentes: el problema de la planificación de trabajos en máquinas paralelas [McN59] y el problema del árbol de expansión cuadrático mínimo [AX92].

Los problemas de caja negra aparecen en multitud de aplicaciones científicas y de ingeniería donde no se dispone de información útil para reforzar el proceso de búsqueda. Estos problemas son comunes cuando se optimizan modelos dinámicos computacionalmente costosos, ingeniería de bioprocesos o cuando la evaluación de las soluciones se lleva a cabo mediante simulaciones. Los algoritmos independientes del contexto son una herramienta útil en estos casos, ya que solo requieren conocer el dominio y el tipo de las variables de decisión para poder ser aplicados. El modelo de algoritmo genético original propuesto por Holland [Hol75] pertenece a esta clase de métodos.

La planificación de trabajos en máquinas paralelas consiste en asignar un conjunto de trabajos en alguna de las máquinas disponibles, de forma que la asignación resultante satisfaga ciertos requerimientos. Dentro de la formulación general del problema, existen diversas variantes dependiendo de las características de las máquinas o las condiciones del mecanismo de planificación. Entre sus principales aplicaciones reales destaca la optimización de los procesos productivos en la industria o la planificación de procesos en ordenadores con múltiples nodos de procesamiento [Wot07].

El problema del árbol de expansión cuadrático mínimo es una extensión del conocido problema del árbol de expansión mínimo [J.B56], donde además de los costos asociados a cada arista, tenemos costos asociados a pares de aristas. Este problema se ha estudiado en profundidad en la literatura debido a su aplicación en una amplia variedad de entornos, como el transporte, las telecomunicaciones, sistemas de riego y transporte de energía. El problema aparece, por ejemplo, cuando se transfiere combustible de una tubería a otra en situaciones en las que el costo depende de la interfaz entre las dos tuberías. Esta misma interacción por parejas la encontramos en la conexión de cables subterráneos y en superficie en una red de carreteras con penalizaciones por retorno [SS10, ZG98].

En esta memoria, presentamos la investigación que hemos realizado en las dos áreas comentadas anteriormente: 1) métodos de optimización para entornos de caja negra mediante HMs que combinan SA y EAs y 2) metaheurísticas constructivas para los problemas de la planificación en máquinas paralelas y del árbol de expansión cuadrático mínimo. Con respecto a la primera tarea, hemos realizado un estudio del estado del arte en cuanto a HMs-EA/SA para optimización combinatoria, clasificando los métodos encontrados de acuerdo a la taxonomía propuesta para este tipo de métodos. Además, hemos presentado nuevos modelos de HMs-EA/SA que cubren categorías de la taxonomía en las cuales no existía ninguna HM-EA/SA en la literatura. Con respecto a la segunda tarea, hemos identificado los métodos estado del arte para el problema de la planificación de trabajos en máquinas paralelas y el problema del árbol de expansión cuadrático mínimo y hemos explorado metaheurísticas constructivas que sean competitivas con respecto a los métodos existentes en la literatura. Para llevar a cabo estas tareas, esta memoria se ha estructurado en dos partes:

- La Parte I está dedicada a la definición del problema, la discusión de los métodos propuestos y las conclusiones obtenidas.
- La Parte II contiene las publicaciones asociadas.

En la Parte I, después de la Introducción, continuamos con la definición del problema y las técnicas usadas (Sección 2), los problemas abiertos que justifican esta tesis (Sección 3) y los objetivos que nos planteamos en la misma (Sección 4). Después, en la Sección 5, resumimos los trabajos realizados a lo largo de esta tesis, resaltando los resultados y conclusiones más importantes obtenidos. Finalmente, presentamos las conclusiones globales de esta tesis y terminamos con el trabajo futuro a realizar después de esta tesis (Sección 7).

En la Parte II, con el objetivo de desarrollar los objetivos propuestos, proporcionamos un compendio de cinco publicaciones relacionados con el estudio emprendido en esta tesis:

- Un método de enfriamiento simulado basado en algoritmos evolutivos especializados.
- Metaheurísticas híbridas basadas en algoritmos evolutivos y enfriamiento simulado: taxonomía, comparativa y estudio de sinergia.
- GRASP con encadenamiento de trayectorias para el problema de la planificación en máquinas no idénticas con minimización del tiempo total de finalización ponderado.
- Un algoritmo IG para el problema de planificación en máquinas no relacionadas para alta escalabilidad.
- Búsqueda tabú con oscilación estratégica para el problema del árbol de expansión cuadrático mínimo.

1. Introduction

An optimisation problem concerns the selection of the best configuration of a set of variables according to some criteria. Optimisation problems can be basically divided into two categories depending on whether these variables are real-valued or discrete. Among the optimisation problems with discrete variables, we find a type called *combinatorial optimisation problems*. According to [PS82], in combinatorial optimisation problems we are looking for an object such as an integer, permutation or graph from a finite (or possibly countable infinite) set.

Due to the importance of combinatorial optimisation problems in science and industry, researchers have devoted great efforts to develop new algorithms to tackle these problems. Algorithms for combinatorial optimisation problems arise in countless *real-world applications*. Just to name a few, these algorithms are used by airline companies to schedule flights and decide their respective prices, by large companies to decide where and what to stock in their warehouses, by delivery companies to decide the routes for their trucks, by GPS navigators to provide driving directions, by word-processors to decide where to introduce blank spaces to justify a paragraph and by webshops to recommend products to their clients.

The existing methods for combinatorial optimisation are classified into two categories: *exact* and *approximate algorithms*. Exact methods guarantee to find an optimal solution for every finite size instance in a bounded time [PS82, NW88]. However, many problems arising in practice are NP-hard and so it is unlikely that we can design exact efficient algorithms for these problems. Thus, the use of approximate algorithms, which focus on finding good solutions in a reduced amount of time, has increased greatly in recent years.

Among approximate algorithms, *metaheuristics* [BR03, GK03, SM08] have been established as one of the most practical approaches to deal with combinatorial optimisation problems. Metaheuristics are a family of approximate methods conformed by iterative processes that guide a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space associated to the combinatorial optimisation problem. Metaheuristics have been applied to problems from very different fields, showing its ability to provide acceptably good solutions (not necessarily optimal) in a reasonable amount of time. There is a group of metaheuristics that follow distinct paradigms and are usually cited as classical metaheuristics, as they have a well-established historical background. This group includes methods such as *simulated annealing* [KGV83] (SA), *tabu search* [GL97], *iterated local search* [LMS03], *variable neighbourhood search* [MH97], *greedy randomised adaptive search procedure* [FR95] (GRASP), *evolutionary algorithms* [BFM97] (EAs), and *scatter search* [Lag03].

Furthermore, over the last few years, a large number of search algorithms have been presented that do not simply follow the concepts of one single classical metaheuristic, but attempt to obtain the best from a set of metaheuristics (and even other kinds of optimisation methods) that perform together and complement each other to produce a profitable synergy from their combination. These approaches are commonly referred to as *hybrid metaheuristics* (HMs) [Tal02, Rai06a, Blu10]. The increasing number of applications of hybrid metaheuristics and the existence of scientific events focused on this subject such as the series of *Workshops on Hybrid Metaheuristics* show the success and importance of this specific line of research.

SA [KGV83, AK89, HJJ03] was presented in 1983, becoming one of the first proposals within classical metaheuristics. However, SA is still the object of further studies [BSMD08], applied to many optimisation problems, or used as a component of other search algorithms [HJJ03, SSF02,

GML09b]. SA is commonly said to be the first algorithm extending local search methods with an explicit strategy to escape from local optima. Local search algorithms try to iteratively improve the current solution by performing movements to neighbouring solutions. The fundamental idea of SA is to allow moves resulting in solutions of worse quality than the current solution. The probability of making such a move is decreased during the search process.

EAs [BFM97, ES03] are currently one of the most popular metaheuristics, being the tool of choice for many researchers to face challenging problems. They are stochastic search methods that mimic the metaphor of natural biological evolution and rely on the concept of a population of individuals (representing search points in the space of potential solutions to a given problem), which undergo probabilistic operators such as mutation, selection, and (sometimes) recombination to evolve toward increasingly better fitness values of the individuals. There has been a variety of slightly different EAs such as *evolutionary programming* [Fog95], *evolutionary strategies* [BS02], and *genetic algorithms* [Gol89]. EAs offer practical advantages to researchers facing difficult optimisation problems because they may locate high performance regions of vast and complex search spaces. Other advantages include the simplicity of the approach, their flexibility, and their robust response to changing circumstances.

The hybridisation of EAs is becoming popular due to its ability to handle several real-world problems involving complexity, noise, imprecision, uncertainty, and vagueness [GA07a, PT99, MS10, VRH09]. In particular, it is remarkable the *use of SA to design HMs with EAs* (HMs-EA/SA) due to its prominent role in the field of hybrid EAs [TB05, LZXM10, WWR05, CWYH09, TW10].

Besides metaheuristics based on the search for new solutions close to the current one, like SA, or by combining solutions, like EAs, we find another group of metaheuristics that generate new solutions by adding one element at a time to a partial solution. This type of methods are known as constructive metaheuristics and we find several examples in the literature such as *iterated greedy* (IG) [JB95], *GRASP* [FR95], and *ant colony optimisation* [DS04]. Constructive methods are typically the fastest between approximate ones, however, they often return solutions of inferior quality with regards to local search algorithms [CHS02]. Thereby, to improve the quality of final solutions, most constructive metaheuristics include a local search method after the construction phase.

GRASP [FR89, FR95] is a multi-start two-phase metaheuristic for combinatorial optimisation basically consisting of a solution construction phase and an improvement phase. The solution construction mechanism builds an initial solution using a randomised heuristic procedure, whose randomness allows solutions to be obtained in different areas of the solution space. Each solution is randomly produced step-by-step by uniformly adding one new element from a restricted candidate list to the current solution. The improvement phase then takes the incumbent solution and performs local perturbations in order to get a locally optimal solution with respect to some predefined neighbourhood. The two-phase process of GRASP is iterative, that is, it continues until a user-defined termination condition is met. The best solution generated during this iterative process is kept as the overall result. Due to its simplicity, GRASP is often used for real-world applications [RR03].

IG [CL96, JB95, RS07] is a very effective metaheuristic recently developed for combinatorial optimization problems that follows a very simple principle, is easy to implement and can show excellent performance; in fact, it has exhibited state-of-the-art performances for a considerable number of problems [FPR10, FL08, LLYL11, URS10]. IG algorithms try to improve iteratively a solution by removing elements from this solution and completing the resulting partial solution using a constructive procedure. Moreover, IG algorithms may make use of an improvement phase that takes the incumbent solution after destruction and reconstruction and performs local perturbations

in order to find a better solution close to the incumbent one.

At the same time, constructive metaheuristics are being the subject of many studies on their hybridisations. In the literature, we find, among many other examples, hybridisations between ACO and GAs [CK12], ACO and fuzzy clustering [YLJ12], GRASP and path relinking [SCD⁺12], GRASP and particle swarm optimisation [MMD10], and IG and variable neighbourhood search [LB10].

For our study, we have considered problems in two distinct scenarios. In the first one, problems in which no useful knowledge that can be used during the solving process is available, except that provided by the problem representation and constraints. These problems are known as *black-box problems* [CLM05, GDLM08]. In the second one, we tackle problems in environments with specific knowledge, such as information about the structure of the objective function. In particular, we have considered two different problems: *parallel machines scheduling problem* [McN59] and *quadratic minimum spanning tree problem* [AX92].

Black-box problems appear in many scientific and engineering optimisation issues where there is not information to be exploited that might reinforce the search process. These problems are common when optimising computationally expensive dynamic models, bioprocess engineering, or where the evaluation of the solutions is carried out by simulations. To address these problems, context independent algorithms are applied due to they only require the type and domain of the decision variables to perform a search on the solution space. The original genetic algorithm proposed by Holland [Hol75] belongs to this class of methods.

Parallel machines scheduling problem consists in allocating a set of jobs in any of the available machines, so that the resulting allocation satisfies certain requirements. Within the general formulation, there are many variations in response to the different components of the problem, such as the machines features or the conditions of the scheduling mechanism. Among its practical applications are the optimisation of production processes in manufacturing industry and the optimisation of process allocation in computer systems with multiple processing nodes [Wot07].

The quadratic minimum spanning tree problem is an extension of the well-known minimum spanning tree problem [J.B56], where in addition to edge costs, we have costs associated with pairs of edges. This problem has been widely studied in the literature due to its applications in a wide variety of settings, including transportation, telecommunication, irrigation, and energy distribution. The problem appears, for example, when transferring oil from one pipe to another in a situation where the cost depends on the type of interface between two pipes. The same pairwise interaction effect arises in the connection of aboveground and underground cables in a road network with turn penalties [SS10, ZG98].

In this thesis memory, we present the research performed on the issues commented above: 1) optimisation methods for black-box environments based on HMs combining SA and EAs and 2) constructive metaheuristics for non-uniform parallel machines scheduling and quadratic minimum spanning tree problems. With regards to the first task, we have performed a study of the state-of-the-art HMs-EA/SA for combinatorial optimisation, classifying them according to a proposed taxonomy for this kind of methods. Moreover, we have presented new HMs-EA/SA that cover categories of the proposed taxonomy for which there are no previous HMs-EA/SA in the literature. Finally, we have compared the experimental performance of the most representative HMs-EA/SA and state-of-the-art methods for combinatorial optimisation. With regards to the second task, we have identified the state-of-the-art metaheuristics for the non-uniform parallel machines scheduling problem and the quadratic minimum spanning tree problem and we have explored constructive metaheuristics that result competitive with regards to state-of-the-art methods. To perform these tasks, we have structured this memory in two parts:

- Part I is dedicated to the problem statement, the discussion of proposed methods and the conclusions drawn.
- Part II contains publications associated with this study.

In Part I, after Introduction, we continue with the problem statement and the techniques used (Section 2), open problems that justify this thesis (Section 3), and the objectives proposed in it (Section 4). Later, in Section 5, we summarise the works performed throughout this thesis, highlighting most interesting results obtained together with the conclusions. Finally, we present the overall conclusions of this thesis (Section 6) and end with future research that remains open after the work performed in this thesis (Section 7).

In Part II, with the aim of developing the proposed objectives, we provide a compendium of five publications related to the accomplished study:

- A simulated annealing method based on a specialised evolutionary algorithm.
- Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: taxonomy, comparison, and synergy test.
- GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times.
- An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem.
- Tabu search with strategic oscillation for the quadratic minimum spanning tree.

2. Problem statement

In this section, we develop the problem statement and the techniques used to solve it with the following sections: Section 2.1 introduces the problem of optimisation; Sections 2.2, 2.3, and 2.4 describe black-box problems, scheduling on parallel machines problem, and quadratic minimum spanning tree problem, respectively. Then, we describe the basic concepts of metaheuristics in Section 2.5 and provide a more detailed description of those metaheuristics that play a major role in this thesis such as SA (Section 2.5.1), EAs (Section 2.5.2), GRASP (Section 2.5.3), and IG (Section 2.5.4); Finally, we introduce hybrid metaheuristics in Section 2.6.

2.1. Optimisation

Optimisation problems consist in finding the best configuration of a set of variables attending to certain criteria. More formally, an optimisation problem $P = (S, f)$ is defined by the following elements:

- A set of variables $X = \{x_1, \dots, x_n\}$.
- Variable domains D_1, \dots, D_n .
- Constraints among variables.
- An objective function f to be minimised (or maximised), where $f : D_1 \times \dots \times D_n \rightarrow \mathbb{R}$.

The search space is conformed by the set of all possible feasible assignments to the variables: $S = \{s = \{(x_1, v_1), \dots, (x_n, v_n)\} | v_i \in D_i, s \text{ satisfies all the constraints}\}$. The solution $s^* \in S$ with minimum (or maximum) objective value, that is, $f(s^*) \leq (\geq) f(s) \forall s \in S$, is known as a global optimum.

Attending to the variables domain of the problem, optimisation problems may be divided into two categories: those in which the solutions are encoded with real variables and those in which the solutions are encoded with discrete variables. Within the latter are combinatorial optimisation problems in which we are looking for an solution from a finite (or possibly countable infinite) set [PS82].

Combinatorial problems arise in many different fields such as economy, commerce, engineering, industry, or medicine. These problems are often very hard to solve in practise. In fact, the inherent difficulty of solving such problems is shown by the fact that many of them are known to be NP-hard [GJ90], which means there is no algorithm known for solving them in polynomial time, assuming that $P \neq NP$. Despite this fact, many of these problems have to be solved in a huge number of practical settings and therefore researchers have proposed a large number of methods to deal with them.

The proposed algorithms may be divided into two different categories: exact and approximate methods. Exact methods guarantee to find an optimal solution for every finite size instance in a bounded time [PS82, NW88]. These methods include techniques such as backtracking, branch and bound, dynamic programming, etc [BB96, PS82]. However, poor performance of exact methods for many problems has encouraged the design of several types of approximate algorithms. These

methods sacrifice the guarantee of finding an optimum solution for obtaining good solutions in a short amount of time.

2.2. Black-box problems

For many scientific and engineering optimisation problems, there is no useful knowledge that can be used during the solving process (*black-box problems*). These problems appear when optimising computationally expensive dynamic models, bioprocess engineering, or where the evaluation of the solutions is carried out by means of simulations. To address these problems, *context-independent algorithms* [CLM05, GDLM08] can be applied. These algorithms refer to methods that do not take advantage of problem structure to explore the solution space because they have no knowledge of specific characteristics of the objective function. They operate by treating the objective function evaluation as a black box. The solution representation is the only information that could be considered as part of the problem context.

Most metaheuristics are flexible and applicable to a wide range of optimisation problems, allowing the design of general purpose optimisers. In fact, the original model of genetic algorithms proposed by Holland [Hol75] belongs to this class of methods. Other recent proposals of this kind of methods include scatter search [GDLM10], a genetic algorithm specialised in intensification [GML09a], and SA [RH05]. Moreover, nowadays, general purpose optimisation is an issue in several commercial tools such as OptQuest (by OptTek Systems, Inc.) and Evolver (by Palisade Corp.). In particular, OptQuest includes several metaheuristics as optimisation tool, such as tabu search, genetic algorithms, and scatter search.

In order to perform the study on the behaviour of HMs-EA/SA as context independent optimisers, we have designed a test suite composed of 27 binary combinatorial optimisation problems, 13 of which were artificial problems and the remaining 14 were obtained from real-world applications. Table I.1 outlines their name, number of bits (D), a value (f^*) that stands for either the fitness value of the global optimum, known best solution, or upper bound presented in the literature, and reference. All of them have been formulated as maximisation problems. BQP and Maxcut instances can be obtained from the corresponding files from the *BiqMacLibrary*¹, and Multiple knapsack problems, from the *SAC – 94Suite*².

2.3. Parallel Machines Scheduling Problem

Scheduling problems deal, in general, with the allocating of resources over time to perform a set of tasks that are parts of certain processes, such as computational and manufacturing processes; see, for example, [BEP⁺07], for an extensive revision of scheduling problems. In particular, the *parallel machines scheduling problem* considers a set of n jobs that have to be processed on m parallel machines so that the resulting allocation is optimal, according to some criterion. Note that a job may be processed by only one machine at a time, and a machine can process at most one job at a time. If a job j is processed on a machine i , it will take a positive integral processing time p_{ij} . Furthermore, each job has a non-negative integer weight w_j . In the case of the identical parallel machines scheduling problem, each job has the same processing time regardless of the machine employed. By contrast, in the case of non-identical parallel machines, the processing time of a job

¹<http://biqmac.uni-klu.ac.at/>

²<http://elib.zib.de/pub/Packages/mp-testdata/ip/sac94-suite/>

Table I.1: Test suite for context independent optimisation

Prob.	Name	D	f^*	Ref.
1	Royal road problem (400, 8)	400	1	[FM93]
2	Trap problem	36	220	[Thi04]
3	Deceptive problem	39	390	[GKD89]
4	Bipolar deceptive problem	396	1	[PGCP00]
5	Overlapping deceptive problem	399	1	[PGCP00]
6	M-Sat(100,1200,3)	100	1	[SHS03]
7	M-Sat(100,2400,3)	100	1	[SHS03]
8	NkLand(48,4)	48	1	[Kau89]
9	NkLand(48,12)	48	1	[Kau89]
10	HIFF(2, 5, true)	32	192	[WP99]
11	HIFF(3, 4, false)	81	211	[WP99]
12	PPeaks(10,100)	100	1	[Kau89]
13	PPeaks(100,100)	100	1	[Spe00]
14	PPeaks(50,150)	150	1	[Spe00]
15	PPeaks(50,200)	200	1	[Spe00]
16	BQP(bqp50-1)	50	2098	[Bea98]
17	BQP(bqp500-1)	500	116586	[Bea98]
18	BQP(be120.3.3)	120	Not known	[Bea98]
19	BQP(be200.8.5)	200	Not known	[Bea98]
20	Maxcut(pm1s_80.6)	80	73	[Kar72]
21	Maxcut(w09_100.2)	100	2738	[Kar72]
22	Maxcut(g05_100.5)	100	1436	[Kar72]
23	Maxcut(pw05_100.6)	100	8217	[Kar72]
24	Maxcut(ising2.5-250_5555)	250	7919449	[Kar72]
25	Multiple knapsack p. (weish03)	30	4115	[Thi02]
26	Multiple knapsack p. (pet5)	28	12400	[Thi02]
27	Multiple knapsack p. (pb4)	29	95168	[Thi02]

depends on the selected machine, in two different ways:

- *Uniform parallel machines*: the processing time of job j on machine i is determined as $p_{ij} = p_j/s_i$, where p_j is the processing time of job j and s_i is the speed of machine i .
- *Unrelated parallel machines*: the values of p_{ij} are unrelated.

Since the introduction of this problem by McNaughton in [McN59], it has received much attention and many papers have been published in this area. For an in-depth review, interested readers may consult the survey by Cheng and Sin [CS90] and Mokotoff [Mok01], the chapter devoted to scheduling on parallel processors in [BEP⁺07] and a recent special issue on computational intelligence in scheduling [KTBS10]. Different real-world applications of scheduling on parallel machines can be found in the literature, covering a wide variety of fields such as human resources [RG87], production management [Bux89, DC91, PR00], mail facilities [JBdS92], robotised systems [Roc98], sport tournaments [CTA99], and chemical processes [BH00].

In the literature, we find different criteria to evaluate a particular scheduling. Below, we describe some of the most important ones:

- *Total weighted completion times*: the objective is to schedule the jobs in such a way that the sum of the weighted completion times of the jobs is minimised: minimise $\sum_{i=1}^n w_j * C_j$, where C_j represents the completion time of job j for a given schedule.
- *Maximum makespan*: the objective is to find a feasible schedule of minimum completion time: minimise $\text{argmax}\{C_j : j = 1, \dots, n\}$.
- *Maximum lateness*: taking into account that each job has a due date $d_j > 0$ by which time the job should be completed, the objective is to minimise maximum lateness: minimise $\text{argmax}\{C_j - d_j : j = 1, \dots, n\}$.
- *Total weighted tardiness*: the objective is to minimise the total weighted tardiness minimise $\sum_{i=1}^n w_j * T_j$, where $T_j = \text{max}(C_j - d_j, 0)$.

According to the notation proposed by Azizoglu et al. [AK99] and Allahverdi et al. [AGA99], each problem is denoted by a standard notation conformed by three fields: $\alpha|\beta|\gamma$. The first one is related to the machines type: identical (P_m), uniform (Q_m) and unrelated (R_m) parallel machines. The second field specifies conditions and information of the scheduling. Some conditions we can find in the literature are [LY09]: non-zero job ready time or dynamic release date (r_j), preemptive scheduling ($pmtn$), precedence constraints ($prec$), sequence-independent setup time (ST_{si}), and sequence-dependent setup time (ST_{sd}). Finally, the last one gives us information about the performance criteria: total weighted completion times ($\sum w_j * C_j$), makespan (C_{max}), maximum lateness (L_{max}) and total weighted tardiness ($\sum w_j * T_j$).

In this thesis, we focus on the problem of scheduling on non-identical parallel machines with minimising total weighted completion times ($Q_m, R_M || \sum w_j * C_j$). It is interesting to note that the majority of the studies have concentrated on the case of identical parallel machines [BM73, BP94, EP74, SAB88], despite the fact that non-identical parallel machines schedules have more practical relevance.

2.4. Quadratic Minimum Spanning Tree Problem

The quadratic minimum spanning tree problem (QMSTP) is an extension of the well-known minimum spanning tree problem, where in addition to edge costs, we have costs associated with pairs of edges. The problem was first introduced by Assad and Xu [AX92, W95], who showed that it is NP-hard. The QMSTP has been widely studied in the literature due to its applications in a wide variety of settings, including transportation, telecommunication, irrigation, and energy distribution. The problem appears, for example, when transferring oil from one pipe to another in a situation where the cost depends on the type of interface between two pipes. The same pairwise interaction effect arises in the connection of aboveground and underground cables in a road network with turn penalties [SS10, ZG98].

More formally, we may define the QMSTP as follows. Let $G = (V, E)$ be an undirected graph where $V = \{v_1, \dots, v_n\}$ is the vertex set and $E = \{e_1, \dots, e_m\}$ is the edge set. Consider that each edge and each pair of edges has an associated cost. In mathematical terms, we have two cost functions: $w : E \rightarrow \mathbb{R}^+$ and $c : (E \times E) - \{(e, e), \forall e \in E\} \rightarrow \mathbb{R}^+$ where as in previous approaches [RG08, PG10], we assume that $c(e_i, e_j) = c(e_j, e_i)$ for $i, j = 1, \dots, m$. The QMSTP consists of finding a spanning tree T of G with edge set $\xi(T) \subseteq E$ that minimises:

$$\sum_{e_i \in \xi(T)} \sum_{\substack{e_j \in \xi(T) \\ e_i \neq e_j}} c(e_i, e_j) + \sum_{e \in \xi(T)} w(e).$$

2.5. Metaheuristics

Metaheuristics [BR03, GK03, RH02, SM08, VOR99] have become one of the most popular tool to deal with combinatorial optimisation problems. In fact, metaheuristics are widely recognised as the most promising methods to address complex optimisation problems [AL97, MF04, Ree93]. Metaheuristics are methods that combine, at a high level of abstraction, approximate methods with the aim of exploring efficiently and effectively S .

In order to find high-quality solutions, metaheuristics have to be able to exploit intensively the regions of the search space with good solutions, while they move to unexplored regions when necessary. The concepts associated with these objectives are often called *intensification* and *diversification*, although the terms *exploitation* and *exploration* are sometimes used instead, mainly in the EA field. Therefore, of great importance hereby is that a dynamic balance is given between diversification and intensification.

Intensification and diversification can be understood as an effect of the components that conform a metaheuristic. In [BR03], they introduce a unified view of both concepts, in such way that each algorithmic or functional component of a metaheuristic that provides an intensification or diversification effect is defined as a *I&D component*. In contrast to the vision characterised by the unique identification as intensification or diversification components, I&D components have both effects. This unified view appears also in [GL97], where it is mentioned that intensification and diversification are not radically opposed. On the contrary, the best form of each one contains aspects of the other one. This way, there is a whole spectrum of alternatives.

This spectrum of alternatives is shown in [BR03] by means of a triangle where the corners correspond to extreme examples of I&D components (Figure 1). The corner OG corresponds to the I&D components that are guided solely by the objective function of problem considered. An example very close to this kind of component would be choosing the best neighbour in a local search

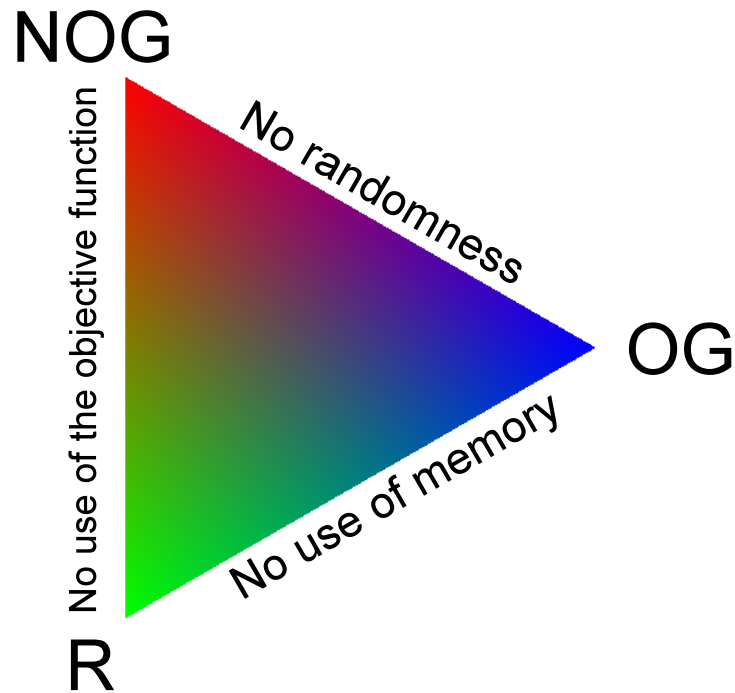


Figure 1: Space of I&D [BR03]

method. The corner *NOG* includes I&D components led by one or more functions other than the objective function, without the use of randomness. An example would be to create a solution, in a deterministic way, according to a report of frequencies. The third corner (*R*) comprises I&D components that are completely random. The creation of a solution totally random would be in this corner. According to the description of the corners, it is clear that *OG* corresponds to I&D components with a very high intensification effect and low diversification. Furthermore, the segment *NOG-R* corresponds to I&D components with a highly diversification effect and low intensification. During the past 30 years, different metaheuristics have emerged, looking to find a balance between intensification and diversification of various forms.

Metaheuristics can be classified according to different features. Depending on the selected feature, we obtain a classification or another, resulting from a specific viewpoint. Next, we highlight four features that give rise to four classification schemes of metaheuristics:

- *Origin inspired by nature.* An intuitive way of classifying metaheuristics is according to the origin of them. We find nature-inspired algorithms such as genetic algorithms, SA or ant colony optimisation [DMC96, DS04], among others, and non-inspired by nature methods as tabu search or iterated local search.
- *Number of solutions processed at the same time.* Another feature that can be used to classify metaheuristics is the number of solutions processed at the same time. The algorithms that work on a single solution are called *trajectory-based methods*. Some examples are local search methods, tabu search, iterative local search or variable neighbourhood search. They all describe a path in the search space during execution. The *population-based methods*, by contrast, perform a search process that describes the evolution of a set of points in the search space. Genetic algorithms and ant colony optimisation, among others, belong to this group.

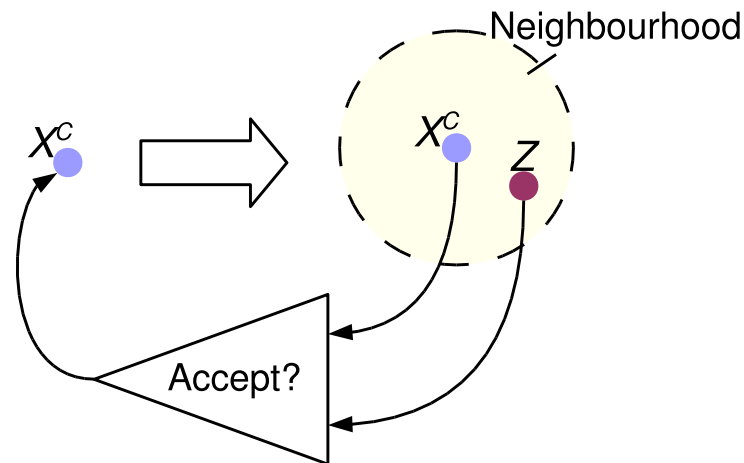


Figure 2: SA model

- *Memory usage.* A very important feature for classifying metaheuristics is the use (or not) of information obtained along the search performed, that is, if they employ memory or not. Memory-less methods decide the next action according solely to the current state of the search process. Many metaheuristics include, increasingly, memory usage. We can highlight tabu search as the first metaheuristic in which memory mechanisms acquire a major relevance.
- *Generation of new solutions:* According to the way new solutions are generated we can distinguish between three types of metaheuristics: by means of neighbourhood structures (tabu search or variable neighbourhood search), adding step by step components to partial solutions to complete new solutions (GRASP, IG, or ant colony optimisation), and combining different solutions (EAs).

2.5.1. Simulated Annealing

SA [AK89, KGV83] is an optimisation technique analogous to the physical process of annealing. SA starts with a high temperature T and any initial state (X^c). A *neighbourhood operator* is applied to the current state X^c (having energy $f(X^c)$) to yield state Z (having energy $f(Z)$) (see Fig. 2). Then, an acceptance mechanism decides which state becomes the new current state. Acceptance mechanism takes into account the current temperature of the system: worse solutions are more often accepted with high temperature. The application of the neighbourhood operator and the probabilistic acceptance of newly generated states are repeated either a fixed number of iterations or until a quasi-equilibrium is reached. The entire above-described procedure is performed repeatedly, each time starting from the current state and from a lower T .

The idea behind SA is *to protect diversification in the initial stages and intensification later*: at the initial stages, high T values favour the exploration of the search space, accepting new states almost regardless their energy; later, low T values increase exploitation, accepting only new better states.

There are two main *acceptance mechanisms* proposed in the literature: *metropolis* and *logistic* rules [AK02]. They define the probability of Z being the new current state. Equations (I.1) and (I.2) show *metropolis* and *logistic* acceptance mechanisms, respectively, for minimisation problems.

On the other hand, there exist several *cooling schemes*, such as *logarithmic*, *fast*, and *geometric* ones [AK02]. Equation (I.3) shows *geometric* cooling scheme (α is a cooling factor often assumed to be a constant in the interval $[0,9,1)$).

$$p(Z) = \begin{cases} 1 & \text{if } f(Z) < f(X^c) \\ e^{(f(X^c)-f(Z))/T} & \text{if } f(Z) \geq f(X^c) \end{cases} \quad (\text{I.1})$$

$$p(Z) = 1 - \frac{1}{1 + e^{(f(X^c)-f(Z))/T}} \quad (\text{I.2})$$

$$T \leftarrow \alpha \cdot T \quad (\text{I.3})$$

Convergence to global optimum is guaranteed for SA if a stationary distribution is reached at each temperature, followed by sufficiently slow cooling, such as logarithmic [VA87]. However, it usually leads to excessive long runs and practitioners use to apply faster cooling schemes at fixed number of iterations [SK91].

2.5.2. Evolutionary Algorithms

EAs are inspired by the ability of species to evolve and adapt to environment changes. *EAs* differ from the previous optimisation techniques in that they involve a search from a population of solutions, not from a single point. In each iteration, a set of operators is applied to individuals of the current population to generate the individuals of the next generation. These operators are commonly referred to as *crossover operator* or *recombination* when new individuals are created from the information of more than one individual of the current population, and *mutation operator*, when new individuals are generated from only an old one. The evolution of the population of individuals is guided by the fitness value of each individual, which commonly is the value of the objective function when evaluating this individual as a solution to the problem. Individuals with better fitness are most likely to produce offspring. This idea fits with the principle of natural evolution: *survival of the fittest*, allowing nature to adapt to changing environments.

Currently, there is a variety of methods belonging to the class of *EAs*. We highlight the following families: *evolutionary programming* [FOW66, Fog95], *evolution strategies* [BS02, Rec73, Rec94], *genetic algorithms* [Gol89, Hol75] and *differential evolution* [PSL05, SP97]. They all follow a process of evolution, which resembles the pseudocode shown in Figure 3, with slight differences between them.

Commonly, individuals in the population of an *EA* represent solutions to the problem, but also could represent partial solutions or solution sets, for example. Generally, the representations are closely linked to the problem being treated, being the most used bit strings, integers, real numbers or permutations of n integers. In the context of genetic algorithms, individuals hold the *genotype* whereas the associated solution is known as *phenotype*. Thus, there is a differentiation between the solution and its representation. The choice of a proper representation is crucial to the success of an *EA*.

In each iteration, the algorithm must decide which individuals should survive and become part of the population of the next generation. This is done through a selection scheme. In the *generational* scheme, individuals of the next generation are chosen exclusively from the population of descendants (P' and P''). In this case, we usually keep the best solution found so far, which is known as *elitism*. If

```

Input:  $f$ 
Output:  $X^b$ 
1  $P \leftarrow \text{GenerateInitialPopulation}();$ 
2  $\text{Evaluate}(P);$ 
3 while stopping condition is no met do
4    $P' \leftarrow \text{Recombination}(P);$ 
5    $P'' \leftarrow \text{Mutation}(P);$ 
6    $\text{Evaluate}(P'');$ 
7    $P \leftarrow \text{SelectIndividuals}(P'' \cup P' \cup P);$ 
8    $X^b \leftarrow \text{BestFoundIndividual}();$ 
9 end
10 return  $X^b;$ 

```

Figure 3: Basic scheme of an EA

it is possible that individuals of the current population are selected to be part of the new population, then the scheme is known as *steady-state*.

2.5.3. Greedy Randomised Adaptive Search

GRASP is a multi-start or iterative process in which each iteration consists of two steps: construction and local search. The construction phase builds a feasible solution that is refined during the local search phase, obtaining a local minimum. This process is repeated until a termination condition is met. The best solution found along the whole search process is returned as result. Figure 4 shows the pseudocode of a basic GRASP scheme.

```

Input:  $f, N$ 
Output:  $X^b$ 
1 while stopping condition is no met do
2    $X^c \leftarrow \text{GreedyRandomisedConstruction}();$ 
3    $X^c \leftarrow \text{LocalSearch}(N);$ 
4    $X^b \leftarrow \text{KeepBestsolution}(X^c, X^b);$ 
5 end
6 return  $X^b;$ 

```

Figure 4: Basic scheme of GRASP

The construction phase is accomplished by means of the function `GreedyRandomisedConstruction()`. This function performs an iterative process to construct a complete solution, adding at each step a new component to the partial solution until a complete solution is obtained. There is a set of candidate elements formed by all elements that can be incorporated to the partial solution and a greedy function to evaluate all the candidate elements. This greedy function usually represents the variation in the objective function due to the incorporation

of this element to the partial solution. At each step, the best α candidate elements are included in a *restricted candidate list* (RCL). The element to be incorporated into the partial solution is picked uniformly at random from RCL. Once the selected element has been included in the current partial solution, the candidate list is updated and the contribution value for each element in the candidate list is revised according to the greedy function. In Figure 5, the pseudocode of the GRASP construction phase is outlined.

<p>Input: Output: X^c</p> <pre> 1 $X^c \leftarrow \emptyset$; 2 while X^c is not completed do 3 $RCL \leftarrow \text{BuildRCL}()$; 4 $s \leftarrow \text{SelectRandomElement}(RCL)$; 5 $X^c \leftarrow \text{AddElement}(X^c, s)$; 6 $\text{UpdateContributionValues}()$; 7 end 8 return X^c; </pre>
--

Figure 5: Basic scheme of GRASP construction phase

The number of elements α included in the RCL determines the degree of randomness of the construction function. In the extreme cases, when $\alpha = 1$ and α is equal to the length of the candidate list, the construction phase is totally deterministic and totally random, respectively. Therefore, the election of a α value is critical to perform a suitable sampling of the search space. A review of the most important methods for determining α value is found in [PR02].

2.5.4. Iterated Greedy

IG algorithm generates a sequence of solutions by iterating over two main phases: destruction and construction. During the destruction phase some components of a previously constructed complete solution are removed. Then, construction procedure applies a greedy constructive heuristic to complete this partial solution. Once a candidate solution has been completed, an acceptance criterion decides whether the newly constructed solution will replace the current solution. This iterative process is performed until an stopping criteria is met. Moreover, IG algorithms may make use of a local improvement phase that takes the incumbent solution after destruction and construction and performs local perturbations in order to improve locally the quality of the incumbent solution. The pseudocode of the IG algorithm is showed in Figure 6.

During the destruction step, as mentioned above, a number of elements are dropped from the current complete solution. This parameter and the acceptance criterion are key in order to ensure IG proper operation. In the literature, we can find different acceptance criteria:

- 'Replace if better' acceptance criterion. The new solution (X^i) is accepted only if its objective function value is better than the one of X^c [YC10].
- An IG algorithm using the 'Replace if better' acceptance criterion may lead to stagnation of the search due to insufficient diversification [RS07]. In order to avoid this, different ac-

<p>Input: f</p> <p>Output: X^b</p> <pre> 1 $X^c \leftarrow \text{GenerateInitialSolution}();$ 2 while <i>stopping condition is not met</i> do 3 $X^p \leftarrow \text{Destruction}(X^c);$ 4 $X^i \leftarrow \text{Construction}(X^p);$ 5 $X^i \leftarrow \text{LocalSearch}(X^i)$ (optional); 6 $X^b \leftarrow \text{KeepBestSolution}(X^c, X^b);$ 7 $X^c \leftarrow \text{SelectNewCurrentSolution}(X^c, X^i);$ 8 end 9 return $X^b;$ </pre>

Figure 6: Basic scheme of IG

ceptance criteria consider replacing the current solution by the new solution although its objective function value is worse. In [RS07], a simulated annealing acceptance criterion is employed, whereas in [LMGM11], the improved solution X^i becomes the new current solution independently of its objective function value.

2.6. Hybrid Metaheuristics

Especially over the last years, it has appeared a large number of algorithms that do not purely follow the concepts of one single traditional metaheuristic, but they combine various algorithmic ideas from different metaheuristics, sometimes also from outside of the traditional metaheuristics field. These approaches are commonly referred to as hybrid metaheuristics. The main motivation behind such hybridisations of different algorithmic concepts is usually to obtain better performing systems that exploit and unite advantages of the individual algorithms, i.e. hybrids methods may be benefited from synergy [Rai06b, BPRR11]. The vastly increasing number of reported applications of hybrid metaheuristics and dedicated scientific events such as the series of Workshops on Hybrid Metaheuristics document the popularity, success, and importance of this specific line of research. In fact, currently it seems that choosing an adequate hybrid approach is determinant for achieving a high performance in solving difficult problems [Rai06b, GA07b, BPRR11].

The great significance and popularity has led to the appearance of multiple works that try to classify the different approaches of hybrid metaheuristics existing in the literature [Cot98, Tal02, EAK05, BRA05, CTA05, PR05]. Through these taxonomies, it is intended to facilitate the comparison of different hybrid methods and the evaluation of their performance [GA07b]. In [Rai06a], Raidl presents a new taxonomy, trying to merge the most important aspects of previous classifications and at some points extending them. This taxonomy contemplates different features of the hybrid approaches in order to classify them:

- *What we hybridise*, i.e. which kind of algorithms. We might combine (a) *different metaheuristics*, (b) metaheuristics with *specific algorithms for the tackled problem*, or (c) metaheuristics with other *techniques coming from fields like operations research and artificial intelligence*. Optimisation methods from other fields that have been successfully combined with metaheu-

ristics are exact approaches like branch and bound, dynamic programming, and integer linear programming and soft-computing techniques like neural networks and fuzzy logic [BPRR11].

- *Level (or strength) at which the different algorithms are combined.* *High-level combinations* retain the individual identities of the original algorithms and cooperate over a relatively well defined interface; there is no direct, strong relationship of the internal workings of the algorithms. On the contrary, algorithms in *low-level combinations* strongly depend on each other, which may mean that individual components or functions of the algorithms are exchanged.
- *Order of execution.* In the *batch or sequential model*, one algorithm is strictly performed after the other one. On the contrary, in the *interleaved and parallel models* the algorithms might interact in more sophisticated ways. A detailed classifications of hybrid parallel metaheuristics can be found in [CTA05, EAK05].
- *Control strategy.* In *integrative approaches*, one algorithm is considered a subordinate, embedded component of another algorithm. Whereas, in *collaborative combinations*, algorithms exchange information, but are not part of each other.

3. Justification

Nowadays, challenging hard optimisation problems arise from the technological advance of many human activities. Mainly, they are characterised by being dynamic problems, by the high number of implied decision variables and the complex relationships among them. Thus, new optimisation algorithm proposals are required to effectively affront these problems. Metaheuristics have been established as one of the most practical approaches to deal with combinatorial optimisation problems, being widely recognised as the most promising methods to address complex optimisation problems [AL97, MF04, Ree93]. In fact, research in metaheuristics has been very active during the last decades, facing successfully a wide spectrum of problems that are very hard to solve optimally.

Over the last few years, a large number of search algorithms have been presented that do not simply follow the concepts of one single metaheuristic, but attempt to obtain the best from a set of metaheuristics (and even other kinds of optimisation methods). The main motivation for the hybridisation of different optimisation algorithms is to exploit the complementary nature of different optimisation strategies, assuming that the hybrid approach will benefit from the synergy between them. In fact, the choice of an appropriate mix of complementary algorithmic concepts can be the key to obtain high performance methods to solve complex optimisation problems. In addition, working with hybrid metaheuristics is an opportunity for collaboration between research communities specialised in different metaheuristics and other optimisation methods. The existence of conferences and working groups focused on hybrid metaheuristics shows the growing popularity of this methodology. There is also specific books on this subject [BARS08, MSV10] and a considerable number of review articles [BPRR11, LGM10, Rai06b, RPB10, JBT09].

Among optimisation problems, there are problems in which there is no useful knowledge that can be used during the search process. These problems are known as black-box problems and appear frequently in scientific and engineering environments, in particular, it may come out when optimising computationally expensive dynamic models, in bioprocess engineering, or where the evaluation of the solutions is carried out by means of simulations. Context-independent algorithms may be a way to deal with these problems. These algorithms only need to know the domain and type of the variables and operate by considering the objective function as a black box. We may highlight that context-independent optimisers play an important role in several commercial software packages for optimisation such as OptQuest (by OptTek Systems, Inc.) and Evolver (by Palisade Corp.).

It is important to note that the first genetic algorithm was conceived as a context-independent algorithm. Since then, EAs have been applied successfully on a huge variety of problems. For this reason, we think that EAs may be a key ingredient to develop high-performance context-independent algorithms. At the same time, the hybridisation of EAs is becoming popular due to its ability to handle several real-world problems involving complexity, noise, imprecision, uncertainty, and vagueness [GA07a, PT99, MS10, VRH09]. A wide variety of metaheuristics such as tabu search [LGH10], greedy randomised adaptive search procedure [CF04], and iterated local search [LGM10], among others, have been employed to develop hybrid approaches with EAs. In particular, it is remarkable the prominent role of SA in the field of hybrid EAs [TB05, LZXM10, WWR05, CWYH09, TW10]. The current relevance of HMs-EA/SA can be shown through the visibility of this topic at the ISI Web of Science. Figure 7 shows an important number of publications and citations per year, as well as an increasing trend. We can conclude that, although the first items related to this topic appeared in 1992, nowadays HMs-EA/SA are subject of great interest and there is an important research community associated to their study. For the reasons

set out so far, we have considered the use of HMs-EA/SA to tackle black-box problems.

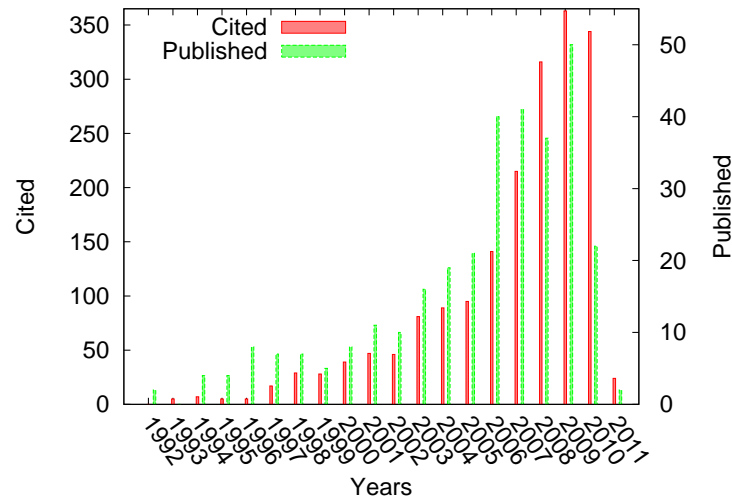


Figure 7: Number of publications and citations per year for HMs-EA/SA (Web of Science)

On the other hand, it is unquestionable that, in those cases where we deal with specific problems for which we have information that can be used during the search process, metaheuristics can improve their behaviour by using this information. In this thesis, moreover, we have chosen two specific problems for which we have considered using their specific information during the search process. These problems are scheduling on non-identical parallel machines with minimising total weighted completion times and quadratic minimum spanning tree. We have selected these problems due to their challenging nature, the interest of the scientific community on them, and their practical applications. In the case of scheduling on parallel machines, different real-world applications can be found on a wide variety of fields such as human resources, production management, mail facilities, robotised systems, sport tournaments, and chemical processes. For the quadratic minimum spanning tree, it has a wide variety of settings, including transportation, telecommunication, irrigation, and energy distribution.

To deal with these problems, we have considered two different metaheuristics: GRASP and IG. They belong to the constructive metaheuristics group and they consider using specific information of the problem during the construction step. Moreover, GRASP and IG usually include a local search method after the construction phase, during which information on the problem is also employed to guide the search. Currently, GRASP and IG has become a tool of choice to deal with challenging optimisation problems due to their efficiency and relative simplicity, exhibiting state-of-the-art performances for a considerable number of problems [FPR10, FL08, LLYL11, URS10, RR03].

We may highlight that hybridisations containing GRASP or IG are also subject of research [MMD10, LB10]. It is especially remarkable the interest on the hybridisations between GRASP and path-relinking [ABR03, ARPT05, LM99, RUW02]. This great interest and their promising results led us to consider also hybrids models between GRASP or IG and other metaheuristics to deal with scheduling on non-identical parallel machines and quadratic minimum spanning tree problems.

4. Objectives

The aim of this thesis is to perform an in-depth study of HMs-EA/SA, focusing on their application as context-independent optimisers for binary combinatorial problems, and the analysis of GRASP and IG as a tool to tackle two challenging combinatorial optimisation problems: scheduling on non-identical parallel machines and quadratic minimum spanning tree. To achieve this aim, we have set the following objectives:

- To paint a more complete picture of HMs-EA/SA than ever before. To do so, we structure and organise the knowledge about the HM-EA/SA approaches found in the literature by proposing a taxonomy for HMs-EA/SA based on those conceived by Talbi [Tal02] and Raidl [Rai06a] for hybrid metaheuristics.
- To develop new HMs-EA/SA that cover categories of the proposed taxonomy for which there are no previous HMs-EA/SA in the literature.
- To study the empirical behaviour of the different kinds of approaches according to the proposed taxonomy, analysing what schemes provide a better performance. At the same time, we aim to compare the behaviour of the best performing HM-EA/SA approaches against some state-of-the-art methods for binary combinatorial problems
- To study the synergistic relationships created by the hybridisation of EAs and SA in these approaches. The combination in a suitable manner of the complementary algorithmic concepts can provide hybrid approaches with a better performance than those obtained by EAs or SA separately.
- To perform an study on the state-of-the-art methods for scheduling on non-identical parallel machines and quadratic minimum spanning tree problems.
- To develop new methods based on constructive metaheuristics, specifically GRASP and IG, for scheduling on non-identical parallel machines and quadratic minimum spanning tree problems that are competitive with state-of-the-art metaheuristics for these problems.
- To compare the behaviour of GRASP and IG models depending on the configuration and components considered.

5. Joint Discussion of Results

This section shows a summary of the different proposals presented in this dissertation, and it presents a brief discussion about the obtained results by each one.

5.1. A Simulated Annealing Method Based on a Specialised Evolutionary Algorithm

As we mentioned in Section 3, the hybridisation of EAs is becoming very popular, facing hard problems in very different fields. In particular, it is remarkable the dominance of SA as component of the hybridisations of EAs. In this work, we present a new HM-EA/SA for binary combinatorial optimisation problems (SASEA). SASEA represents a novel approach to the development of HMs-EA/SA, proposing a *SA method in which the neighbourhood function is performed by means of an specialised EA*. The flexibility offered by the evolutionary paradigm allows specialised models to be obtained with the aim of performing as other search methods do, but more satisfactorily. This practise is a recent alternative to design new hybrid metaheuristics [GML08, GML09a, LHKM04, NI08, LGM10]. SASEA is conformed by a steady-state EA that creates one single candidate solution at each iteration, by crossing over the current solution and another one from the population. Afterwards, SASEA applies an acceptance mechanism, e.g. metropolis or logistic, to decide which solution becomes the new current solution, either the candidate solution or the current one. The other solution is inserted into the population by a replacement strategy. The general scheme of SASEA is shown in Figure 8.

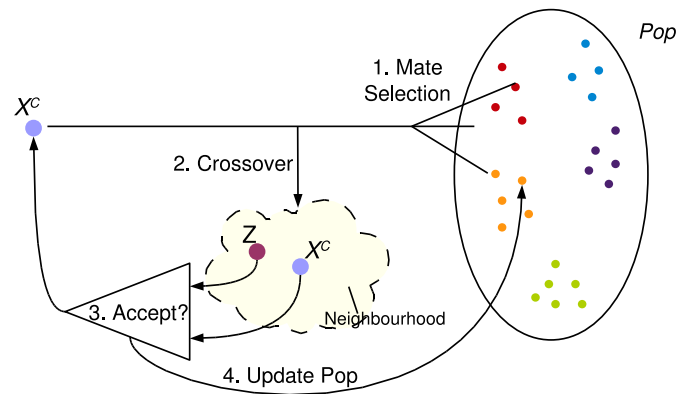


Figure 8: SASEA model

In addition to this new approach when designing hybrid models, SASEA presents a second important novelty, promoting a novel neighbourhood structure that presents two specific characteristics.

- It explores larger and dynamic neighbourhood sizes. Classic SA algorithms for binary optimisation usually consider one-flip operator, which flips just one bit of the current solution, to explore the neighbourhood of the current solution. However, SASEA may flip several bits of the current solution and the number of flips is adapted dynamically at each iteration.

- Neighbourhood exploration is guided by the mate to promote diversification and/or intensification. At initial stages of the search process, it intends to promote diversification by coupling the current solution with dissimilar individuals; whereas at latter stages, it attempts promoting intensification by pairing with similar individuals. This feature is further enhanced by the crossover mechanism employed by SASEA.

Both larger and dynamic neighbourhood sizes [Yao91, Liu99] and neighbourhood exploration guided by non-uniform probability distributions [Fox93, VT07] are reported in the literature as elements that may enhance efficiency of SA.

In this work, we present an extensive experimental study on the test suite outlined in Table I.1. This study is divided into three parts:

- In the first one, we compare the performance of SASEA with regards to classic SA and a SA model with dynamic neighbourhood (*dynamic SA*). SASEA obtains better results than the dynamic SA on almost all the problems considered. Moreover, classic SA and SASEA attain comparable results on many problems. However, SASEA shows a light superiority on several artificial problems (royal road, deceptive and bipolar deceptive, and three peaks instances) and two multiple knapsack problems.
- Secondly, we study the behaviour of other HMs-EA/SA existing in the literature focused on the evolution of a set of solutions in parallel. Thus, the notion of evolving populations from EAs is the centre of attention and SA ideas just define some actions on its members. Comparing the behaviour of SASEA and these approaches, we observe that most of the algorithms early get trapped in local optima or suffer premature convergence to poor solutions. Then, they have difficulties to improve further. However, the search process of SASEA is quietly driven toward promising solutions, overcoming local optima and reaching better final results than the other methods.
- Finally, we compare the performance of SASEA with regards to several state-of-the-art optimisers for binary combinatorial optimisation. These optimisers show a similar behaviour to HMs-EA/SA studied before. Most of the methods seem to get trapped in local optima early or experience slow convergence toward good solutions. However, though SASEA reaches solutions of equal quality later, its ability to protect diversification at the initial stages of the process and gradually increase intensification, lets SASEA to overcome local optima and reach better results.

The main conclusions of this work are the following:

- The neighbourhood structure of SASEA, which dynamically adapts its size and can be used to promote intensification and/or diversification, becomes beneficial when compared to the classic neighbourhood structure (one-flip) of SA and another one presented in the literature that dynamically adapts its size.
- The alternative scheme for combining ideas from SA and EAs introduced by SASEA may outperform other hybrid metaheuristics based on SA and EAs.
- SASEA provides a significantly better performance than other state-of-the-art algorithms for binary-coded problems, on the test suite considered.

The associated journal article to this part is:

C. García-Martínez, M. Lozano, F.J. Rodríguez, A simulated annealing method based on a specialised evolutionary algorithm. **Applied Soft Computing** **12:2** (2012) 473-488. doi: [10.1016/j.asoc.2011.11.007](https://doi.org/10.1016/j.asoc.2011.11.007).

5.2. Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test

The good results obtained by the proposed HM-EA/SA in the previous work and the difference in performance compared to other HMs-EA/SA lead us to make a thorough study of such methods. The aim of this study is threefold. Firstly, we attempt to paint a more complete picture of HMs-EA/SA than ever before. To do so, we structure and organise the knowledge about the HM-EA/SA approaches found in the literature by proposing a taxonomy for HMs-EA/SA which particularises the taxonomies conceived by Talbi [Tal02] and Raidl [Rai06a] for hybrid metaheuristics. Secondly, we study the empirical behaviour of the different kinds of approaches according to the proposed taxonomy, analysing what schemes provide a better performance. Finally, we study the synergistic relationships created by the hybridisation of EAs and SA in these approaches.

In Table I.2, we summarise the HMs-EA/SA found in the literature and the category they belong to according to the taxonomy proposed in this work. We may highlight that we have found categories for which, as far as we know, there are no HMs-EA/SA that can be classified into them. With the intention to fill this gap in the literature and thus perform a more comprehensive study, in this work, we have presented two HMs-EA/SA that belong to these unexplored categories: DCHCSA and GA-PSA.

In order to compare the behaviour of the different approaches according to the proposed taxonomy, we have chosen a representative set of the HMs-EA/SA found in the literature. This set has been built combining recent proposals and those that best fit the general scheme of each category. In Figure 9, we can see the average ranking obtained by each algorithm on the test suite presented in Table I.1 and the category to which it belongs. Ranking for each algorithm is computed according to Friedman's test, so that the lower its ranking is, the better its results on this test suite are. Figure 9 presents the ranking of each algorithm in a way that the height of each column is proportional to the ranking of its associated algorithm and the colour and shape depends on the category to which it belongs. In view of these results, in this work we have obtained the following conclusions about the different approaches to construct HMs-EA/SA:

- Those categories in which the hybridisation EA/SA is built employing incomplete formulations of either EA or SA, i.e. isolated components of SA or EAs are used to compose the resulting HM-EA/SA, present the poorest performance. On the contrary, those conceived as hybrid metaheuristics composed of self-contained EAs and SAs show better performances.
- The integrative approach is present in three out of the four best ranked HMs-EA/SA. In these hybrid approaches, one metaheuristic is specialised in to play a specific role inside the other metaheuristic. Specifically, in AGA, SA affords EA refined solutions and, in SALGeS and GAMSA, EA acts as an SA neighbourhood operator.
- The good results shown by GAMSA and SALGeS evidence the effectiveness of the novel hybridisation paradigm, which involves replacing some components in metaheuristics with

Table I.2: Taxonomy for HMs-EA/SA

General Categories		HM-EA/SA Categories	Instances
Collaborative	Teamwork	Multiple EAs and SAs	DCHCSA
		Multiple SAs	SSSA [TB05], CSA [XSVB06], ESA [AY05], GAMSA [RDGML10]
	Relay	EA then SA	HHSAGA [CLPM07], SAGA [BHS89]
		SA then EA	GA-PSA
Integrative	TeamWork	MA with SA as local search	AGA [LKH93], GASAHA [HDCL10], IGA-SA [LW07], GSAAL [ZWJZ08], GSAA [LZXM10]
		SA-based EA selection	HGA-BTS [Go190], GESA [YP95], HGA-BS[DT92]
		SA-based EA mutation and crossover	SAGACIA [LJ00], ARSAGA [HH06], GSAAIA [HQ06], HGA-SAM/R [Ad193]
		SA-based EA replacement	PRSA [MG95], PGSA [WWR05], GSA [CFW98], NPOSA [COC98], MPGSAA [CWYH09], GSA-MLE [YTY07]
	Relay	EA-based SA component	SALGeS [GML09b] GAMSA [RDGML10]

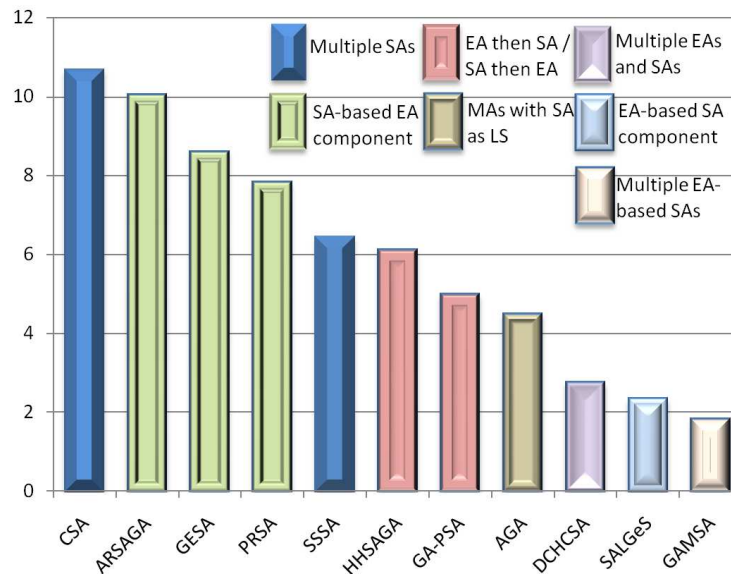


Figure 9: Average rankings of the HMs-EA/SA versions

customised EAs (*evolutionary components* [LGM10]) to develop the same work more effectively and with a relatively low computational cost, employed in these approaches.

- The teamwork collaborative hybridisation technique, instantiated by DCHCSA, presents better results than the relay collaborative approaches, represented by HHSAGA and GA-PSA.

Finally, to complete the in-depth study on HMs-EA/SA, we have studied the synergy produced by the combination of the composing metaheuristics. Synergy is one of the most important aspects when analysing the behaviour of a hybrid metaheuristic. In fact, exploiting the complementary character of the different optimisation strategies involved in hybrid metaheuristics is the main motivation behind the hybridisation, that is, hybrids are believed to benefit from synergy [BPRR11]. In order to assess the amount of synergy that appears when combining two or more components, or whether it does or does not appear, the usual practise involves the comparison between the hybrid algorithm and the sole usage of its components [Ant09, PTCY10, HLS05]. In this work, we have studied the synergy produced by combining SA and EAs in hybrid metaheuristics with regards to the sole usage of its components.

Figure 10 summarises the results of the synergy study accomplished in this work. In this figure, we show for each HM-EA/SA whether its results are statistically better than (+), worse than (−) or equal to those of a standalone SA, canonical generational and steady-state genetic algorithms (CGGA and CSSGA), and CHC [ES91]. In addition, last column indicates whether the corresponding algorithm presents a complete positive result in terms of synergy (green circle), only partially (orange circle) or there is no any synergy (red circle) according to the previous results. In light of the results outlined in Figure 10, we conclude that the simple combination of several metaheuristics does not ensure success. It is necessary to study the way the composing metaheuristics are combined in order to achieve a positive synergy between them. In fact, only three out of eleven HMs-EA/SA present a complete or partial positive result in terms of synergy.

The associated journal article to this part is:

F.J. Rodríguez, C. García-Martínez, M. Lozano, Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test. **IEEE Transac-**












HMs	SA	CGGA	CSGA	CHC	Synergy
GAMSA	+	+	+	+	
SALGeS	+	+	+	+	
DCHCSA	~	+	+	+	
AGA	-	~	+	~	
GA-PSA	-	-	~	-	
SSSA	-	-	+	-	
HHSAGA	-	-	+	-	
PRSA	-	-	-	-	
GESA	-	-	-	-	
ARSAGA	-	-	-	-	
CSA	-	-	-	-	

Figure 10: Synergy study: HMs-EA/SA vs. Standalone SA, Canonical GAs, and CHC

tions on Evolutionary computation. In Press. doi: 10.1109/TEVC.2012.2182773.

5.3. GRASP with Path-Relinking for the Non-Identical Parallel Machine Scheduling Problem with Minimising Total Weighted Completion Times

In this work, we tackle the problem of scheduling a set of jobs on a set of non-identical parallel machines with the goal of minimising the total weighted completion times (Section 2.3). The research efforts to deal with this problem have focused on three main research lines: exact procedures, approximation algorithms through solving relaxations of the problem, and metaheuristic procedures. Exact algorithms are generally limited to problem instances of moderate size, due to the exponential increase in CPU time and memory when the problem size increases. Therefore, in practice, approximate algorithms are necessary to find (not necessarily optimal) solutions to the considered problem. In particular, there are in the literature different approaches based on metaheuristics such as genetic algorithms [ZJLK10, LPF11], tabu search and multi-start local search [VH02], and differential evolution and variable neighbourhood descent [ZJLK10].

To deal with this problem, in this work we have considered the application of a GRASP metaheuristic (Section 2.5.3), which follows a constructive methodology to explore the search space unlike metaheuristics presented so far in the literature to face this problem. GRASP has been successfully applied to deal with several real-world applications [RR03] and it is very often combined with *path-relinking* [Glo96]. Path-relinking consists of exploring trajectories that connect high-quality solutions and, generally, may lead to significant improvements in solution quality when combined with GRASP [ABR03, ARPT05, LM99, RUW02].

The proposed hybrid between GRASP and path-relinking extends the basic GRASP scheme by incorporating path-relinking into two different steps of our proposal. Firstly, path-relinking is incorporated as an improvement procedure of solutions obtained after each iteration of basic GRASP. For this, a set of so-called elite solutions (P) is used to store high-quality solutions found during the search process. Path-relinking is then generally applied to the solution generated in the current iteration and one solution from P . In particular, we have employed *backward path-relinking* that considers as initial solution the best from the current solution and a solution from P . Secondly, we incorporate evolutionary path-relinking as a post-processing phase for our proposal, applying path-relinking to pairs of solutions chosen from the set P . Each resulting offspring solution is tested for membership in the population of the next generation.

Among the contributions that are made in this paper, we highlight the following:

- We have presented an hybrid method between GRASP and path-relinking to deal with this problem, presenting a novel approach through a constructive metaheuristic unlike other metaheuristics proposed in the literature so far. Path-relinking has been included in the proposed method as an improvement procedure for both solutions provided by the basic GRASP and solutions of the elite set.
- We have proposed a greedy randomised procedure, which is responsible for providing initial solution to our method, based on a previous greedy algorithm [WLR01].
- The improvement phase of the proposed hybrid algorithm between GRASP and path-relinking combines two different local search methods. On the one hand, we have considered a variable neighbourhood descent proposed before [ZJLK10]. On the other hand, we have employed a simple local search based on insertion moves. Both methods are executed alternatively, and, since variable neighbourhood descent is more time-consuming than the simple local search method, the variable neighbourhood descent procedure is only used every a certain number of iterations.

We have performed an experimental study on a set of instances with up to 200 jobs and 20 machines, considering both uniform and unrelated cases. We should point out that the majority of the studies performed so far have concentrated on the case of identical parallel machines. This study has three objectives:

- To find a suitable configuration for the different parameters of the proposed hybrid approach between GRASP and path-relinking.
- To compare the performance of previous metaheuristics for this problem and our proposal.
- To study the synergy produced by the combination of pure GRASP and path-relinking in the proposed approach.

The main conclusions of this work are the following:

- The proposed hybrid algorithm between GRASP and path-relinking outperforms the competing algorithms from the literature on both uniform and unrelated parallel machines.
- The combination of basic GRASP with path-relinking helps us to achieve solutions of higher quality than those obtained by the pure GRASP.

- Instances with a large number of jobs and machines represents a great challenge for methods dealing with this problem.

The associated journal article to this part is:

F.J. Rodríguez, C. Blum, C. García-Martínez, M. Lozano, GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times. **Annals of Operations Research. In Press.** doi: [10.1007/s10479-012-1164-8](https://doi.org/10.1007/s10479-012-1164-8).

5.4. An Iterated Greedy Algorithm for the Large-Scale Unrelated Parallel Machines Scheduling Problem

In this work, we pose deal with problems in which there is a large number of jobs and machines. As we saw above, dealing with large-scale instances presents major challenges for metaheuristics. In fact, nowadays, large scale optimisation problems have been one of the most interesting trends in the last years for what concerns research on evolutionary algorithms and metaheuristics. This is due to the fact that many real-world problems are of large size. Unfortunately, the performance of many available optimisation algorithms deteriorates rapidly as the dimensionality of the search space increases. Thus, scalability for high-dimensional problems becomes an essential requirement for modern optimisation algorithms. In particular, we can find metaheuristics to face parallel machine scheduling problems that incorporate techniques for dealing with large size instances [FPR11]. However, for the present problem, previous studies have focused on studying the performance of the proposed metaheuristics on small/medium-size instances.

IG is a very simple and effective constructive metaheuristic recently developed for combinatorial optimisation problems that follows a very simple principle, is easy to implement and can show excellent performance; in fact, it has exhibited state-of-the-art performances for a considerable number of problems [FPR10, FL08, LLYL11, URS10]. In addition, IG has shown a good performance dealing with large instances on other problems [FPR11, LMGM11]. One of the most important differences between IG and GRASP is that the former is able to use information from previous iterations to more effectively explore the search space in subsequent iterations, which can be beneficial when dealing with large-size instances. However, we should point out that hybrid models between GRASP and path-relinking try to overcome this drawback through using the elite solutions set that characterises path-relinking.

In this work, we have developed a model than combine IG with a local search method, allowing reaching high-quality solutions especially for large-size instances, in scenarios with up to 1000 jobs and 50 machines, which represents an extremely substantial difference with respect to previous works. In particular, we have focused on instances contemplating unrelated parallel machines, which represent the more general scenario. The main novelties of this work are the following:

- In the first place, we have considered a new acceptance criterion to choose the new current solution. A uniform random number p_a between 0 and 1 is generated. If $p_a \leq 0,5$, candidate solution becomes the new current solution, independently of its objective function value. Otherwise, candidate solution is discarded. By using this acceptance criterion, we try to avoid stagnation of the search due to insufficient diversification [RS07].
- Secondly, we propose two destructive strategies for the destruction phase. The most simple and extended strategy to perform the IG destructive procedure consists in randomly choosing the elements to be deleted (DR). However, as it was showed by Fanjul et al. [FPR10] for the case

of scheduling on parallel machines with minimising makespan, more elaborated destruction procedures can lead to better solutions. Based on this idea, we have studied two different destruction strategies:

- In the first strategy (D1), one machine is uniformly selected at random and a binary tournament is performed in order to select the job to be deleted from this machine. Two jobs are randomly chosen and that with a lower p_{ij}/w_j ratio will be dropped. This destructive strategy tries to move the jobs scheduled at the beginning in each machine, which are delaying the jobs scheduled after them, to a more suitable machine.
- The second strategy (D2) also randomly selects a machine. However, it defines a different strategy to choose a job. In particular, this strategy selects the jobs whose processing times are shorter in other machines, and, consequently, could be completed earlier in there ([FPR10]). Once a machine i is selected, the job k that has the largest processing time difference with respect to the other machines is deleted:

$$k = \operatorname{argmax}\{p_{ij} - p_{lj} : l = 1, \dots, m; j = 1, \dots, n; l \neq i\}.$$

We may highlight that we have used instances with unrelated parallel machines considering 20 different combinations of the number of jobs and the number of machines. Moreover, seven sets of ten problem instances were randomly generated for each instance type following different ways of choosing the processing time of job j on machine i (p_{ij} , $i = 1, \dots, n$ and $j = 1, \dots, m$) and its weight (w_j). All the instances used can be classified into two broad categories: correlated and uncorrelated instances [FPR10].

The experimental study performed on this set of instances have two main goals: 1) to find suitable values for the parameters of our model and understand their influence on the behaviour of the algorithm and 2) to perform a comparison between the proposed method combining IG with a local search method and state-of-the-art metaheuristics from the literature. From the results of this study, we can obtain the following conclusions:

- The use of the proposed acceptance criterion leads to the best results for our approach compared with other acceptance criteria.
- Using more elaborate destructive strategies such as D1 and D2 clearly improves the performance of the proposed IG for what concerns correlated instances. However, when considering the uncorrelated instances, there are no significant performance differences between D1 and DR. Moreover, for uncorrelated instances, D2 provides a poorer performance than DR. Taking these results into account, we consider that our model with D1 provides the best performance over all the remaining alternatives.
- The proposed IG obtains statistically better results than its competitors for the set of 140 different instance types contemplated in this work. In summary, this experimental analysis confirms that the IG model presented in this work is a very attractive alternative to the existing approaches for this problem, especially for large-size problem instances.

The associated journal article to this part is:

F.J. Rodríguez, M. Lozano, C. Blum, C. García-Martínez, An Iterated Greedy Algorithm for the Large-Scale Unrelated Parallel Machines Scheduling Problem. **Computers & Operations Research**. Submitted on second revision.

5.5. Tabu Search with Strategic Oscillation for the Quadratic Minimum Spanning Tree

In this work, we deal with the quadratic minimum spanning tree problem (QMSTP) (Section 2.4). This problem appears in real-world situations such as transferring oil from one pipe to another in a situation where the cost depends on the type of interface between two pipes and the connection of aboveground and underground cables in a road network with turn penalties [SS10, ZG98]. Existing methods for solving this problem are divided into exact procedures (limited to small problems containing from 6 to 15 vertices) and metaheuristics such as genetic algorithms, artificial bee colony, and tabu search.

We have explored two different metaheuristics to deal with the QMSTP: an hybrid metaheuristic combining tabu search [GL97] and strategic oscillation [Glo77, GL97] (OS+TLS) and an IG (Section 2.5.4) constructive metaheuristic.

Tabu search is a trajectory-based metaheuristic that employs explicitly a search history with two objectives: to escape from local optima and to implement a strategy to enhance diversification. The basic tabu search selects the best neighbour solution, even if it is worse than the current solution. To avoid cycles, it employs a *short-term memory* that intends to avoid moves towards solutions already visited. Moreover, often, tabu search uses a *long-term memory* that gathers information about whole search process to use it to guide the search. Strategic oscillation is closely linked to the origins of tabu search, and operates by orienting moves in relation to a critical level, as identified by a stage of construction.

In our SO+TLS approach for the QMSTP, we consider a constructive/destructive type of strategic oscillation, where constructive steps add elements and destructive steps drop elements (at this point, we may highlight the resemblances between IG and strategic oscillation). SO+TLS for QMSTP applies, in the first place, strategic oscillation and then tabu search is employed as a local improvement procedure to solutions obtained by strategic oscillation. We may highlight that both IG and strategic oscillation share the constructive procedures designed in this work for QMSTP. However, strategic oscillation adds memory structures to construction and destruction phases, following a strategy that is in line with tabu search principles.

The main novelties of this work are the following:

- We have presented a constructive metaheuristic based on the IG scheme and an hybrid approach between tabu search and strategic oscillation for the QMSTP.
- As we mentioned above, in SO+TLS, tabu search is employed as local improvement procedure to solutions obtained by strategic oscillation. For this, we have proposed a *short term memory tabu local search* (TLS) based on a existing local search method [SS10] to which we add a memory structure and apply candidate list strategies for move selection.
- We have developed two new constructive strategies (C1 and C2) for IG and strategic oscillation based on Kruskal's algorithm [J.B56] and the sequential fixing method proposed by Assad and Xu [AX92], respectively. In addition, we have proposed a destructive strategy for IG based on the reverse version of Kruskal's algorithm.
- We show commonalties shared by strategic oscillation and IG. At the same time, we identify implications of their differences regarding the use of memory structures.

The experimental study performed in this work has the following main objectives:

- To analyse the influence of the parameters and settings associated with IG.
- To show the benefits of the use of memory structures in strategic oscillation and TLS.
- To compare the behaviour of IG, OS+TLS, and a state-of-the-art memory-based approach for the QMSTP (ITS [PG10]).

The main conclusions of this experimental study are the following:

- The acceptance criterion has the largest impact on the IG performance. The best results are achieved by the IG version considering the 'random walk' acceptance criterion [LMGM11]. Moreover, IG version using C1 shows a clear advantage over those with C2. It is interesting to remark that the simplest constructive algorithm, C1, becomes the most effective one.
- The effects derived from the combination of the exploration power of TLS and a moderate diversification by strategic oscillation, both effects derived from the use of memory structures, are key for OS+TLS to attain a high level of robustness on the set of instances considered.
- OS+TLS is superior for the most complex instance set, IG for the large instances, and ITS becomes the winner for the case of the easiest instances.

Thus, regarding these results and with the aim of producing a robust operation, we have built a hybrid algorithm, dubbed HSII, that combines OS+TLS, IG, and ITS. Specifically, HSII is a relay collaborative hybrid metaheuristic [Tal02] that executes OS+TLS, IG, and ITS in a pipeline fashion. The idea of this hybridisation scheme is: 1) to use OS+TLS as diversification agent to reach good initial solutions for the most difficult problem instances, then 2) to allow IG enough time to offer a suitable behaviour on large instances, and finally 3) to employ ITS as an effective improvement procedure for refining the best solution found in the previous stages.

Finally, we expand the experimental study by comparing HSII behaviour with regards to state-of-the-art methods for QMSTP and HSII components. The results of this study show that HSII outperforms its competitors on most of the problem instances, achieving an acceptable level of robustness across a wide range of different QMSTP instances.

The associated journal article to this part is:

M. Lozano, F. Glover, C. García-Martínez, F.J. Rodríguez, R. Martí, Tabu Search with Strategic Oscillation for the Quadratic Minimum Spanning Tree. **IEE Transactions**. **Submitted on second revision**.

6. Conclusiones

Esta sección resume brevemente los resultados obtenidos y presenta diversas conclusiones

1. *Estudio de HMs-EA/SA para problemas de caja negra en optimización binaria combinatoria.*

En esta tesis, hemos proporcionado una visión general de las distintas formas en las que los EAs y SA pueden ser combinados para obtener HMs-EA/SA. Hemos organizado las distintas aproximaciones encontradas en la literatura mediante una taxonomía que extiende las propuestas por Talbi y Raidl para metaheurísticas híbridas [Tal02, Rai06a] y hemos propuesto nuevos modelos de HMs-EA/SA que siguen enfoques no explorados hasta ahora en la literatura. Además, hemos desarrollado, hasta donde sabemos, el primer estudio experimental en el que se compara una gran variedad de modelos de HMs-EA/SA desde tres puntos de vista:

- Hemos comparado el rendimiento de los modelos de HMs-EA/SA de forma individual y por categorías, extrayendo conclusiones relevantes acerca de las categorías de la taxonomía presentada.
- Hemos realizado un test de sinergia que ha identificado dos modelos de HMs-EA/SA que realmente presentan características sinérgicas.
- Hemos comparado diversos modelos de HMs-EA/SA frente a algoritmos evolutivos estado del arte, obteniendo prometedores resultados en el marco de pruebas considerado.

Nuestro estudio nos permitió obtener una conclusión destacada: la hibridación de los EAs y SA es un área de investigación futura para encontrar algoritmos de búsqueda más eficientes.

2. *Metaheurísticas híbridas y constructivas para el problema de la planificación en máquinas paralelas con minimización del tiempo total de finalización ponderado.*

Para el problema de la planificación en máquinas paralelas con minimización del tiempo total de finalización ponderado, hemos presentado dos aproximaciones distintas en esta tesis:

- En primer lugar, hemos propuesto un algoritmo híbrido para la planificación en máquinas uniformes y no relacionadas que combina el esquema básico de GRASP con otros elementos como encadenamiento de trayectorias y encadenamiento de trayectorias evolutivo. Además, hemos realizado una comparativa entre el algoritmo propuesto y diversas metaheurísticas existentes para este problema. Este estudio muestra que GRASP + encadenamiento de trayectorias proporciona realmente el mejor rendimiento de entre los algoritmos comparados. Además, elementos como la construcción voraz aleatorizada de las soluciones iniciales y el encadenamiento de trayectorias, que distinguen el método propuesto de las metaheurísticas anteriores, han mostrado su utilidad para mejorar la calidad de las soluciones y obtenerlas en un tiempo razonable. Finalmente, el mejor rendimiento de la hibridación entre GRASP y el encadenamiento de trayectorias con respecto al modelo básico de GRASP muestra que realmente se consigue una sinergia beneficiosa de esta hibridación.
- En segundo lugar, hemos propuesto un algoritmo IG para afrontar el problema de la planificación en máquinas paralelas no relacionadas, extendiendo este estudio a instancias más grandes que en el trabajo anterior y considerando diferentes condiciones para la generación de las instancias. El estudio experimental realizado muestra que: 1) el algoritmo IG propuesto es muy competitivo con respecto a los algoritmos estado del arte para este

problema; específicamente, podemos destacar las sustanciales mejores obtenidas cuando afrontamos instancias de gran tamaño (consideran gran número de trabajos o máquinas) y 2) nuestra propuesta de IG ha sido ejecutada sobre un gran número de instancias diferentes, resultando ser muy robusta. Además, desde el punto de vista metodológico, el algoritmo IG propuesto presenta dos novedades que pueden ayudar a mejorar el rendimiento de IG en otros problemas de optimización. En primer lugar, el criterio de aceptación con aleatoriedad ha resultado ser muy útil para mejorar el rendimiento de IG. En segundo lugar, la inclusión de una estrategia heurística en la fase de destrucción ha permitido mejorar los resultados del algoritmo propuesto en ciertos tipos de instancia difíciles. De esta forma, pensamos que el uso de estrategias más elaboradas que la simplemente aleatoria en la etapa de destrucción deben de ser consideradas cuando se afrontan problemas de optimización complejos mediante IG.

3. *Metaheurísticas híbridas y constructivas para el problema del árbol de expansión cuadrático mínimo.*

Finalmente, en esta tesis, hemos tratado el problema del árbol de expansión cuadrático mínimo y hemos comparado el comportamiento en el mismo de dos metaheurísticas que siguen una estrategia constructiva como IG y la oscilación estratégica. Nuestra investigación demuestra la efectividad de una estrategia que alterna etapas constructivas y destructivas, como fue propuesto originalmente en la oscilación estratégica y más recientemente en IG.

Nuestras experimentos revelan que el método de oscilación estratégica basado en el uso de memoria es capaz de resolver mejor que los algoritmos existentes anteriormente instancias complejas. Por otro lado, nuestra implementación de IG resulta más exitosa cuando se afrontan instancias grandes pero no excesivamente complejas, mientras que la búsqueda tabú iterativa (ITS) se comporta mejor en las instancias más fáciles. Basándonos en estos descubrimientos, desarrollamos un método híbrido, HSII, que combina estos tres algoritmos, lo que resultó ser muy efectivo para todas las instancias, con la excepción de una clase de instancias donde ITS continua siendo el ganador.

Para finalizar, la habilidad de las estrategias constructivas/destructivas para dar mejores resultados en instancias grandes, junto con el valor del uso de memoria en instancias complejas y de su ausencia en las más sencillas, invita a considerar otras formas de oscilación estratégica, particularmente, considerando el transpaso de los límites de factibilidad (cuyo valor se subrayó en [GH11]) y yendo más allá de la simple aleatorización en los movimientos destructivos introduciendo estrategias más avanzadas para seleccionar estos movimientos (como se indicó en [GL97] y en nuestro trabajo previo sobre IG para el problema de la planificación en máquinas paralelas).

6. Conclusions

This section briefly summarise the obtained results and present several conclusions.

1. *Study of HMs-EA/SA for black-box problems in binary combinatorial optimisation.*

In this thesis, we provided an overview of the ways EAs and SA may be combined with each other to obtain HMs-EA/SA. We have organised the approaches found in the literature by proposing a taxonomy based on those introduced by Talbi and Raidl for hybrid metaheuristics [Tal02, Rai06a] and we have proposed new HM-EA/SA models that follow unexplored paths so far in the literature. Moreover, we have developed, to our knowledge, the first experimental study analysing a large spectrum of HM-EA/SA models from three points of view:

- We have compared the performance of the HM-EA/SA models, individually and by categories, extracting relevant conclusions regarding the categories of the presented taxonomy.
- We have performed a synergy test that has identified two HM-EA/SA models that really provide synergistic properties.
- We have compared HM-EA/SA models with the state-of-the-art evolutionary algorithms for binary combinatorial optimisation, obtaining promising results on the considered testbed.

Our study allowed us to draw an outstanding conclusion: the hybridisation of EAs and SA becomes a prospective research area for finding more effective search algorithms.

2. *Hybrid and constructive metaheuristics for scheduling on non-identical parallel machines with minimising total weighted completion times.*

For the non-identical parallel machines scheduling problem with minimising total weighted completion times, we have proposed two different approaches in this thesis:

- In the first place, we have proposed a hybrid algorithm for the scheduling on uniform and unrelated parallel machines problem, which combines the basic GRASP scheme with other elements such as path-relinking and evolutionary path-relinking. Moreover, we have performed a comparative study between the proposed algorithm and the previously existing metaheuristics for this problem. This study has shown that the GRASP + path-relinking algorithm provides the best performance from among the studied algorithms. In addition, elements such as the greedy randomised initial solutions and path-relinking, which distinguish the proposed method from compared metaheuristics, have proven to be quite useful increasing the quality of the solutions and achieving high-quality solutions in a reasonable time. Finally, the better performance of the hybrid GRASP + path-relinking over the pure GRASP shows that their combination really provide a profitable synergy.
- Secondly, we have proposed an IG algorithm to deal with the scheduling on unrelated parallel machines problem, expanding the study to instances larger than the previous work and considering different conditions for generating instances. The computational experiments performed show that: 1) the proposed IG is very competitive with other

state-of-the-art algorithms for this problem; specifically, significant improvements were obtained for large size problem instances (involving a large number of jobs or machines) and 2) Our IG has been tested on a great number of different kinds of instances, proving to be very robust. Moreover, from the methodological point of view, the proposed IG presents two novel elements that can help to improve the performance of IG on other optimisation problems. In the first place, the acceptance criterion including randomness has proved to be quite useful to improve its performance. Secondly, the heuristic strategy for the destruction step has allowed improving the results of the proposed IG on certain types of difficult instances. This way, we think that using more elaborated strategies in the destructive step than the classical random one should be considered when dealing with complex optimisation problems thorough IG algorithms.

3. *Hybrid and constructive metaheuristics for the quadratic minimum spanning tree problem.*

Finally, in this thesis, we have faced the quadratic minimum spanning tree problem and compared the behaviour on this problem of two different metaheuristics that follow a constructive strategy as SO and IG. Our research study demonstrates the effectiveness of a strategy for solving this problem that alternates between constructive and destructive phases, as originally proposed in strategic oscillation and more recently in IG method.

Our tests disclose that the memory-based strategic oscillation method is able to solve complex instances better than other previous algorithms. On the other hand, our implementation of the IG algorithm succeeds in performing most effectively for large problems that are not highly complex, while the iterated tabu search (ITS) algorithm performs best in application to easier instances. Based on these findings we developed a hybrid method, HSII, that combines these three algorithms, which proved very effective across the board with the exception of one class of problem instances, where ITS remained the winner.

Overall, the ability of the alternating constructive/destructive strategies to give superior outcomes for larger problems, with memory proving valuable for complex problems and a disregard for memory proving valuable for easier problems, invites further consideration of other forms of strategic oscillation, particularly by crossing feasibility boundaries (whose value is underscored in [GH11]) and by going beyond the reliance on randomisation in carrying out destructive moves by introducing more advanced strategies for selecting these moves (as indicated in [GL97] and in our previous work on IG for scheduling on parallel machines).

7. Future Work

In this dissertation, we have performed an in-depth analysis of HMs-EA/SA for black-box problems in binary combinatorial optimisation and we have proposed new metaheuristics based on constructive methodologies for two challenging combinatorial optimisation problems: scheduling on non-identical parallel machines with minimising total weighted completion times and quadratic minimum spanning tree. In the previous section we have shortly mentioned some different results that we have obtained in each area, but still there exists more work to be done. Next, we present some future open research lines raised from the proposals made in this memory.

HMs-EA/SA for black-box problems.

The research line focused on HMs-EA/SA is indeed worthy of further studies. We will intend to explore three interesting avenues of research. Firstly, *multiobjective and constrained problems* are subject of great interest, existing in the literature a great number of proposals for dealing with this kind of problems based on EAs [Deb01, CLV06, Coe02] and SA [PO08, HF06]. Therefore, it is possible to adapt the design of HMs-EA/SA to this type of problems. Secondly, teamwork collaborative HMs-EA/SA are able to take advantage of parallel hardware (multicore processors, clusters, etc.) and software [CMT04] that has become very affordable and widely available nowadays. This clearly favors the *implementation on parallel hardware* of HMs-EA/SA [WWR05, MG95, CFW98] that may lead to improved results due to the speed-up in the search process, which becomes a very appealing option for dealing with large-scale optimisation tasks. Finally, an important issue when combining different algorithms concerns the number of evaluations that each algorithm should consume throughout the run to create the conditions for the appearance of collaborative synergies among all the composing algorithms. Adaptive strategies that identify the best performing technique at each phase of the evolution with a minimum overhead [MS10, VRH09, LMPn10] may be used to build *adaptive HMs-EA/SA* with the aim of allowing profitable synergies to arise from the adjusted intervention of EAs and SA.

Hybrid and constructive metaheuristics for other combinatorial optimisation problems.

There are in the literature multiple variants of the scheduling on parallel machines problem that appear when we add specific conditions and information of the scheduling such as due dates, job release dates, setup times, preemptive scheduling, and precedence constraints. Considering all these constraints in the parallel machines scheduling problem makes the approach to the problem more faithful to the different conditions that can be found in real-world applications. However, at the same time, the complexity of the problem grows significantly. Thus, we intend to modify the different approaches presented in this thesis in order to consider some of these constraints.

Secondly, as we mentioned in Section 2.3, there are different objectives when dealing with the scheduling on parallel machines problem: minimise total weighted completion times, minimise maximum makespan or maximum lateness, and minimise total weighted tardiness. Most of the works presented in the literature have focused on a single objective that is optimised independently, however, many industries such as aircraft, electronics, semiconductors manufacturing, etc., have tradeoffs to their scheduling problems where multiple objectives need to be considered in order to optimise the overall performance of the system. Thus, an interesting avenue of research would be the application, when considering more than one simultaneous objectives, of multi-objective GRASP models that have recently appeared in the literature [RMVCD11] or population-based IG models [BBB12] adapted for dealing with multiple objectives.

Finally, we intend to apply hybrid and constructive metaheuristics to tackle other challenging optimisation problems such as the *maximum diversity grouping problem* [GLMD11], *permutation flowshop scheduling* [RS08], and *community detection in complex networks* [HWJ⁺12].

Bio-inspired metaheuristics

During the past half century, a set of metaheuristics based on natural evolution and swarm intelligence has appeared, showing a crucial impact on the optimisation field. We can highlight EAs, ant colony optimisation, and particle swarm optimisation. In recent years, a new set of bio-inspired techniques has appeared as result of the growing interest in discovering new strategies used by living beings and whose implementation can provide benefits in the field of optimisation. Within this new generation of bio-inspired methods, we can highlight algorithms such as *artificial bee colony* [KGOK12], *artificial immune systems* [LZT11], *biogeography-based optimisation* [Sim08], and *bacterial foraging optimisation* [DDAB09]. Our goal in this research avenue is twofold: 1) First, we intend to analyse the performance of this new bio-inspired methods on the problems studied in this thesis and new ones and 2) Secondly, we aim to explore new hybrid metaheuristics combining novel bio-inspired methods and other metaheuristics analysed in this thesis: HMs-EA/SA, GRASP, and IG.

Part II. Publications: Published, Accepted and Submitted Papers

1. A simulated annealing method based on a specialised evolutionary algorithm

The journal paper associated to this part is:

- C. García-Martínez, M. Lozano, F.J. Rodríguez, A simulated annealing method based on a specialised evolutionary algorithm. *Applied Soft Computing* 12:2 (2012) 473-488. doi: 10.1016/j.asoc.2011.11.007.
 - Status: **Published**.
 - Impact Factor (JCR 2011): 2.612.
 - Subject Category: Computer Science, Artificial Intelligence. Ranking 13 / 111 (Q1).
 - Subject Category: Computer Science, Interdisciplinary Applications. Ranking 16 / 99 (Q1).



A simulated annealing method based on a specialised evolutionary algorithm

C. García-Martínez^{a,*}, M. Lozano^b, F.J. Rodríguez-Díaz^b

^a Department of Computing and Numerical Analysis, University of Córdoba, 14071, Spain

^b Department of Computer Science and Artificial Intelligence, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, 18071, Spain

ARTICLE INFO

Article history:

Received 17 March 2010
 Received in revised form 15 May 2011
 Accepted 1 November 2011
 Available online 15 November 2011

Keywords:

Simulated annealing
 Evolutionary computation
 Combinatorial optimisation
 Hybrid algorithms

ABSTRACT

The flexible architecture of evolutionary algorithms allows specialised models to be obtained with the aim of performing as other search methods do, but more satisfactorily. In fact, there exist several evolutionary proposals in the literature that play the role of local search methods. In this paper, we make a step forward presenting a specialised evolutionary approach that carries out a search process equivalent to the one of *simulated annealing*. An empirical study comparing the new model with classic simulated annealing methods, hybrid algorithms and state-of-the-art optimisers concludes that the new alternative scheme for combining ideas from simulated annealing and evolutionary algorithms introduced by our proposal may outperform this kind of hybrid algorithms, and achieve competitive results with regard to proposals presented in the literature for binary-coded optimisation problems.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Metaheuristics (MHs) [1,2] arose around 30 years ago, extending basic heuristic methods by means of their application in iterative frameworks that augmented their exploration capabilities. Blum and Roli [3] offer an overview of various existing methods. These stochastic algorithms have proved to be useful when coping with difficult optimisation problems because they usually obtain good solutions in a reduced amount of time.

Over the last years, a large number of search algorithms were reported that do not purely follow the concepts of one single classical MH, but they attempt to obtain the best from a set of MHs (and even other kinds of optimisation methods) that perform together and complement each other to produce a profitable synergy from their combination. These approaches are commonly referred to as *hybrid MHs* [4–6].

Simulated annealing (SA) [7,8] is commonly said to be the first algorithm extending local search methods with an explicit strategy to escape from local optima. The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution in order to escape from local optima. The probability of doing such a move is decreased during the search process. Despite of being proposed in 1983, SA is still object of further studies, tool tackling many optimisation problems, and component of other search algorithms [9–14]. Precisely, its outstanding role in the

MH field encourages further studies to obtain more effective SA models.

Evolutionary algorithms (EAs) [15,16] rely on the concept of a population of individuals, which undergo probabilistic operators such as mutation, selection, and (sometimes) recombination to evolve toward increasingly better fitness values of the individuals. Genetic Algorithms, Evolutionary Strategies, Evolutionary Programming and Genetic Programming are examples of this family of MHs.

The flexibility offered by the evolutionary paradigm allows specialised models to be obtained with the aim of performing as other search methods do, but more satisfactorily. This practice is a recent alternative to design new hybrid MHs, and an overview of *specialised EAs* on intensification and diversification can be found in [17]. In particular, there exist several evolutionary proposals acting as local search methods and performing more effectively [18–21].

The design of hybrid MHs with ideas from the SA and EAs fields is a fruitful research line. There exist several proposals in the literature that either introduce the acceptance mechanism of SA in an EA [22,23], or apply a SA algorithm to optimise the members of the population of an EA [24], or exploit the advantage of a population of SA processes [25,26]. In this paper, we present *SA based on a specialised EA* (SASEA), an innovative hybrid MH that brings out an alternative scheme for combining ideas from SA and EAs. SASEA can be seen as either a specialised EA that carries out the search process typical of SA, or a SA approach based on ideas of the EA field. It consists in a steady-state EA [27,28] that creates one single candidate solution at each iteration, by crossing over a given solution, or current solution, and another one from the population. Afterwards, SASEA applies an acceptance mechanism to decide which solution

* Corresponding author. Tel.: +34 957 212660; fax: +34 957 218630.
 E-mail address: cgarcia@uco.es (C. García-Martínez).

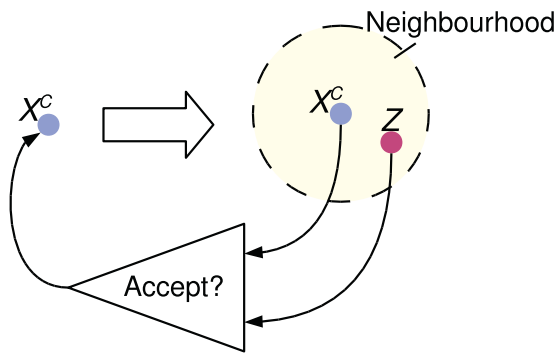


Fig. 1. SA model.

becomes the new current solution, either the candidate solution or the current one. The other solution is inserted into the population by a replacement strategy.

We have carried out an experimental study, deeply extending the one presented in García-Martínez and Lozano [29], on a test suite composed of binary combinatorial optimisation problems, with the aim of analysing the benefits of SASEA with regard to other SA approaches, hybrid MHs with ideas from SA and EAs, and other state-of-the-art search algorithms for this kind of problems. We have concluded that the innovative scheme for combining ideas from SA and EAs introduced by SASEA may improve the performance of this kind of hybrid MHs, and achieve competitive results with regard to state-of-the-art search algorithms.

The paper is set up as follows. In Section 2, SA is introduced. In Section 3, we present SASEA. Its components are deeply described. In Section 4, we define the empirical framework used to analyse the benefits of SASEA. In Section 5, we analyse the neighbourhood structure introduced by SASEA with regard to related work presented in the literature. We also conduct an empirical comparison with two SA proposals and CHC. In Section 6, we outline the differences between SASEA and other hybrid MHs combining ideas from EAs and SA. The benefits of SASEA with regard to those hybrid MHs are empirically studied. In Section 7, we pit SASEA against state-of-the-art search algorithms for binary-coded problems. Finally, conclusions and future works are presented in Section 8.

2. Simulated annealing

SA [8,7] is an optimisation technique analogous to the physical process of annealing. SA starts with a high temperature T and any initial state (X^c). A neighbourhood operator is applied to the current state X^c (having energy $f(X^c)$) to yield state Z (having energy $f(Z)$) (see Fig. 1). When tackling binary combinatorial optimisation problems, the most frequently applied neighbourhood operator is the *one-flip*, which flips one bit of the current solution. Then, an acceptance mechanism decides which state becomes the new current state. Acceptance mechanism takes into account the current temperature of the system: worse solutions are more often accepted with high temperature. The application of the neighbourhood operator and the probabilistic acceptance of newly generated states are repeated either a fixed number of iterations or until a quasi-equilibrium is reached. The entire above-described procedure is performed repeatedly, each time starting from the current state and from a lower T .

The idea behind SA is to protect diversification in the initial stages and intensification later: at the initial stages, high T values favour the exploration of the search space, accepting new states almost regardless their energy; later, low T values increase exploitation, accepting only new better states.

There are two main acceptance mechanisms proposed in the literature: *metropolis* and *logistic* rules [30]. They define the probability of Z being the new current state. Eqs. (1) and (2) show *metropolis* and *logistic* acceptance mechanisms, respectively, for minimisation problems. On the other hand, there exist several cooling schemes, such as *logarithmic*, *fast*, and *geometric* ones [30]. Eq. (3) shows *geometric* cooling scheme, which is used in this work (α is a cooling factor often assumed to be a constant in the interval $[0.9, 1)$):

$$p(Z) = \begin{cases} 1 & \text{if } f(Z) < f(X^c) \\ e^{(f(X^c)-f(Z))/T} & \text{if } f(Z) \geq f(X^c) \end{cases} \quad (1)$$

$$p(Z) = 1 - \frac{1}{1 + e^{(f(X^c)-f(Z))/T}} \quad (2)$$

$$T \leftarrow \alpha \cdot T \quad (3)$$

Convergence to global optimum is guaranteed for SA if a stationary distribution is reached at each temperature, followed by sufficiently slow cooling, such as logarithmic [31]. However, it usually leads to excessive long runs and practitioners use to apply faster cooling schemes at fixed number of iterations [32].

3. SA based on a specialised EA

SASEA is a *steady-state* EA that creates one single candidate solution at each iteration, by crossing over the current solution and another one from the population. Afterwards, SASEA applies an acceptance mechanism, e.g. *metropolis* or *logistic*, to decide which solution becomes the new current solution, either the candidate solution or the current one. The other solution is inserted into the population by a replacement strategy.

An outstanding feature of SASEA is that it describes a *trajectory* in the search space, as classical SA procedures do. Both, classical SA algorithms and SASEA, commence from a single solution and, at each step, a candidate solution is generated using a neighbourhood operator of some sort. They simply move the search from the current solution to a candidate solution according to the acceptance mechanism. The basic idea of SASEA is to use a SA accepting mechanism as the move accepting criterion of the search and crossover as the neighbourhood operator. In fact, the neighbourhood structure defined by the crossover operator of SASEA, together with the individual selected from the population, is one of the most important differences with regard to classic SA methods. Several researchers have pointed out that crossover operators may promote neighbourhood structures [33–36]. Precisely, first trajectory methods based on crossover were suggested in 1995 [37,38], and they have been applied to obtain different *local EAs* [20,21]. The main novelty of SASEA concerns the application of these ideas to design a new SA model.

SASEA can be seen as an extension of *Binary-coded Local Genetic Algorithm* (BLGA) [39,19], a recent EA that performs local search to an external solution. BLGA was compared with classic local search methods obtaining promising results with regard to efficacy and efficiency. SASEA extends BLGA by including the concepts that distinguish SA from local search methods: *temperature*, *acceptance criterion*, and *cooling scheme*. In addition, some components are modified in order to better fit the SA search process.

In Section 3.1, we describe steady-state EAs. In Section 3.2, we introduce the general scheme of SASEA. Sections 3.3, 3.4 and 3.5 describe in detail, selection mechanism, crossover operator and replacement strategy of SASEA, respectively.

Input: Pop, f
Output: Pop

```

{P1, P2} ← Select two parents from Pop;
Z ← Create an offspring applying crossover to the {P1, P2};
f(Z) ← Evaluate Z with the fitness function;
R ← Select an individual from Pop, which may be replaced by Z;
Decide if R is replaced (Pop ← (Pop \ {R}) ∪ {Z});

```

Fig. 2. Basic scheme of steady-state EAs.

3.1. Steady-state EAs

The generational EA creates new offspring from the members of an old population using the genetic operators and places these individuals in a new population that becomes the old population when the whole new population is created. The *steady-state EA* [27,28] is different to the generational model in that there is typically one single new member inserted into the new population at any one time. A *replacement/deletion* strategy defines which member in the current population is forced to perish (or vacate a slot) in order to make room for the new offspring to compete (or, occupy a slot) in the next iteration. Steady-state EAs are overlapping systems, since parents and offspring compete for survival.

The basic algorithm step of steady-state EAs is shown in Fig. 2. These steps are repeated until a termination condition is achieved. In step 4, one can choose the *replacement strategy* (e.g., replacement of the worst, the oldest, or a randomly chosen individual). In step 5, one can choose the *replacement condition* (e.g., replacement if the new individual is better or unconditional replacement).

3.2. General scheme of SASEA

Fig. 3 outlines the pseudocode of SASEA. It starts with a high temperature T , an initial state X^c , and a random uniformly generated population (Pop). Then, the following steps are carried out (see Fig. 4):

1. *Mate selection*: an individual Y is selected from the population applying *parametrised assortative mating* (Section 3.3).
2. *Crossover*: X^c is crossed over with Y by means of *guiding neighbourhood exploration crossover operator*, generating offspring Z (Section 3.4). Notice that the crossover-based neighbourhood structure has been depicted as a cloud in Fig. 4 to stress the difference with classic neighbourhood structures of SA.
3. *Acceptance*: Z replaces X^c according to the applied acceptance mechanism (metropolis, logistic, etc.).

Input: f
Output: X^c , Pop

```

Initialise temperature, current solution  $X^c$ , and Pop;
while stop condition is not reached do
  Y ← Select an individual from Pop;
  Z ← Cross over  $X^c$  and Y;
  Evaluate Z;
  if Z is accepted as new  $X^c$  then
    Insert  $X^c$  in Pop;
     $X^c$  ← Z;
  else
    Insert Z in Pop;
  end
  Anneal temperature if cooling condition is reached;
end

```

Fig. 3. Pseudocode of SASEA.

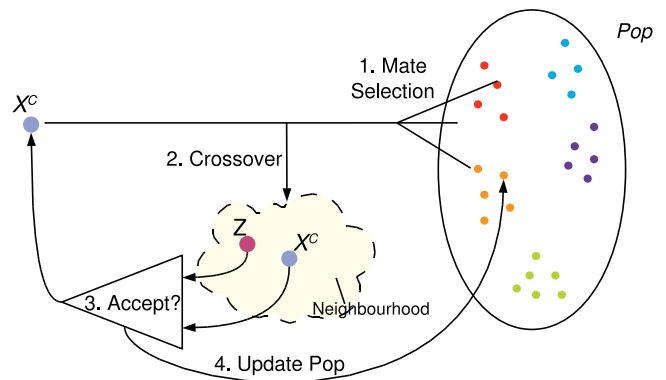


Fig. 4. SASEA model.

4. *Update population*: if Z replaced X^c , then old X^c is inserted into the population using *restricted tournament selection* (Section 3.5). Otherwise, Z is inserted in the population using the same replacement scheme. This replacement strategy helps SASEA to maintain a diverse set of solutions in Pop .
5. *Annealing*: if the cooling condition has been reached, apply the selected cooling scheme, e.g. geometric cooling scheme.

All these steps are repeated until termination condition is reached.

3.3. Parametrised assortative mating

Assortative mating is the natural occurrence of mating between individuals of similar phenotype more or less often than expected by chance. Mating between individuals with similar phenotype more often is called positive assortative mating and less often is called negative assortative mating. Fernandes and Rosa [40,41] implement these ideas to design two mating selection mechanisms. A first parent is selected by the roulette wheel method and n_{ass} individuals are selected with the same method. Afterwards, similarity between each of these individuals and first parent is computed (similarity between two binary-coded solutions is defined as the Hamming distance between them). If assortative mating is negative, then the one with less similarity is chosen. If it is positive, the genome more similar to the first parent is chosen to be the second parent.

We introduce a new mating mechanism, called *parametrised assortative mating*, that probabilistically regulates similarity between mated individuals. First parent is always X^c and second parent is chosen being more or less similar to X^c according to a parameter of the operator (p) (ties are broken up by selecting one element uniformly at random). Pseudocode of parametrised assortative mating is presented in Fig. 5 ($Random(Pop)$, uniformly samples a solution from Pop).

Parameter p adjusts likeness mating trends of X^c . This parameter takes values in the interval $[0, 1]$. If p is 0, X^c tends to mate dissimilar individuals and as p moves closer to 1, X^c tends to mate individuals more alike.

With the aim of preserving SA idea, to protect diversification in the initial stages and intensification later, SASEA applies a deterministic adaptation rule [42] on parameter p . At each iteration, it is set to the fraction of consumed evaluations:

$$p = \frac{FES}{Max.FES}, \quad (4)$$

where FES is the number of consumed evaluations and $Max.FES$ is the maximum number of evaluations allowed. The objective is to mate dissimilar individuals at the initial stages of the search

```

Input:  $X^c$ ,  $n_{\text{ass}}$ ,  $p$ , Pop
Output:  $Y$ 

 $R \leftarrow \emptyset$ ;
 $D \leftarrow \emptyset$ ;

for  $i \leftarrow 0$  to  $n_{\text{ass}} - 1$  do
     $r \leftarrow \text{Random}(\text{Pop})$ ;
     $\text{dist}_r \leftarrow \text{distance}(X^c, r)$ ;
     $R \leftarrow R \cup \{r\}$ ;
     $D \leftarrow D \cup \{\text{dist}_r\}$ ;
end

 $\text{maxDiff} \leftarrow \text{maximum}(D)$ ;
 $\text{minDiff} \leftarrow \text{minimum}(D)$ ;
 $\text{range} \leftarrow \text{maxDiff} - \text{minDiff}$ ;
 $\text{selectedDiff} \leftarrow \text{maxDiff} - \text{range} \cdot p$ ;
 $Y \leftarrow \text{find member in } R \text{ with } \text{dist}_r \text{ the closest to } \text{selectedDiff}$ ;

```

Fig. 5. Parametrised assortative mating.

process, when p is close to 0, favouring diversification. Then, to gradually tend to mating similar individuals at latter stages, because p linearly approaches value 1, increasing intensification.

3.4. Guiding neighbourhood exploration crossover operator

SASEA applies a specific crossover, called *guiding neighbourhood exploration crossover operator*, that is aimed at guiding the exploration of the neighbourhood of X^c . It carries through two stages. At the first stage, offspring is pushed from X^c toward the mate Y . Y can be seen as an agent guiding the exploration of X^c neighbourhood. Y also defines the intensity by which offspring is pushed away. If Y is close to X^c , offspring is sampled next to X^c , otherwise, if Y is far from X^c , offspring undergoes a stronger drive. At the second stage, offspring experiences a stochastic perturbation depending on the intensity of previous drive. The aim of the second stage is to add randomness to neighbourhood exploration. This perturbation may reduce, enforce previous drive, or even add new genetic material. Fig. 6 represents the operation of the crossover operator.

Fig. 7 outlines pseudocode of guiding neighbourhood exploration crossover operator. `Random` returns one element from the given set, uniformly sampled at random, and N refers to the number of bits of the problem. Given X^c and a mate Y :

1. The operator initially creates offspring Z as a copy of X^c and detects differences with Y .

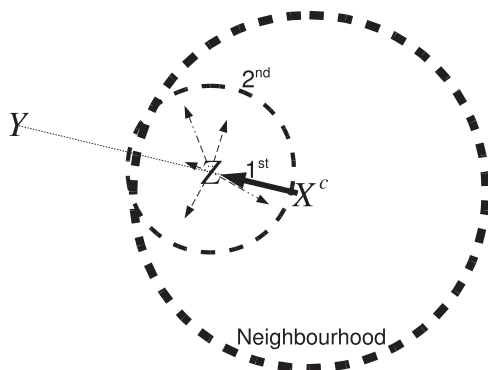


Fig. 6. Guiding neighbourhood exploration crossover operator.

2. First stage: genes from Y , where Y and X^c differ, are included in Z with probability p_y , parameter of the operator.
3. Second stage: genes are flipped according to a probability that depends on the number of genes from Y included in previous stage.
4. At last, if offspring Z has not included genes from Y nor experienced perturbation, a random uniformly chosen gene is flipped.

It is interesting to mention that this new crossover operator shares some similarities with *half uniform crossover operator* (HUX) of CHC [43]. In particular, first stage of guiding neighbourhood and half uniform crossover operators actuate according to the differences detected between both parents. Whereas half uniform operator assigns exactly half of these differences to each produced

```

Input:  $X^c = (x_0^c, \dots, x_{N-1}^c)$ ,  $Y = (y_0, \dots, y_{N-1})$ ,  $p_y$ 
Output:  $Z = (z_0, \dots, z_{N-1})$ 

 $Z \leftarrow X^c$ ;
 $\text{diffs} \leftarrow \{i \mid x_i^c \neq y_i\}$ ;
 $\text{numDiffs} \leftarrow |\text{diffs}|$ ;
 $\text{numChanges} \leftarrow 0$ ;

/* First stage: Pushing toward Y */
for  $i \leftarrow 0$  to  $\text{numDiffs} - 1$  do
    if  $\text{Random}([0,1]) < p_y$  then
         $\text{index} \leftarrow \text{Random}(\text{diffs})$ ;
         $\text{diffs} \leftarrow \text{diffs} \setminus \{\text{index}\}$ ;
         $z_{\text{index}} \leftarrow y_{\text{index}}$ ;
         $\text{numChanges}++$ ;
    end
end

/* Second stage: Random perturbation */
for  $i \leftarrow 0$  to  $N - 1$  do
    if  $\text{Random}([0,1]) < \text{numChanges}/N$  then
         $z_i \leftarrow 1 - z_i$ ;
         $\text{numChanges}++$ ;
    end
end

/* Forced change */
if  $\text{numChanges} = 0$  then
     $\text{index} \leftarrow \text{Random}(\{0, \dots, N-1\})$ ;
     $z_{\text{index}} \leftarrow 1 - z_{\text{index}}$ ;
end

```

Fig. 7. Pseudocode of guiding neighbourhood exploration crossover operator.


```

Input: Pop, I,  $n_T$ 
Output: Pop

 $G_T \leftarrow$  Select  $n_T$  random members from Pop;
 $R \leftarrow$  Find member from  $G_T$  most similar to I;

if I is better than R then
  | Pop  $\leftarrow$  (Pop  $\setminus$  R)  $\cup$  I;
end

```

Fig. 8. Restricted tournament selection.

offspring, guiding neighbourhood exploration crossover operator considers control parameter p_y for that assignation.

3.5. Restricted tournament selection

Crowding methods [44,45] attempt to maintain diversity in the population by means of the replacement procedure as follows: new individuals are more likely to replace existing individuals in the parent population that are similar to themselves based on genotypic similarity. They have been used for locating and preserving multiple local optima in multimodal functions.

SASEA uses *restricted tournament selection* [46] as the replacement method. This strategy was applied previously by BLGA [39,19]. Its main idea is to replace the closest individual R to the one being inserted in the population, I , from a set of n_T randomly selected ones, uniformly and with-replacement, if I is better than R . Possible ties are broken up by choosing one individual uniformly at random. Fig. 8 outlines its pseudocode.

4. Empirical framework

We have carried out different experiments to analyse and assess the efficacy of SASEA with regard to other algorithms for binary-coded problems. In this section, we detail the test problems (Section 4.1) and running conditions (Section 4.2) that were used for the following studies. Some comments on the *no free lunch theorems* [47] are addressed in Section 4.3.

4.1. Test problems

Experiments were executed on a test suite composed of 27 binary-coded test problem instances from 12 different problem classes, 8 classes (13 instances) from the artificial intelligence field and 4 (14 instances) from real-world applications. They are described in Appendix A. Table 1 outlines their name, number of bits, and a value (f^*) that stands for either the fitness value of the global optimum, known best solution, or upper bound presented in the literature. All of them have been formulated as maximisation problems. BQP and Maxcut instances can be obtained from the corresponding files from the *Biq Mac Library*,¹ and Multiple knapsack problems, from the *SAC-94 Suite*.² Though Table 1 indicates that 1 is the maximum possible fitness value for M-Sat and NkLand problems, there not usually exists any optimal solution with that fitness value, which depends on the current problem instance.

For problems where the global optimum is the all-ones string (Royal-road, Trap, Deceptive, Bipolar deceptive, Overlapping deceptive, and HIFF ones), we firstly build a random bit string S that will become the *shifted optimum* of the problem. The new fitness value of a candidate solution X is computed as the original fitness value of the string $X \oplus S$ (i.e., the result of applying first

Table 1
Tackled problems.

Prob.	Name	N	f^*
1	Royal road problem (400, 8)	400	1
2	Trap problem	36	220
3	Deceptive problem	39	390
4	Bipolar deceptive problem	396	1
5	Overlapping deceptive problem	399	1
6	NkLand(48,4)	48	1
7	NkLand(48,12)	48	1
8	HIFF(2, 5, true)	32	192
9	HIFF(3, 4, false)	81	211
10	PPeaks(10,100)	100	1
11	PPeaks(100,100)	100	1
12	PPeaks(50,150)	150	1
13	PPeaks(50,200)	200	1
14	M-Sat(100,1200,3)	100	1
15	M-Sat(100,2400,3)	100	1
16	BQP(bqp50-1)	50	5176
17	BQP(bqp500-1)	500	121,270
18	BQP(be120.3.3)	120	Not known
19	BQP(be200.8.5)	200	Not known
20	Maxcut(pm1s.80.6)	80	73
21	Maxcut(w09_100.2)	100	2738
22	Maxcut(g05_100.5)	100	1436
23	Maxcut(pw05_100.6)	100	8217
24	Maxcut(ising2.5-250.5555)	250	7,919,449
25	Multiple knapsack p. (weish03)	30	4115
26	Multiple knapsack p. (pet5)	28	12,400
27	Multiple knapsack p. (pb4)	29	95,168

exclusive OR between X and S , and subsequently, inversion). This way, we avoid the biased advantages of some search algorithms that tend to sample the all-ones string at the beginning of the search process.

4.2. Running conditions

One advantage of trajectory MHs over other heuristics is that, when solving some problems, the search space can be searched very efficiently: instead of calculating the objective value of a new candidate solution, it is sufficient to compute the difference Δf with regard to the fitness of the current solution by utilising problem-specific knowledge. This technique, widely known as *delta evaluation* [48], is highly profitable in terms of computation time whenever feasible. For example, in the BQP problem, the calculation of Δf takes time $O(n)$, while the calculation of the fitness takes $O(n^2)$. As delta evaluation can be performed on several of the tackled problems (and many others [49]), it will be applied by all the trajectory search methods (or trajectory search components) considered. In particular, SASEA will apply delta evaluation as well, which is possible due to guiding neighbourhood exploration crossover operator, applying a low p_y value, creates offspring near the first parent, i.e., few bits are changed.

In most real-world optimisation problems the evaluation of a solution requires a simulation process that is usually very time-consuming, i.e., solution evaluation is the bottleneck of the search process. For that reason, in order to perform a fair comparison between different search methods, we will run every algorithm with the same budget of fitness evaluations (independently delta evaluation is being applied or not, which does not affect the algorithms' performance in terms of efficacy). In general, each run of a search algorithm on a test problem will perform at most 10^5 fitness evaluations, but convergence characteristics will be studied as well. The performance measure is the average of the best fitness value found over 50 independent runs, because we are interested in the regular performance of the compared algorithms, instead of the best results attained.

¹ <http://biqmac.uni-klu.ac.at/>.

² <http://elib.zib.de/pub/Packages/mp-testdata/ip/sac94-suite/>.

Non-parametric tests can be used for comparing the results of different optimisers [50–52]. Given that the non-parametric tests do not require explicit conditions for being conducted, it is recommended that the sample of results are obtained following the same criterion, which is, to compute the same aggregation (average, mode, etc.) over the same number of runs for each search algorithm and problem. In particular, we have used the *Wilcoxon matched-pairs signed-ranks* [53] test to compare the results of our proposal with the ones of other MHs. We explain this statistical test, in detail, in Appendix B.

4.3. On the no free lunch theorem

No free lunch theorem for optimisation [47] states that all not-resampling algorithms that search for an extremum of a cost function in a finite space perform exactly the same when averaged over all possible fitness functions.

Since, the application of not-resampling algorithms on problems with many variables usually involves impossible memory and/or computation requirements (a general binary-coded problem with N variables requires 2^N evaluations to be solved to optimality), resampling is allowed for most MHs. Though, no free lunch theorem is not applicable to those cases, there is a belief, as many researchers write so, that all (resampling) MHs might perform exactly the same when averaged over all possible fitness functions, too. In particular, proposals' deficiencies are often explained by sentences like "*According to the no free lunch theory, it is impossible for algorithm X to outperform always*", even when they are being proposed. This fact points out the significance of this theorem. Recently, Marshall and Hinton [54] have proved that NFL theorems do not hold when considering resampling algorithms, however, random search with replacement is expected anyway to be the best approach when performances are averaged over all possible functions (or closed under permutation sets, as defined below).

Schumacher et al. [55] revealed that no free lunch theorem even holds when averaging over subclasses of problems if and only if these sub-classes are *closed under permutations* of the search space. F is said to be a subclass of functions closed under permutations if for any function $f \in F$ and any permutation π of the search space, $f \circ \pi$ is also in F (notice that the number of permutations of a binary-coded search space with N variables is 2^N !). Later, Igel and Toussaint [56] suggested that the class of real-world applications is hardly closed under permutations. In particular, they proved that the class of problems with some topological structures based on nontrivial neighbourhood relations is not closed under permutations. This result implies that it might be possible to detect performance differences between some algorithms when averaged over the class of all real-world problems.

Since, our testbed is not closed under permutations, because there exist some neighbourhood relations between the solutions in their search spaces (notice that the procedure for shifting the optimum of several problems does not close them under permutations because neighbourhood relations are not affected and, just because a maximum of 2^N transformations are possible), it is expected that the compared algorithms might not perform exactly the same on average, and we could rank them according to the average performance. Our intention is to analyse whether SASEA is able to outperform its competitors when averaged on the considered testbed and under the specified running conditions, not to defy the no free lunch theorem, but to forecast similar performances on other real-world problems (with similar neighbourhood relations) with regard to the compared algorithms. On the contrary, we know that the set containing all the binary static combinatorial problems is really closed under permutations, and thus, any averaged performance difference should not to be expected. Finally, we shall

mention that, the practice followed in this work is widely applied by the evolutionary computation community [57,58,34,59–63].

5. Study of the neighbourhood structure of SASEA

The neighbourhood structure of SASEA, as the result of the combination of guiding neighbourhood exploration crossover operator and the way parametrised assortative mating selects individuals from the population, is a major difference with classic SA methods. In Section 5.1, we point out the properties of the neighbourhood structure promoted by SASEA. In Section 5.2, we empirically study whether the new neighbourhood structure of SASEA outperforms other SA neighbourhood structures for binary-coded problems proposed in the literature. Besides, we empirically analyse the performance synergy between guiding neighbourhood exploration crossover operator and SASEA with regard to the one between HUX and CHC in Section 5.3, according to the proper adjustment of its parameter p_y .

5.1. Properties of the neighbourhood structure of SASEA

The neighbourhood structure promoted by SASEA presents the following two properties:

- *It explores larger and dynamic neighbourhood sizes*: classic SA algorithms for binary optimisation usually consider one-flip operator, which flips just one bit of the current solution, to explore the neighbourhood of X^c . However, SASEA may flip several bits of X^c and the number of flips is adapted dynamically at each iteration. When X^c is crossed over with dissimilar individuals, offspring undergoes strong drives from X^c . This fact occurs at the initial stages of the search process because p is near to 0 and parametrised assortative mating tries to mate X^c with unlike individuals. On the opposite, when X^c and the mate are similar, offspring is sampled close to X^c . It happens at last stages because p is near to 1 and parametrised assortative mating attempts coupling X^c with similar individuals.

This is a promising characteristic because Yao [64] theoretically proved that employing larger and dynamic neighbourhood operators may enhance efficiency of SA. In fact, in Liu [65], the impact of neighbourhood size on the performance of SA is analysed and another SA algorithm with dynamic neighbourhood sizes is proposed for flowshop scheduling problems. We shall mention as a note that, though the ideas in Yao [64] and in SASEA point out in the same direction, the study in Yao [64] does not explicitly cover crossover-based neighbourhood structures.

- *Neighbourhood exploration is guided by the mate to promote diversification and/or intensification*: at the first stage, crossover operator pushes Z from X^c toward the mate. Thus, mate guides the exploration of X^c neighbourhood. At this point, parametrised assortative mating becomes determinant: at initial stages of the search process, it intends to promote diversification by coupling X^c with dissimilar individuals; whereas at latter stages, it attempts promoting intensification by pairing X^c with similar individuals.

This characteristic has also been considered beneficial for the performance of SA in previous works. Fox [66] states that neighbourhood exploration might be guided by non-uniform probability distributions. Instead of this blind exploration process, he suggests that modified distributions might promote diversification and/or intensification. A particular example for continuous optimisation problems is found in [67]. The presented neighbourhood exploration favours diversification in the initial stages of the search process by sampling opposite neighbours.

Table 2

SASEA vs. classic SA and dynamic SA (Wilcoxon's test with p -value = 0.05 and critical value = 107).

Algorithms	R^+	R^-	Sig. differences?
SASEA vs. classic SA	289.5	88.5	+
SASEA vs. Dyn. SA	377.5	0.5	+

5.2. Comparison with other SA neighbourhood structures

With the aim of analysing the significance of the neighbourhood structure of SASEA, we compare the results of the proposal with the ones of two other SA algorithms with different neighbourhood structures, classic SA and dynamic SA [65], on the experimental framework described in Section 4. Though SASEA was previously compared with classic SA in [29], the latter is included here for the sake of completeness.

Initial temperature was set in the following manner for all the search algorithms: firstly, two random solutions are generated. We set a desired probability p_d of accepting the worst solution from the best one. Then, we compute the corresponding T_0 value, according to the applied acceptance criterion. This procedure, which has not been proposed before as far as we know, allows us to probabilistically get an appropriate value for the initial temperature according to the range of the fitness function and a desired initial probability for worse solutions, without consuming many evaluations.

Initial experiments comparing SASEA and classic SA showed that geometric cooling and any of the two acceptance mechanisms (metropolis or logistic) were the combinations that obtained the best results for both models, classic SA and SASEA [29]. Besides, best results were obtained with p_d equal to 0.4, 100 iterations per cooling event, and 0.99 as the cooling factor. Therefore, all methods will apply geometric cooling, logistic acceptance mechanism, and the mentioned parameters values.

Dynamic SA applies the following formula [65] to compute the probability of flipping each bit of X^c :

$$1 + \left(\frac{N}{2} - 1\right) \left(1 - \frac{FEs}{Max_FEs}\right)^{0.4}, \quad (5)$$

where N is the number of bits of the problem, FEs is the number of evaluations consumed, and Max_FEs is the maximum number of evaluations.

SASEA maintains 500 individuals in the population and n_T is set to 15 for restricted tournament selection (parameter values taken from BLGA [39]). In addition, p_y , for the crossover operator, has been set equal to $3/N$, and n_{ass} to 10 for parametrised assortative mating.

Tables 8–10, in Appendix C, outline the results of the studied algorithms when tackling each test problem. Table 2 summarises the results of applying the Wilcoxon matched-pairs signed ranks test for p -value = 0.05, where the values of R^+ (associated to SASEA) and R^- of the test are specified. If R^- is smaller than R^+ and the critical value of the test, SASEA is statistically better than the corresponding algorithm; if R^+ is inferior to R^- and the critical value, SASEA is statistically worse than its competitor; if neither R^+ nor R^- is smaller than the critical value, Wilcoxon test does not find statistical differences (p -value = 0.05 and 27 problem instances make critical value to be 107). Last column indicates whether SASEA performs statistically better (+), worse (−), or without significant differences (~) than its competitor.

From Table 2, we clearly notice that the optimisation process performed by SASEA is statistically superior to the ones of the other two algorithms. It is also interesting to see that SASEA outperforms dynamic SA on almost every test problem because the associated R^- value is close to 0.

We have carried out a statistical analysis for p -value equal to 0.05 to measure the performance differences between SASEA and

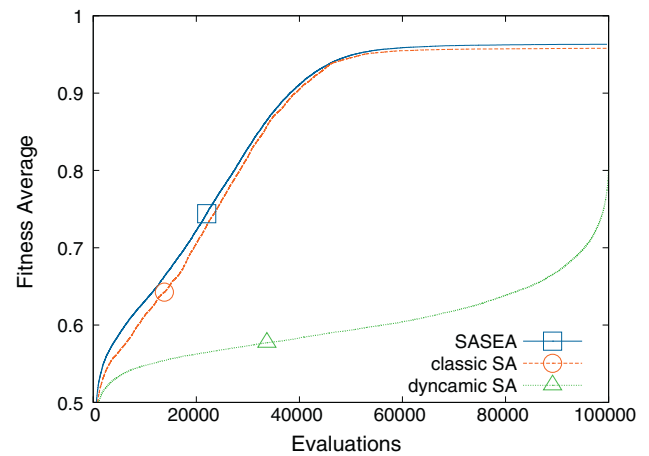


Fig. 9. Convergence graphs of SA algorithms with different neighbourhood structures.

the other algorithms on each problem separately. This statistical analysis is based on *Mann–Whitney* test [68]. Plus (+), minus (−), and approximate (~) signs in Tables 8–10 indicate superior, inferior, or similar performance of SASEA, respectively, with regard to the corresponding algorithm on each problem. We notice that:

- SASEA obtains better results than the dynamic SA on almost all the problems considered. This fact is in accord with the results in Table 2.
- Classic SA and SASEA attain comparable results on many problems. However, SASEA shows a light superiority on several artificial problems (Royal road, Deceptive and Bipolar deceptive problems, and three PPeaks ones) and two Multiple knapsack ones. Classic SA achieves better results on just two problem instances (Overlapping deceptive problem and Maxcut(ising2.5-250.5555)).

Next, we study the behaviour of the three search algorithms along the whole run. In order to obtain a convergence graph summarising algorithms' behaviours on all the problems, we have accomplished the following two steps:

1. Taking into consideration highest and lowest fitness values achieved by all the studied algorithms on each test problem, we have normalised every result, along the whole runs, into the interval [0, 1].
2. Subsequently, mean values, over the 27 problem instances, have been obtained for each algorithm, along the 10^5 evaluations.

Fig. 9 shows the results for classic SA, dynamic SA, and SASEA.

From Fig. 9, we see that the results of SASEA are superior to both SA algorithms along the whole run. Besides, we know that, at least at the end of the run, the differences are statistically significant (Table 2). On the other hand, we observe that dynamic SA lasts too much time before leading the search process toward promising solutions, which occurs when the number of flips on X^c is low. This effect explains the poor results of dynamic SA in Tables 2 and 8–10.

The conclusion of this study is that the natural way SASEA implements the ideas commented in Section 5.1, dynamic neighbourhood sizes and guided neighbourhood exploration for diversification and/or intensification, becomes beneficial to outperform the results of other SA approaches.

Table 3
SASEA vs. CHC (Wilcoxon's test with p -value = 0.05 and critical value = 107).

Algorithms	R^+	R^-	Sig. differences?
SASEA(3/N) vs. CHC(0.5)	339.5	38.5	+
SASEA(3/N) vs. CHC(0.3)	346.5	31.5	+
SASEA(3/N) vs. CHC(3/N)	378	0	+
SASEA(0.3) vs. CHC(0.5)	219.5	158.5	~
SASEA(0.3) vs. CHC(0.3)	302.5	75.5	+
SASEA(0.3) vs. CHC(3/N)	378	0	+
SASEA(0.5) vs. CHC(0.5)	157.5	220.5	~
SASEA(0.5) vs. CHC(0.3)	233.5	144.5	~
SASEA(0.5) vs. CHC(3/N)	378	0	+

5.3. Guiding neighbourhood and SASEA vs. HUX and CHC

In this section, we address the proper adjustment of parameter p_y of guiding neighbourhood exploration crossover operator for the proper operation of SASEA, and compare it with the operation of HUX in CHC. The motivation is that both crossover operators initially work according to the differences between the chosen parents, however, our initial experiments suggested to set p_y to 3/N (this is the probability of selecting genes from parent Y), whereas the implicit parameter in HUX considers a value of 0.5 (it exchanges exactly half of the differences between both parents) [69].

We have performed experiments according to the empirical framework described in Section 4, comparing SASEA and CHC with different parameter values for their crossover operators. In particular, we will compare the results of both algorithms with every combination of values for $p_y \in \{3/N, 0.3, 0.5\}$. Notice that we have implemented a HUX version that exchanges a number of differences between the parents that depends on the given parameter p_y . The population of CHC consists of 50 individuals.

Table 3 presents every possible pairwise comparison between both algorithms by means of Wilcoxon statistical test with p -value = 0.05 (critical value is 107). Algorithms are denoted as SASEA(p_y) and CHC(p_y), respectively. R^+ and R^- values are presented for every comparison. If R^- is smaller than the critical value, then, Wilcoxon test finds statistical differences favouring SASEA. No statistical differences favouring CHC were obtained.

We observe that:

- SASEA(3/N) (which is the usual configuration) outperforms CHC regardless the parameter value considered.
- Though CHC(0.5) seems to outperform SASEA(0.5), Wilcoxon test does not find statistical differences.
- SASEA provides the best results when $p_y = 3/N$ (because it outperforms all the CHC approaches), i.e., when X^c is perturbed locally, which is at the foundation of classic SA methods. However, CHC obtains its best results when $p_y = 0.5$ (it is more competitive with regard to the SASEA instances), which coincides with its original conception [43,69], i.e., when it performs a global search, which is as well at the core of evolutionary algorithms. Nevertheless, when both algorithms are tuned, Wilcoxon finds statistical differences favouring SASEA.

To sum up, we may conclude that p_y has to be tuned according to the demands of the algorithm: X^c must be modified locally for trajectory methods such as SASEA, whereas a global exploration has to be provided for methods such as CHC.

6. SASEA vs. other hybrid MHs based on SA and EAs

Nowadays, there exist several hybrid MH models combining ideas from the EA and SA fields. Usually, their aim is to improve the performance of SA when it is applied in a limited resources

framework, i.e., a maximum allotted time, number of evaluations, etc. Several of these proposals are:

- **Parallel recombinative SA (PRSA)** [22]. It borrows the ideas from EAs to maintain several copies of SA running in parallel, with mutation as the neighbourhood operator, and crossover recombining independent solutions. PRSA may be seen as an EA, with random selection of parents, one-point-crossover and standard mutation operators, where offspring replace their parents according to the acceptance mechanism of SA.
- **Annealing-Genetic Algorithm (AGA)** [24]. It is a SA algorithm with the population-based state transition and with the genetic-operator-based quasi-equilibrium control. AGA can be understood as an EA where best solutions are optimised by SA processes. AGA selects parents based on their fitness values and applies two-point-crossover and mutation operators. Its inner SA procedure considers the metropolis acceptance mechanism. In addition, AGA includes a specific procedure to compute the initial temperature (T_0), and cooling scheme is applied at every iteration.
- **SA, Genetic Algorithm, Chemotaxis algorithm, Integrated Algorithm (SAGACIA)** [23]. SAGACIA is based on proper integration of SA, Genetic Algorithm, and Chemotaxis Algorithm for solving complex optimisation problems. It combines three mechanisms: rough search, fine search and disturbance one. At the first stage, new solutions are sampled in the neighbourhoods of the members of the population; rough search usually makes large jumps in the search space; then, new solutions replace their parents according to the metropolis criterion. At the fine search stage, the best member of the population is optimised by a local search method with a budget of N evaluations (the number of bits of the problem). At the latter stage, the members of the population are disturbed with a low probability by a mutation operator.

Besides, we found other models that implement the idea of a population of SA processes, which makes them to be similar to the aforementioned hybrid MHs:

- **Sample-Sort SA (SSSA)** [25]. SSSA maintains an array of samplers operating at static temperatures. At each iteration, each sampler firstly considers to accept any of the states of its neighbouring samplers; biased acceptance mechanisms are applied at this phase; subsequently, each sampler performs a standard iteration of SA. Metropolis acceptance mechanism is applied. SSSA includes a specific procedure to compute the temperature of first and last samplers. Afterwards, the temperature values of the other samplers are set according to $T_k = T_{last} \cdot \alpha^{m-k}$, where m is the number of samplers and α was previously computed satisfying that $T_{first} = T_{last} \cdot \alpha^m$.
- **Coupled SA (CSA)** [26]. It considers a population of samplers where the acceptance probability of any sampler making an uphill move depends on the states of the other samplers. In particular, when sampler i th generates a new solution Y_i from the state X_i , the probability of accepting such a move is:

$$\exp \frac{(f(X_i) - \max_{X \in \Theta} (f(X))) / T}{\sum_{X \in \Theta} \exp \left(\frac{f(X) - \max_{X \in \Theta} (f(X))}{T} \right)}, \quad (6)$$

where Θ is the set of current states of the samplers. Notice that the acceptance probability does not depends on $f(Y_i)$. Besides, CSA includes a procedure that controls the temperature of the system. The aim of this procedure is to maintain the variance of the acceptance probabilities of the samplers close to a desired variance value, which is computed as $0.99 \times (|samplers| - 1) / (|samplers|^2)$.

Table 4

SASEA vs. other hybrid MH (Wilcoxon's test with p -value=0.05 and critical value = 107).

Algorithms	R^+	R^-	Sig. differences?
SASEA vs. PRSA	378	0	+
SASEA vs. AGA	354.5	23.5	+
SASEA vs. SAGACIA	312.5	65.5	+
SASEA vs. SSSA	377.5	0.5	+
SASEA vs. CSA	378	0	+

All the search algorithms described in this section emphasise the evolution of a set of solutions in parallel. Thus, the notion of evolving populations from EAs is the centre of attention and SA ideas just define some actions on its members. SASEA is a new alternative to this scheme for combining EAs and SA. Search process performed by SASEA focuses on just one solution (X^c) that wanders over the search space looking for the best configuration. SA ideas govern this process, whereas EA components (i.e. mating selection mechanism, crossover operator, and replacement strategy), which operate on a population of solutions, just modifies X^c neighbourhood exploration. Because of that fact, SASEA can be seen either as a SA approach based on a specialised EA or as a specialised EA that carries out the search process of SA.

Now, we are interested in determining whether SASEA really provides an improved alternative to the schemes for combining EAs and SA. In order to do this, we pit SASEA against aforementioned algorithms. The experimental framework is the one described in Section 4. We have followed the recommendations of the original publications for the parameters settings:

- PRSA applies a mutation probability equal to $1/N$, which is high enough for the mutation operator to be regarded as the neighbourhood operator. Population size is set to 60, which is a standard value for EAs. On the other hand, the remainder design decisions were made as for SASEA: PRSA applies logistic acceptance criterion, geometric cooling with α set to 0.99, 100 iteration per cooling event, and p_d is 0.4.
- AGA uses a mutation probability equal to $1/(10 \cdot N)$, which is as low as it occurs in EAs. Population size is 60. In addition, it applies geometric cooling with α equal to 0.99 as SASEA.
- SAGACIA uses a population with 60 individuals. Mutation probability for the rough search stage is set to $10/N$, with the aim of producing large jumps. Mutation probability for the disturbance stage is 0.025. T_0 is computed as for SASEA with p_d set to 0.4.
- SSSA maintains 100 samplers. Neighbourhood size is 1, i.e., each sampler may accept any of the two states of its consecutive samplers (just one for the extreme samplers).
- CSA has 5 samplers. Its parameter α , for the variance control procedure, is set to 0.05. In addition, its acceptance probability formulae has been adapted for maximisation problems as:

$$\exp \left(\frac{\min_{X \in \Theta} (f(X)) - f(X_i)}{T} \right)$$

$$\gamma = \sum_{X \in \Theta} \exp \left(\frac{\min_{X \in \Theta} (f(X)) - f(X)}{T} \right), \quad (7)$$

Tables 8–10, in Appendix C, contain the results of the search algorithms when tackling each test problem. Table 4 summarises the results of applying the Wilcoxon matched-pairs signed ranks test for p -value = 0.05. The superiority of SASEA is clearly seen on the results presented in Table 4. Hence, the alternative scheme for combining ideas from SA and EAs introduced by SASEA becomes a promising research line to improve the performance of this kind of hybrid MHs. In particular, these results show that a single SA process consuming the allowed number of evaluations may obtain

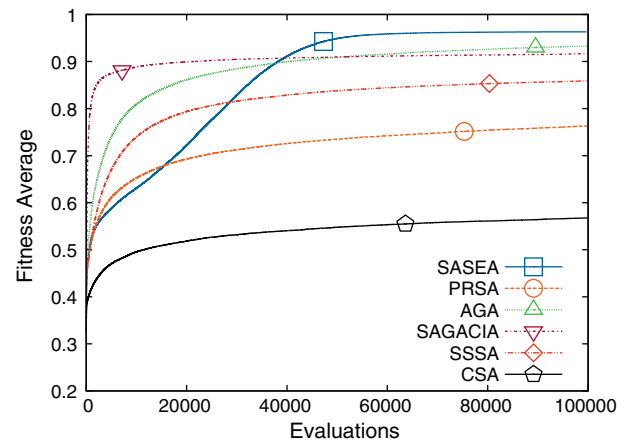


Fig. 10. Convergence graphs of hybrids MHs based on SA and EAs.

better results than a set of SA processes that compete for the computation resources.

Regarding the results of the algorithms on the problems separately (see Tables 8–10), we notice that SASEA generally outperforms its competitors, with the following exceptions:

- AGA attains similar results to those of SASEA on problems NkLand(48,4), BQP(bqp50-1), and Multiple knapsack problem(pb4). It is interesting to see that SASEA obtains the best results for those three problems and many algorithms get similar results (7, 6, and 10 out of 12 algorithms, respectively). Therefore, these problems seem to be easy to optimise. On the other hand, AGA outperforms SASEA on just one problem, HIFF(2,5,true).
- SAGACIA achieves the good results of SASEA on the mentioned easy problems and two another ones, M-Sat(100, 1200, 3) and Maxcut(pm1s.80.6). Moreover, SAGACIA outperforms SASEA on five problems: Overlapping deceptive problem, M-Sat(100, 2400, 3), and Maxcut problems w09-100.2, g05-100.5, and pw05-100.6. These results are in accordance with the ones in Table 4 and SAGACIA appears as the most competitive algorithm with regard to SASEA among these hybrid MHs. In particular, SAGACIA obtains better results than SASEA on three out of five Maxcut problems, and it is surpassed on just one of them.

Fig. 10 shows convergence graphs for the studied algorithms, obtained by the same procedure described in Section 5.2. We see that most of the algorithms early get trapped in local optima or suffer premature convergence to poor solutions. Then, they have difficulties to improve further. However, the search process of SASEA is quietly driven toward promising solutions, overcoming local optima and reaching better final results than the other methods.

It is also interesting to notice that the graph of AGA surpasses the one of SAGACIA, though its results were poorer according to the statistical tests applied (Tables 4 and 8–10). The reason is that there are several problems for which AGA gets better results than SAGACIA, but not enough to outperform SASEA. On the contrary, SAGACIA really outperforms SASEA on several problems, but obtains results poorer than AGA on another ones (compare the results of AGA and SAGACIA on Royal road, HIFF problems, and Maxcut(ising2.5-250.5555), for instance).

7. SASEA vs. state-of-the-art algorithms for binary combinatorial problems

In this section, we intend to assess the performance of SASEA with regard to relevant optimisers for binary-coded problems found in the literature:

- *Cross-generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation* (CHC) [43]. Though this algorithm was already considered in Section 5.3, it is included here for the sake of completeness. CHC involves the combination of a selection strategy with a very high selective pressure and several components inducing diversity. CHC was tested against different Genetic Algorithms, giving better results especially on hard problems [69]. So, it has arisen as a reference point in the literature. Its population consists of 50 individuals.
- *Variable Dissortative Mating Genetic Algorithm* (VDMGA) [41]. It is a recent steady-state EA, similar to CHC, in which the number of new chromosomes entering the population in each generation is controlled on-line by a threshold value, genetic diversity, and population's state of convergence. The results in [41] displayed the superior performance of VDMGA when compared to other Genetic Algorithms in which CHC was included. Its population size parameter is set to 100. In addition, the mutation probability is set to 0.006.
- *Sawtooth Genetic Algorithm* (Saw-GA) [70]. It is a Genetic Algorithm that uses a variable population size and periodic partial reinitialisation of the population in the form of a saw-tooth function. For a wide range of problems, the performance of Saw-GA was statistically superior to a standard and a micro Genetic Algorithm. The average population size is set to 80; crossover and mutation probabilities are set to 0.85 and 0.05, respectively; period and amplitude parameters are adjusted to 40 and 75, respectively.
- *Context-Independent Scatter Search* (CISS) [71]. CISS is a proposal explicitly designed to tackle general binary-coded problems. Its performance was compared with the one of several general-purpose commercial optimisation tools, obtaining promising results. Population and reference set sizes have been set to 300 and 6, respectively. The other parameters have been set as in [71].
- *Versatile Quantum-inspired EA* (vQEA) [72]. vQEA is a recent EA approach based on quantum computing principles. It considers the *quantum bit* (Qbit) as the smallest information unit, which is defined by the probability at which the corresponding state (0 or 1) is likely to appear when it is collapsed, i.e., read or measured. vQEA considers a population of quantum individuals (a quantum individual is composed of a Qbit string, a realization, and an attractor) that evolve through quantum gate operations. The population of vQEA is divided into g groups each containing d individuals. Attractors are periodically synchronized between individuals in the same group and between different groups. vQEA considers a population of 10 individuals distributed into an unique group. Attractors synchronization takes place every generation and $\Delta\theta$, associated to quantum gate, is $\pi/100$.

Tables 8–10, in Appendix C, show the results of the search algorithms when tackling each test problem. Table 5 summarises the results of applying the Wilcoxon matched-pairs signed ranks test for p -value = 0.05. We notice that SASEA obtains results statistically better than the ones of the other optimisers. In addition, differences between R^+ and R^- values are relevant. Therefore, SASEA arises as a promising tool to tackle binary-coded problems.

Taking a deeper look into Tables 8–10, we see that:

- Saw-GA is outperformed by SASEA on almost all the functions. The statistical test does not find significant differences between

Table 5

SASEA vs. state-of-the-art optimisers (Wilcoxon's test with p -value = 0.05 and critical value = 107).

Algorithms	R^+	R^-	Sig. differences?
SASEA vs. CHC	339.5	38.5	+
SASEA vs. VDMGA	325	53	+
SASEA vs. CISS	298.5	79.5	+
SASEA vs. Saw-GA	371.5	6.5	+
SASEA vs. vQEA	363.5	14.5	+

their results on just 5 problems, which are Trap, Deceptive, NkLand(48, 4), and two Multiple knapsack problems, weish03 and pb4. An interesting fact is that the number of variables of those problems is significantly low with regard to the ones of the other problems.

- SASEA outperforms CHC on 17 out of the 27 problem instances considered. In particular, SASEA generally shows superiority on the M-sat, Maxcut, BQP, and Knapsack real-world problems. On the other hand, CHC obtains statistically better results than SASEA on two problems, Bipolar deceptive and PPeaks(10,100), and it proves to be competitive with regard to SASEA on several another ones: Trap, Nkland problems, HIFF(2, 5, true), PPeaks(100,100), BQP(bqp50-1), Maxcut(ising2.5-250.5555), and Multiple knapsack problem pb4. According to these results, we did not find a specific class of problems where CHC appeared superior to SASEA.
- VDMGA attains similar results to those of SASEA on many problems, however, it only achieves better results on just one problem, Royal road. On the other hand, SASEA is superior to VDMGA on M-sat, Maxcut, and BQP real-world problems.
- We see that SASEA achieves better results than CISS on several artificial problems such as HIFF, Royal road, Trap, Deceptive and Bipolar deceptive, and on real-world ones such as BQP and Maxcut, too. On the other hand, CISS shows superiority on the artificial Nkland problems, as well as on the Overlapping deceptive one. In our opinion, its particular mechanisms to protect and enhance diversification are the reasons for its good results on those commonly said complex problems. Both algorithms obtain similar results on all the PPeaks and Multiple knapsack problems.
- Finally, SASEA outperforms vQEA almost on every function. vQEA just obtains similar results to those of SASEA on three PPeaks problems.

Fig. 11 shows convergence graphs for the studied optimisers. They were obtained by the same procedure described in Section 5.2. The behaviour of the search algorithms are very similar to those of the hybrid MHs in Section 6. Most of the methods seem to

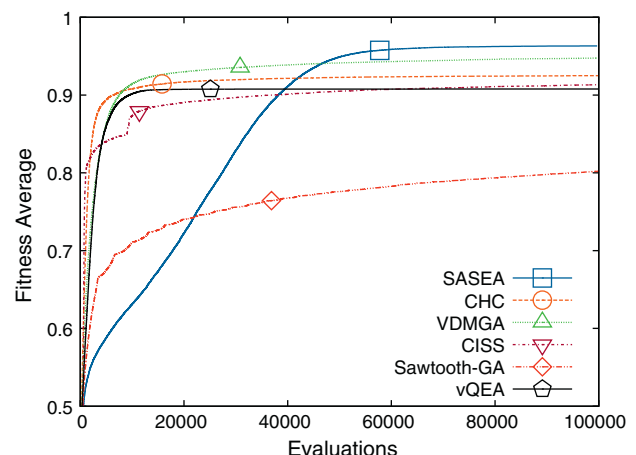


Fig. 11. Convergence graphs of state-of-the-art optimisers and SASEA.

get trapped in local optima early or experience slow convergence toward good solutions. However, though SASEA reaches solutions of equal quality later, its ability to protect diversification at the initial stages of the process and gradually increase intensification, by means of the proposed mating mechanism, crossover operator, and replacement strategy, lets SASEA to overcome local optima and reach better results.

8. Conclusion

In this work, we proposed SASEA, an innovative hybrid MH based on a specialised EA that executes a search process equivalent to the one of SA. We have performed several experimental studies concluding that:

- The neighbourhood structure associated to the components of SASEA, which dynamically adapts its size and can be used to promote intensification and/or diversification, becomes beneficial when compared to the classic neighbourhood structure (one-flip) of SA and another one presented in the literature that dynamically adapts its size.
- The alternative scheme for combining ideas from SA and EAs introduced by SASEA may outperform other hybrid MHs based on SA and EAs.
- SASEA provides a significantly better performance than other state-of-the-art algorithms for binary-coded problems, on the test suite considered.

The research line focused in this paper is indeed worth of further studies. We are currently extending our investigation to build evolutionary models incorporating SASEA to perform a search process on a set of points in parallel, and to take benefits from parallel hardware. In addition, we intend to extend our investigation to different test-suits, other coding schemes and other real-world problems, such as *training set selection in data mining* [73].

Acknowledgements

This work was supported by Research Projects TIN2008-05854 and P08-TIC-4173.

Appendix A. Test suite

The test suite that we have used for the experiments consists of 27 binary-coded test problem instances from 12 different problem classes, 8 classes (13 instances) from the artificial intelligence field and 4 (14 instances) from real-world applications. They are described in the following sections.

A.1. Royal road

To construct a *royal road* problem [74], we select a random optimum string and break it up into a number of small building blocks. We then assign values to each low-order schema and use those values to compute the fitness of a bit string X in terms of the schemes of which it is an instance. The value of each low-order schema is equal to its size. We have used a royal road problem with length 400 where low-order schemes consist of groups of 8 consecutive bits.

A.2. Trap problem

Trap problem [36] consists of misleading subfunctions of different lengths. Specifically, the fitness function $f(x)$ is constructed by

Table 6
Fitness values of the subfunctions F_i .

	0	1	2	3
F_3	4	2	0	10
F_2	5	0	10	
F_1	0	10		

Table 7
Deceptive order-3 problem.

Chromosomes	000	001	010	100	110	011	101	111
Fitness	28	26	22	14	0	0	0	30

adding subfunctions of length 1 (F_1), 2 (F_2), and 3 (F_3). Each subfunction has two optima: the optimal fitness value is obtained for an all-ones string, while the all-zeroes string represents a local optimum. The fitness of all other string in the subfunction is determined by the number of zeroes: the more zeroes, the higher the fitness value. This causes a large basin of attraction toward the local optimum. The fitness values for the subfunctions are specified in Table 6 where the columns indicate the number of ones in the subfunctions F_1 , F_2 , and F_3 . The fitness function $f(x)$ is composed of 4 F_3 subfunctions, 6 F_2 subfunctions, and 12 F_1 subfunctions. The overall length of the problem is thus 36. There are 2^{10} optima of which only one is the global optimum: the string with all ones having a fitness value of 220.

$$f(x) = \sum_{i=0}^3 F_3(x_{[3i:3i+2]}) + \sum_{i=0}^5 F_2(x_{[2i+12:2i+13]}) + \sum_{i=0}^{11} F_1(x_{24+i}) \quad (\text{A.1})$$

A.3. Deceptive problems

In *deceptive problems* [75], there are certain schemata that guide the search toward some solution that is not globally competitive. The schemata that have the global optimum do not bear significance and so they may not proliferate during the genetic process. The used deceptive problem consists of the concatenation of k subproblems of length 3. The fitness for each 3-bit section of the string is given in Table 7. The overall fitness is the sum of the fitness of these deceptive subproblems. We have used a deceptive problem with 13 subproblems.

A.4. Bipolar deceptive problem

A deceptive function of order 3 is defined as

$$f_{deceptive}^3 = \begin{cases} 0.9 & \text{if } u = 0 \\ 0.8 & \text{if } u = 1 \\ 0 & \text{if } u = 2 \\ 1 & \text{otherwise} \end{cases} \quad (\text{A.2})$$

where u is the sum of three input bits.

A *bipolar deceptive function* of order 6 [76] is defined with the use of deceptive subproblems of length 3 as follows:

$$f_{bipolar}^6(X) = f_{deceptive}^3(|3 - u|), \quad (\text{A.3})$$

where X is a vector of 6 binary variables, and u is the sum of the input bits. We have included a bipolar problem with 66 $f_{bipolar}^6$ subproblems. The overall fitness is the average of the fitness of these subproblems.

A.5. Overlap deceptive problem

A deceptive function composed of deceptive functions of order 3 (previous section) that are overlapping in one bit in a chain-like

structure will be referred to as *3-deceptive with overlapping* [76]. It is defined as follows:

$$f_{3dec-overlap}(X) = \sum_{i=0}^{(n-3)/2} f_{deceptive}^3(S_i), \quad (A.4)$$

where $X = (X_0, \dots, X_{n-1})$ is a vector of bits, and S_i is the concatenation of bits X_{2i}, X_{2i+1} , and X_{2i+2} . The overlap deceptive problem included in the test suite consists of 199 $f_{3dec-overlap}$ subfunctions, making the length of the problem to be equal to 399. The overall fitness is the average of the fitness of the subfunctions.

A.6. NK-Landscapes

In the *NK model* [77], N represents the number of genes in a haploid chromosome and K represents the number of linkages each gene has to other genes in the same chromosome. To compute the fitness of the entire chromosome, the fitness contribution from each locus is averaged as follows:

$$f(s) = \frac{1}{N} \sum_{i=1}^N f(\text{locus}_i) \quad (A.5)$$

where the fitness contribution of each locus, $f(\text{locus}_i)$, is determined by using the (binary) value of gene i together with values of the K interacting genes as an index into a table T_i of size 2^{K+1} of randomly generated numbers uniformly distributed over the interval $[0, 1]$. For a given gene i , the set of K linked genes may be randomly selected or consist of the immediately adjacent genes.

We have used two set of instances of the NK-Landscape problem: one with $N=48$ and $K=4$, and another with $N=48$ and $K=12$. They are denoted as $NKLand(N, K)$. They have been obtained from [78]. Each run i , of every search algorithm, uses a different seed ($seed_i$) for generating the $NKLand(N, K)$ instance, i.e. the i th execution of every method has used the same $seed_i$, whereas the j th execution has used $seed_j$.

A.7. Hierarchical if-and-only-if problems

Hierarchical if-and-only-if problems (HIFF) [79] are defined as follows: A binary string $X = (x_1, \dots, x_n)$, where $n = k^p$, represents a hierarchical block structure, where k is the number of sub-blocks in a block, and p is the number of levels in the hierarchy. Blocks do not need to consist of consecutive sub-blocks, i.e. they can be shuffled. Each block at the bottom level of this hierarchy, consisting of k bits each, will be converted into a single symbol by a transform function, t . This transform function defines the meaning of each block. This creates a new string (with length $k^{(p-1)}$) that is the decoding of the block structure to the first level. This process is repeated for each level in the hierarchy to give the single symbol that is the meaning of the whole structure. Thus, the recursive transform function, T , transforms any block structure to its meaning, a single symbol:

$$T(X) = \begin{cases} x_i & \text{if } |X| = 1, \\ t(T(X^1), \dots, T(X^k)) & \text{otherwise} \end{cases} \quad (A.6)$$

where t is a base function that defines the resultant symbol from a block of k symbols and X^i is the i th sub-block of X , i.e. $\{x_{(i-1)d+1}, \dots, x_{id}\}$ (where $d = |B|/k$).

Now we may use $T(X)$ to construct $f(X)$, the fitness of a block structure. Specifically, the fitness of a block structure will be the fitness contribution of its transform (scaled for its size) plus the

sum of the fitness of its sub-blocks. Hence the recursive function, f , defined using the base function F :

$$f(X) = \begin{cases} F(X) & \text{if } |X| = 1, \\ |X|F(T(B)) + \sum_{i=1}^k f(X^i) & \text{otherwise} \end{cases} \quad (A.7)$$

where F is a base function giving the fitness of a single bit.

We now give the base functions, F and t , that provide an interesting fitness landscape. First we define $t(\{A, B\})$ by arbitrarily assigning 0 and 1 to the two solutions of IFF (i.e. $t(\{0, 0\}) = 0$, $t(\{1, 1\}) = 1$, and null for any other combination of 0, 1, and null). $F(A)$ naturally defines the two non-null transform values as desirable and null as undesirable.

We have used two instances of this problem. The first one has 5 levels and 2 shuffled subblocks per block, and the second one, 4 levels and 3 consecutive subblocks per block.

A.8. P-Peak problems

P-Peak problem generator [80] creates instances with a certain number of peaks (the degree of multi-modality). For a problem with P peaks, P bit strings of length L are randomly generated. Each of these string is a peak (a local optima) in the landscape. Different heights can be assigned to different peaks based on various schemes (equal height, linear, logarithm-based, and so on). To evaluate an arbitrary solution S , first locate the nearest peak in Hamming space, call it $Peak_n(S)$. Then, the fitness of s is the number of bits the string has in common with $Peak_n(S)$, divided by L , and scaled by the height of the nearest peak. In case there is a tie when finding the nearest peak, the highest peak is chosen.

We have used different groups of P-Peak instances denoted as $PPeaks(P, L)$. Each run i , of every search algorithm, uses a different seed ($seed_i$) for generating the $PPeaks(P, L)$ instance. Linear scheme have been used for assigning heights to peaks in $[0.6, 1]$.

A.9. Max-Sat problem

The satisfiability problem in propositional logic (SAT) [81] is the task to decide whether a given propositional formula has a model. More formally, given a set of m clauses $\{C_1, \dots, C_m\}$ involving n boolean variables X_1, \dots, X_n the SAT problem is to decide whether an assignment of values to variables exists such that all clauses are simultaneously satisfied.

Max-Sat is the optimisation variant of SAT and can be seen as a generalisation of the SAT problem: Given a propositional formula in conjunctive normal form (CNF), the Max-Sat problem is to find a variable assignment that maximises the number of satisfied clauses. It returns the percentage of satisfied clauses.

We have used two set of instances of the Max-Sat problem with 100 variables, 3 variables by clause, and 1200 and 2400 clauses, respectively. They have been obtained from [78]. They are denoted as $M-Sat(n, m, l)$, where l indicates the number of variables involved in each clause (3). Each run i , of every search algorithm, uses a specific seed ($seed_i$) for generating the $M-Sat(n, m, l)$ instance, i.e. i th execution of every method uses the same $seed_i$, whereas j th execution uses $seed_j$.

A.10. Unconstrained Binary Quadratic Programming problem

The objective of the *Unconstrained Binary Quadratic Programming* (BQP) [82] is to find, given a symmetric rational $n \times n$ matrix

$Q=(Q_{ij})$, a binary vector of length n that maximises the following quantity:

$$f(x) = x^t Q x = \sum_{i=1}^n \sum_{j=1}^n q_{ij} x_i x_j, \quad x_i \in \{0, 1\} \quad (\text{A.8})$$

We have used 4 instances with different values for n . They can be obtained from the corresponding files from the *Biq Mac Library*.³

A.11. Max-Cut problem

The *Max-Cut problem* [83] is defined as follows: Let an undirected and connected graph $G=(V, E)$, where $V=\{1, 2, \dots, n\}$ and $E \subset \{(i, j) : 1 \leq i < j \leq n\}$, be given. Let the edge weights $w_{ij} = w_{ji}$ be given such that $w_{ij} = 0 \forall (i, j) \notin E$, and in particular, let $w_{ii} = 0$. The Max-Cut problem is to find a bipartition (V_1, V_2) of V so that the sum of the weights of the edges between V_1 and V_2 is maximised.

We have used 5 instances of the Max-Cut problem that can be obtained from the corresponding files from the *Biq Mac Library*.³

A.12. Unconstrained knapsack problem

This problem is a generalisation of the *simple knapsack problem*. We are given m knapsacks with capacities c_1, c_2, \dots, c_m and n objects each with a profit p_i . Every object i has a weight w_{ij} when it is included in the knapsack j . Note that contrary to the simple knapsack problem the weights of the objects are not constant but instead they depend on the knapsack they have been allocated to. Objects are either allocated to all knapsacks or they are not allocated at all. The goal is to find an allocation of the objects to the knapsacks such that the total profit is maximised. Calling $X=(x_1, \dots, x_n)$ the allocation binary string, the goal can be formalised as maximising $\sum_{i=1}^n x_i \cdot p_i$ constrained by the knapsack capacities, or $\sum_{i=1}^n w_{ij} \cdot x_i \leq c_j$. Whenever at least one knapsack is overfilled the string represents an infeasible solution. We include a penalty term in the fitness function that becomes larger the farther the solution is from feasibility. The fitness function to be maximised is:

$$f(x) = \sum_{i=1}^n p_i \cdot x_i - s \cdot \max(p_i) \quad (\text{A.9})$$

where s is the number of overfilled knapsacks.

We have used 3 instances of this problem that can be obtained from the corresponding files from the *SAC-94 Suite*.⁴

Appendix B. Wilcoxon matched-pairs signed-ranks test

Wilcoxon's test [53] is used for answering this question: *do two samples represent two different populations?* It is a non-parametric procedure employed in a hypothesis testing situation involving a

design with two samples. It is a pairwise test that aims to detect significant differences between the behaviour of two algorithms.

The null hypothesis for Wilcoxon's test is $H_0 : \theta_D = 0$; in the underlying populations represented by the two samples of results, the average of the difference scores equals zero. The alternative hypothesis is $H_1 : \theta_D \neq 0$, but also can be used $H_1 : \theta_D > 0$ or $H_1 : \theta_D < 0$ as directional hypothesis.

In the following, we describe the tests computations. Let d_i be the difference between the performance scores of the two search algorithms on i th out of N functions. The differences are ranked according to their absolute values (we have previously normalised the results of every algorithm on each test problem into the interval $[0, 1]$, taking into consideration highest and lowest fitness values achieved on each test problem); average ranks are assigned in case of ties. Let R^+ be the sum of ranks for the functions on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i > 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i) \quad \text{and} \quad R^- = \sum_{d_i < 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i) \quad (\text{B.1})$$

Let T be the smallest of the sums, $T = \min(R^+, R^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for N degrees of freedom (Table B.12 in [84]), the null hypothesis of equality of means is rejected.

The obtaining of the p -value associated to a comparison is performed by means of the normal approximation for the Wilcoxon T statistic (Section VI, Test 18 in [85]). Furthermore, the computation of the p -value for this test is usually included in well-known statistical software packages (SPSS, SAS, R, etc.).

Appendix C. Results

Tables 8–10 show the average of fitness values obtained by the studied search algorithms. Best results for each problem are boldfaced. Moreover, we have carried out a statistical analysis with p -value equal to 0.05 to measure the performance differences between SASEA and the other algorithms on each problem separately. This statistical analysis is based on *Mann–Whitney test* [68]. A plus sign (+) indicates that the performance of SASEA is superior to the corresponding algorithm on that problem. A minus sign (–) means that the performance of SASEA is inferior to the corresponding algorithm on that problem. And an approximate sign (~) is written when the tests did not find significant differences between the performances of SASEA and the corresponding algorithm on the problem.

³ <http://biqmac.uni-klu.ac.at/>.

⁴ <http://elib.zib.de/pub/Packages/mp-testdata/ip/sac94-suite/>.

Table 8
Results of the search algorithms on test problems 1–9.

Alg	1	2	3	4	5	6	7	8	9
Classic SA	0.988 +	218 ~	379 +	0.894 +	0.899 –	0.763 ~	0.741 ~	161 ~	175 ~
Dyn. SA	0.181 +	215 +	379 +	0.857 +	0.760 +	0.745 +	0.717 +	146 +	152 +
PRSA	0.366 +	209 +	375 +	0.860 +	0.768 +	0.727 +	0.713 +	151 +	159 +
AGA	0.690 +	216 +	370 +	0.893 +	0.894 +	0.759 ~	0.714 +	179 –	171 +
SAGACIA	0.193 +	212 +	379 +	0.892 +	0.903 –	0.758 ~	0.734 +	154 +	155 +
SSSA	0.123 +	214 +	373 +	0.875 +	0.845 +	0.741 +	0.718 +	160 ~	156 +
CSA	0.065 +	179 +	365 +	0.813 +	0.635 +	0.674 +	0.692 +	116 +	120 +
CHC	0.337 +	220 ~	378 +	0.878 +	0.899 –	0.760 ~	0.746 ~	164 ~	170 +
VDMGA	1 –	220 ~	382 +	0.850 +	0.896 ~	0.765 ~	0.701 +	162 ~	176 ~
Saw-GA	0.192 +	219 ~	385 ~	0.853 +	0.753 +	0.755 ~	0.713 +	155 +	155 +
CISS	0.130 +	210 +	378 +	0.892 +	0.915 –	0.761 ~	0.751 –	148 +	152 +
vQEA	0.536 +	219 +	366 +	0.886 +	0.894 +	0.747 +	0.710 +	137 +	154 +
SASEA	0.997	220	384	0.899	0.898	0.767	0.743	165	175

Table 9
Results of the search algorithms on test problems 10–18.

Alg	10	11	12	13	14	15	16	17	18
Classic SA	0.985 +	0.984 ~	0.987 +	0.985 +	0.958 ~	0.939 ~	5176 ~	121,270 ~	11,598 ~
Dyn. SA	0.953 +	0.942 +	0.885 +	0.847 +	0.944 +	0.926 +	5139 +	55,573 +	9618 +
PRSA	0.841 +	0.828 +	0.805 +	0.747 +	0.933 +	0.921 +	4641 +	59,834 +	8140 +
AGA	0.988 +	0.976 +	0.967 +	0.980 +	0.953 +	0.936 +	5176 ~	117,034 +	11,597 +
SAGACIA	0.989 +	0.978 +	0.984 +	0.988 +	0.958 ~	0.939 –	5176 ~	119,805 +	11,598 +
SSSA	0.939 +	0.935 +	0.937 +	0.923 +	0.952 +	0.932 +	5152 +	80,116 +	10,792 +
CSA	0.680 +	0.709 +	0.656 +	0.627 +	0.912 +	0.903 +	3378 +	16,728 +	3958 +
CHC	0.994 –	0.990 ~	0.990 +	0.990 +	0.957 +	0.937 +	5174 ~	117,453 +	11,548 +
VDMGA	1 ~	0.990 ~	0.992 +	0.995 ~	0.957 +	0.938 +	5176 ~	120,948 +	11,589 +
Saw-GA	0.941 +	0.929 +	0.870 +	0.831 +	0.943 +	0.926 +	5159 +	51,714 +	9420 +
CISS	0.997 ~	0.991 ~	0.995 ~	0.993 ~	0.955 +	0.937 +	5176 ~	119,056 +	11,583 +
vQEA	0.999 ~	0.988 ~	0.996 ~	0.993 +	0.954 +	0.936 +	5167 +	117,401 +	11,529 +
SASEA	0.993	0.987	0.997	0.996	0.958	0.938	5176	121,265	11,598

Table 10
Results of the search algorithms on test problems 19–27.

Alg	19	20	21	22	23	24	25	26	27
Classic SA	50,447 ~	71 ~	2707 ~	1434 ~	8196 ~	7,585,236 –	4039 +	12,335 +	118,204 ~
Dyn. SA	36,219 +	62 +	2324 +	1388 +	7957 +	5,136,800 +	4035 +	12,380 +	118,204 ~
PRSA	34,193 +	51 +	2122 +	1366 +	7819 +	5,153,437 +	3915 +	12,189 +	116,566 +
AGA	50,311 +	70 +	2675 +	1422 +	8125 +	7,369,494 +	3965 +	11,994 +	118,204 ~
SAGACIA	50,388 +	72 ~	2720 –	1435 –	8209 –	7,222,749 +	3960 +	12,279 +	118,204 ~
SSSA	44,338 +	68 +	2529 +	1412 +	8074 +	6,546,047 +	3975 +	12,244 +	118,204 ~
CSA	14,305 +	31 +	1510 +	1319 +	7463 +	2,812,404 +	3720 +	12,090 +	109,247 +
CHC	49,852 +	68 +	2623 +	1422 +	8153 +	7,526,181 ~	4005 +	12,396 +	118,204 ~
VDMGA	50,428 +	69 +	2647 +	1423 +	8168 +	7,522,897 +	4003 +	12,398 ~	118,204 ~
Saw-GA	34,427 +	61 +	2267 +	1385 +	7923 +	4,949,495 +	4074 ~	12,385 +	118,204 ~
CISS	50,407 +	68 +	2681 +	1430 +	8190 ~	7,333,148 +	4096 ~	12,400 ~	118,204 ~
vQEA	50,161 +	65 +	2551 +	1406 +	8102 +	7,249,980 +	3954 +	12,392 +	118,204 ~
SASEA	50,444	72	2694	1434	8193	7,558,447	4081	12,399	118,204

References

- [1] F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, 2003.
- [2] P. Siarry, Z. Michalewicz (Eds.), Advances in Metaheuristics for Hard Optimization. Natural Computing, Springer, 2008.
- [3] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (2003) 268–308.
- [4] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (6) (2011) 4135–4151.
- [5] G. Raidl, A unified view on hybrid metaheuristics, in: F. Almeida, M.B. Aguilera, C. Blum, J.M. Vega, M.P. Pérez, A. Roli, M. Sampels (Eds.), Hybrid Metaheuristics, Springer, 2006, pp. 1–12.
- [6] E. Talbi, A taxonomy of hybrid metaheuristics, J. Heuristics 8 (2002) 541–564.
- [7] S. Kirkpatrick, C. Gelatt Jr., M. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
- [8] E. Aarts, J. Korst, Simulated Annealing and Boltzmann Machines, John Wiley & Sons, 1989.
- [9] H. Cheng, X. Wang, S. Yang, M. Huang, A multipopulation parallel genetic simulated annealing-based QoS routing and wavelength assignment integration algorithm for multicast in optical networks, Appl. Soft Comput. 9 (2009) 677–684.
- [10] D. Henderson, S.H. Jacobson, A.W. Jacobson, The theory and practice of simulated annealing, Handbook of Metaheuristics 57 (2003) 287–320.
- [11] N. Li, J. Cha, Y. Lu, A parallel simulated annealing algorithm based on functional feature tree modeling for 3D engineering layout design, Appl. Soft Comput. 10 (2010) 592–601.
- [12] C. Queirolo, L. Silva, O. Bellon, M. Segundo, 3d face recognition using simulated annealing and the surface interpenetration measure, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 206–219.
- [13] P. Salamon, P. Sibani, R. Frost, Facts, conjectures and improvements for simulated annealing, Monographs on Mathematical Modeling and Computation, SIAM, 2002.
- [14] R. Venkata Rao, P. Pawar, Parameter optimization of a multi-pass milling process using non-traditional optimization algorithms, Appl. Soft Comput. 10 (2010) 445–456.
- [15] T. Bäck, D. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, Institute of Physics Publishers, 1997.

- [16] A. Eiben, J. Smith, *Introduction to Evolutionary Computing*, Springer-Verlag, 2003.
- [17] M. Lozano, C. García-Martínez, Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report, *Comput. Oper. Res.* 37 (2010) 481–497.
- [18] C. García-Martínez, M. Lozano, Local search based on genetic algorithms, in: P. Siarry, Z. Michalewicz (Eds.), *Advances in Metaheuristics for Hard Optimization*, Natural Computing, Springer, 2008, pp. 199–221.
- [19] C. García-Martínez, M. Lozano, Evaluating a local genetic algorithm as context-independent local search operator for metaheuristics, *Soft Comput.* 14 (2010) 1117–1139.
- [20] M. Lozano, F. Herrera, N. Krasnogor, D. Molina, Real-coded memetic algorithms with crossover hill-climbing, *Evol. Comput.* 12 (2004) 273–302.
- [21] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, *IEEE Trans. Evol. Comput.* 12 (2008) 107–125.
- [22] S. Mahfoud, D. Goldberg, Parallel recombining simulated annealing: a genetic algorithm, *Parallel Comput.* 21 (1995) 1–28.
- [23] B. Li, W. Jiang, A novel stochastic optimization algorithm, *IEEE Trans. Syst. Man Cybern. B* (2000) 193–198.
- [24] F.T. Lin, C.Y. Kao, C.C. Hsu, Applying the genetic approach to simulated annealing in solving some np-hard problems, *IEEE Trans. Syst. Man Cybern.* 23 (1993) 1752–1767.
- [25] D. Thompson, G. Bilbro, Sample-sort simulated annealing, *IEEE Trans. Syst. Man, Cybern. B* 35 (2005) 625–632.
- [26] S. Xavier-de-Souza, J. Suykens, J. Vandewalle, D. Bollé, Cooperative behavior in coupled simulated annealing processes with variance control, in: *Symposium on Nonlinear Theory and Its Applications*, 2006.
- [27] G. Sywerda, Uniform crossover in genetic algorithms, in: J. Schaffer (Ed.), *Proc. of the Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 1989, pp. 2–9.
- [28] D. Whitley, The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best, in: J. Schaffer (Ed.), *Proc. of the Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 1989, pp. 116–121.
- [29] C. García-Martínez, M. Lozano, Simulated annealing based on local genetic search, in: *Proc. of the IEEE Int. Conf. Evolutionary Computation*, 2009, pp. 2569–2576.
- [30] E. Aarts, J. Korst, Selected topics in simulated annealing, in: *Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers Group, 2002, pp. 1–37.
- [31] P. Van Laarhoven, E. Aarts, *Simulated Annealing: Theory and Applications*, Kluwer Academic Publishers, Norwell, 1987.
- [32] P. Strenski, S. Kirkpatrick, Analysis of finite length annealing schedules, *Algoritmica* 6 (1991) 346–366.
- [33] K. Deb, A population-based algorithm-generator for real-parameter optimization, *Soft Comput.* 9 (2005) 236–253.
- [34] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, A. Sánchez, Global and local real-coded genetic algorithms based on parent-centric crossover operators, *Eur. J. Oper. Res.* 185 (2008) 1088–1113.
- [35] M. Raghuvanshi, O. Kakde, Probability distribution based recombination operator to solve unimodal and multimodal problems, *Int. J. Knowl. Intell. Eng. Syst.* 10 (2006) 247–255.
- [36] D. Thierens, Population-based iterated local search: restricting neighborhood search by crossover, in: K. Deb, R. Poli, W. Banzhaf, H.G. Beyer, E. Burk, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A. Tyrrel (Eds.), *Proc. of the Genetic and Evolutionary Computation Conf.*, Springer, 2004, pp. 234–245.
- [37] T. Jones, Crossover, macromutation, and population-based search, in: L. Eshelman (Ed.), *Proc. of the Sixth Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 1995, pp. 73–80.
- [38] U. O'Reilly, F. Oppacher, Hybridized crossover-based search techniques for program discovery, in: *Proc. of the World Conf. on Evolutionary Computation*, 1995, pp. 573–578.
- [39] C. García-Martínez, M. Lozano, D. Molina, A local genetic algorithm for binary-coded problems, in: T. Runarsson, H.G. Beyer, E. Burke, J. Merelo-Guervós, L. Whitley, X. Yao (Eds.), *Proc. of the Int. Conf. on Parallel Problem Solving from Nature*, Springer, 2006, pp. 192–201.
- [40] C. Fernandes, A. Rosa, A study on non-random mating and varying population size in genetic algorithms using a royal road function, in: *Proc. of the Congress on Evolutionary Computation*, IEEE Press, 2001, pp. 60–66.
- [41] C. Fernandes, A. Rosa, Self-adjusting the intensity of assortative mating in genetic algorithms, *Soft Comput.* 12 (2008) 955–979.
- [42] G. Eiben, M. Schut, New ways to calibrate evolutionary algorithms, in: P. Siarry, Z. Michalewicz (Eds.), *Advances in Metaheuristics for Hard Optimization*, Springer-Verlag, 2008, pp. 153–177.
- [43] L. Eshelman, J. Schaffer, Preventing premature convergence in genetic algorithms by preventing incest, in: R. Belew, L. Booker (Eds.), *Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 1991, pp. 115–122.
- [44] S. Mahfoud, Crowding and preselection revised, in: R. Männer, B. Manderick (Eds.), *Parallel Problem Solving from Nature*, Elsevier Science Publishers, 1992, pp. 27–36.
- [45] S.W. Mahfoud, *Niching methods for genetic algorithms*, Ph.D. Thesis, University of Illinois, Champaign, IL, USA, 1995.
- [46] G. Harik, Finding multimodal solutions using restricted tournament selection, in: L. Eshelman (Ed.), *Proc. of the Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 1995, pp. 24–31.
- [47] D. Wolpert, W. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.* 1 (1997) 67–82.
- [48] M. Birattari, P. Balaprakash, T. Stützle, M. Dorigo, Estimation-based local search for stochastic combinatorial optimization using delta evaluations: a case study on the probabilistic traveling salesman problem, *INFORMS J. Comput.* 20 (2008) 644–658.
- [49] P. Merz, On the performance of memetic algorithms in combinatorial optimization, in: *Second Workshop on Memetic Algorithms*, Genetic and Evolutionary Computation Conference, Morgan Kaufmann, 2001, pp. 168–173.
- [50] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm Evol. Comput.* 1 (2011) 3–18.
- [51] S. Garcia, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, *Soft Comput.* 13 (2009) 959–977.
- [52] S. Garcia, D. Molina, M. Lozano, 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, *J. Heuristics* 15 (2009) 617–644.
- [53] F. Wilcoxon, Individual comparisons by ranking methods, *Biometrics* 1 (1945) 80–83.
- [54] J.A.R. Marshall, T.G. Hinton, Beyond no free lunch: realistic algorithms for arbitrary problem classes, in: *IEEE Congress on Evolutionary Computation*, 2010, pp. 18–23.
- [55] C. Schumacher, M. Vose, L. Whitley, The no free lunch theorem and description length, in: *Genetic and Evolutionary Computation Conference (GECCO)*, 2001, pp. 565–570.
- [56] C. Igel, M. Toussaint, On classes of functions for which no free lunch results hold, *Information processing letters* 86 (2003) 317–321.
- [57] E. Alba, B. Dorronsoro, The exploration/exploitation tradeoff in dynamic cellular algorithms, *IEEE Trans. Evol. Comput.* 9 (2005) 126–142.
- [58] G. Chen, C. Low, Z. Yang, Preserving and exploiting genetic diversity in evolutionary programming algorithms, *IEEE Trans. Evol. Comput.* 13 (2009) 661–673.
- [59] S. He, Q. Wu, J. Saunders, Group search optimizer: an optimization algorithm inspired by animal searching behavior, *IEEE Trans. Evol. Comput.* 13 (2009) 973–990.
- [60] D. Molina, M. Lozano, C. García-Martínez, F. Herrera, Memetic algorithms for continuous optimization based on local search chains, *Evol. Comput.* 18 (2010) 27–63.
- [61] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization, Technical Report, Nanyang Technological University, 2005.
- [62] J. Vrugt, B. Robinson, J. Hyman, Self-adaptive multimethod search for global optimization in real-parameter spaces, *IEEE Trans. Evol. Comput.* 13 (2009) 243–259.
- [63] S. Yuen, C. Chow, A genetic algorithm that adaptively mutates and never revisits, *IEEE Trans. Evol. Comput.* 13 (2009) 454–472.
- [64] X. Yao, Simulated annealing with extended neighbourhood, *Int. J. Comput. Math.* 40 (1991) 169–189.
- [65] J. Liu, The impact of neighbourhood size on the process of simulated annealing: computational experiments on the flowshop scheduling problem, *Comput. Ind. Eng.* 37 (1999) 285–288.
- [66] B. Fox, Integrating and accelerating tabu search, simulated annealing, and genetic algorithms, *Ann. Oper. Res.* 41 (1993) 47–67.
- [67] M. Ventresca, H. Tizhoosh, Simulated annealing with opposite neighbors, in: *Proc. of the IEEE Symposium on Foundations of Computational Intelligence*, 2007, pp. 186–192.
- [68] H. Mann, D. Whitney, On a test of whether one of two random variables is stochastically larger than the other, *Ann. Math. Stat.* 18 (1947) 50–60.
- [69] D. Whitley, S. Rana, J. Dzuberá, E. Mathias, Evaluating evolutionary algorithms, *Artif. Intell.* 85 (1996) 245–276.
- [70] V. Koumousis, C. Katsaras, A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance, *IEEE Trans. Evol. Comput.* 10 (2006) 19–28.
- [71] F. Gortazar, A. Duarte, M. Laguna, R. Martí, Context-independent scatter search for binary problems, Technical Report, Colorado LEEDS School of Business, University of Colorado at Boulder, 2008.
- [72] M. Platel, S. Schliebs, N. Kasabov, Quantum-inspired evolutionary algorithm, *IEEE Trans. Evol. Comput.* 13 (2009) 1218–1232.
- [73] J. Cano, F. Herrera, M. Lozano, On the combination of evolutionary algorithms and stratified strategies for training set selection in data mining, *Appl. Soft Comput.* 6 (2006) 323–332.
- [74] S. Forrest, M. Mitchell, Relative building block fitness and the building block hypothesis, in: L. Whitley (Ed.), *Foundations of Genetic Algorithms 2.*, Morgan Kaufmann, 1993, pp. 109–126.
- [75] D. Goldberg, B. Korb, K. Deb, Messy genetic algorithms: motivation, analysis, and first results, *Complex Syst.* 3 (1989) 493–530.
- [76] M. Pelikan, D. Goldberg, E. Cantú-Paz, Linkage problem, distribution estimation, and bayesian networks, *Evol. Comput.* 8 (2000) 311–340.
- [77] S. Kauffman, Adaptation on rugged fitness landscapes, *Lec. Sci. Complex* 1 (1989) 527–618.
- [78] K. De Jong, M. Potter, W. Spears, Using problem generators to explore the effects of epistasis, in: T. Bäck (Ed.), *Proc. of the Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 1997, pp. 338–345.

- [79] R. Watson, J. Pollack, Hierarchically consistent test problems for genetic algorithms, in: *Proc. of the Congress on Evolutionary Computation*, 1999, p. 1413.
- [80] W. Spears, *Evolutionary Algorithms: The Role of Mutation and Recombination*, Springer, 2000.
- [81] K. Smith, H. Hoos, T. Stützle, Iterated robust tabu search for MAX-SAT, in: J. Carbonell, J. Siekmann (Eds.), *Proc. of the Canadian Society for Computational Studies of Intelligence Conf.*, Springer, 2003, pp. 129–144.
- [82] J. Beasley, Heuristic algorithms for the unconstrained binary quadratic programming problem, Technical Report, The Management School, Imperial College, 1998.
- [83] R. Karp, Reducibility among combinatorial problems, in: R. Miller, J. Thatcher (Eds.), *Complexity of Computer Computations*, Plenum Press, 1972, pp. 85–103.
- [84] J. Zar, *Biostatistical Analysis*, Prentice Hall, 1999.
- [85] D. Sheskin, *The Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman & Hall/CRC, 2000.

2. Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test

The journal paper associated to this part are:

- F.J. Rodríguez, C. García-Martínez, M. Lozano, Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test. IEEE Transactions on Evolutionary computation. In Press doi: 10.1109/TEVC.2012.2182773.
 - Status: **Accepted**.
 - Impact Factor (JCR 2011): 3.341.
 - Subject Category: Computer Science, Artificial Intelligence. Ranking 9 / 119 (Q1).
 - Subject Category: Computer Science, Theory & Methods. Ranking 3 / 99 (Q1).

Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test

Francisco J. Rodriguez, Carlos García-Martínez, and Manuel Lozano

Abstract—The design of hybrid metaheuristics with ideas taken from the simulated annealing and evolutionary algorithms fields is a fruitful research line. In this paper, we first present an overview of the hybrid metaheuristics based on simulated annealing and evolutionary algorithms presented in the literature and classify them according to two well-known taxonomies of hybrid methods. Secondly, we perform an empirical study comparing the behaviour of a representative set of the hybrid approaches based on evolutionary algorithms and simulated annealing found in the literature. In addition, a study of the synergy relationships provided by these hybrid approaches is presented. Finally, we analyse the behaviour of the best performing hybrid metaheuristic with regards to several state-of-the-art evolutionary algorithms for binary combinatorial problems. The experimental studies presented provide useful conclusions about the schemes for combining ideas from simulated annealing and evolutionary algorithms that may improve the performance of these kinds of approaches and suggest that these hybrids metaheuristics represent a competitive alternative for binary combinatorial problems.

Index Terms—Hybrid metaheuristics, Simulated annealing, Evolutionary algorithms, Combinatorial optimisation.

I. INTRODUCTION

OVER the last few years, a large number of search algorithms have been presented that do not simply follow the concepts of one single classical metaheuristic [1], [2], but attempt to obtain the best from a set of metaheuristics (and even other kinds of optimisation methods) that perform together and complement each other to produce a profitable synergy from their combination. These approaches are commonly referred to as *hybrid metaheuristics* (HMs) [3], [4], [5].

Simulated annealing (SA) [6], [7], [8] is commonly said to be the first algorithm extending local search methods with an explicit strategy to escape from local optima. The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution in order to escape from local optima. The probability of making such a move is decreased during the search process. Although it was proposed in 1983, SA is still the object of further studies [9], applied to many optimisation problems, or used as a component of other search algorithms

[8], [10]. It is precisely because of its outstanding role in the metaheuristic field that further studies to obtain more effective SA models are encouraged.

Evolutionary algorithms (EAs) [11], [12] are stochastic search methods that mimic the metaphor of natural biological evolution. EAs rely on the concept of a *population* of individuals (representing search points in the space of potential solutions to a given problem), which undergo probabilistic operators such as mutation, selection, and (sometimes) recombination to evolve toward increasingly better fitness values of the individuals. There has been a variety of slightly different EAs that, basically, fall into three different categories that have been developed independently from each other. These are *evolutionary programming* [13], *evolution strategies* [14], and *genetic algorithms* (GAs) [15]. EAs offer practical advantages to researchers facing difficult optimisation problems because they may locate high performance regions of vast and complex search spaces. Other advantages include the simplicity of the approach, their flexibility, and their robust response to changing circumstances.

The hybridisation of EAs is becoming popular due to its ability to handle several real-world problems involving complexity, noise, imprecision, uncertainty, and vagueness [16], [17], [18], [19]. A wide variety of metaheuristics such as tabu search [20], greedy randomized adaptive search procedure [21], and iterated local search [22], among others, have been employed to develop hybrid approaches with EAs. In this work, we focus on the use of SA to design HMs with EAs (HMs-EA/SA) due to its prominent role in the field of the hybrid EAs [23], [24], [25], [26], [27].

The current relevance of HMs-EA/SA can be shown through the visibility of this topic at the ISI Web of Science. Figure 1 shows an important number of publications and citations per year, as well as an increasing trend. We can conclude that, although the first items related to this topic appeared in 1992, nowadays HMs-EA/SA are subject of great interest and there is an important research community associated to their study. Moreover, in Figure 2 we can observe that the number of works at the ISI Web of Science considering HMs-EA/SA is greater than those of hybrid metaheuristics combining EAs and other metaheuristics (greedy randomized adaptive search procedure (GRASP), iterated local search (ILS), variable neighbourhood descent (VND), iterated greedy (IG), and tabu search). In addition, we can highlight that many different instantiations of HMs-EA/SA are presented to solve real-world problems covering domains that range from

Francisco J. Rodriguez is with the Department of Computer Science and Artificial Intelligence, University of Granada, Spain, 18071, e-mail: fjrdriguez@decsai.ugr.es.

Carlos García-Martínez is with the Department of Computing and Numerical Analysis, University of Córdoba, Spain, 14071.

Manuel Lozano is with the Department of Computer Science and Artificial Intelligence, University of Granada, Spain, 18071.

Manuscript received ; revised -

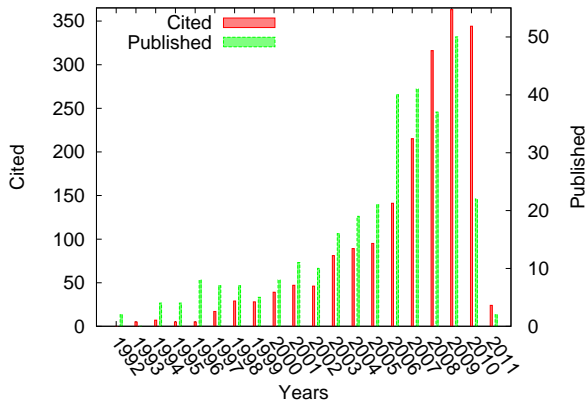


Fig. 1: Number of publications and citations per year for HMs-EA/SA (Web of Science)

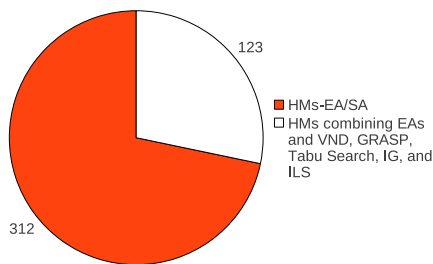


Fig. 2: Comparison between publications considering HMs-EA/SA and hybrid algorithms combining EAs and other metaheuristics (Web of Science)

the design of combinational circuits [28], routing problems in telecommunications [29], cluster analysis [30], optimisation of support vector machines parameters [31], scheduling problems [32], protein structure prediction [33] and image processing [34], to name but a few.

The goal in this article is threefold. Firstly, we attempt to paint a more complete picture of HMs-EA/SA than ever before. To do so, we structure and organise the knowledge about the HM-EA/SA approaches found in the literature by proposing a taxonomy for HMs-EA/SA based on those conceived by Talbi [3] and Raidl [4] for HMs. Our second objective is to study the empirical behaviour of the different kinds of approaches according to the proposed taxonomy, analysing what schemes provide a better performance. Finally, we want to study the synergistic relationships created by the hybridisation of EAs and SA in these approaches. Suitably combining the complementary algorithms concepts can provide hybrid approaches with a better performance than that obtained by EAs or SA separately.

The remainder of this paper is organized as follows. In Section II, we present an overview of the HM-EA/SA approaches found in the literature and propose a taxonomy that characterises them. In Section III, we describe the experimental framework employed in this work. In Section IV, we compare the performance of a set of representative HMs-EA/SA instances belonging to different categories. In Section V, we analyse the performance of the groups of HMs-

TABLE I: Taxonomy for HMs-EA/SA

General Categories		HM-EA/SA Categories	Instances
Collaborative	Teamwork	Multiple EAs and SAs	DCHCSA
		Multiple SAs	SSSA [23], CSA [35], ESA [36], GAMSA [37]
	Relay	EA then SA	HHSAGA [38], SAGA [39]
		SA then EA	GA-PSA
Integrative	TeamWork	MA with SA as local search	AGA [40], GASAHA [31], IGA-SA [41], GSAAL [42], GSAA [24]
		SA-based EA selection	HGA-BTS [43], GESA [44], HGA-BS[45]
		SA-based EA mutation and crossover	SAGACIA [46], ARSAGA [47], GSAAIA [48], HGA-SAM/R [49]
		SA-based EA replacement	PRSA [50], PGSA [25], GSA [51], NPOSA [52], MPGSAA [26], GSA-MLE [53]
	Relay	EA-based SA component	SALGeS [54], GAMSA [37]

EA/SA studied. In Section VI, we perform a synergy study of the HMs-EA/SA compared. In Section VII, we pit the best performing HM-EA/SA approach against some state-of-the-art EAs for binary combinatorial problems. Finally, in Section VIII, we present conclusions and future work.

II. HMs BASED ON EAS AND SA: OVERVIEW AND TAXONOMY

We have grouped different instances of HMs-EA/SA appearing in the literature into two broad categories (Table I summarises the HMs-EA/SA found and the category they belong to). These two groups are specified following two well-known existing taxonomies for HMs [3], [4], which are based on the architecture of the algorithms:

- *Collaborative HMs*. These are based on the exchange of information between different self-contained metaheuristics (and possibly other optimisation techniques) running sequentially or in parallel.
- *Integrative HMs*. In this case, one algorithm is considered a subordinate, embedded component of a master metaheuristic, which governs the search process.

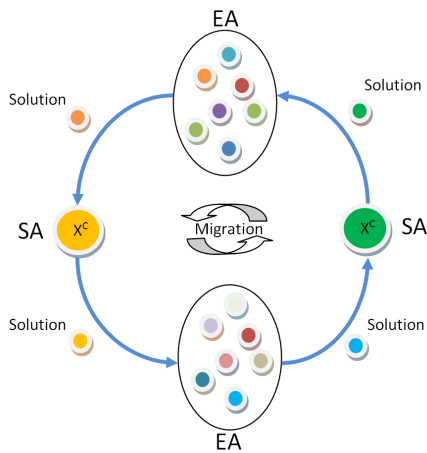


Fig. 3: Multiple EAs and SAs running in parallel

A. Collaborative HMs-EA/SA

Collaborative HMs apply different self-contained metaheuristics. These metaheuristics can be regarded as black boxes and the cooperation takes place through the exchange of some kind of information. According to the way metaheuristics are executed, collaborative HMs-EA/SA can be subdivided into teamwork or relay categories [3].

1) *Teamwork Collaborative HMs-EA/SA*: In *teamwork collaborative HMs*, there are several metaheuristics that work in parallel and exchange solutions, parameters, etc. from time to time. To the best of our knowledge, there is no proposed method arranging *multiple EAs and SAs* running in parallel. However, we may easily devise teamwork collaborative HM-EA/SA models based on distributed GAs [55]. In this scheme, a single population is decentralised by partitioning it into several subpopulations (islands or demes), where island GAs are run performing sparse exchanges (migrations) of individuals. It is easy to conceive a model that replaces some GAs with SAs. In this case, we can consider some subpopulations that are optimized by EAs and other subpopulations formed by only one individual and optimised by an SA process (Figure 3). An instance of this kind of approach called *DCHCSA* will be presented in Section IV.

There are several proposals in the literature worthy of mention in this section, which consider the execution of multiple SAs that cooperate to explore the search space by exchanging solutions, parameters, etc (Figure 4). They do not consider the application of self-contained EAs, but evolutionary concepts underlie the general scheme. These algorithms consider a population of agents that evolve by means of neighbourhood and stochastic selection operators. For this reason, authors may argue that they resemble EAs. Though these models are hardly seen as HM-EA/SA approaches, they are considered in this study for the sake of coverage. They are referred to as *multiple SAs*.

The *evolutionary SA* (ESA) algorithm [36] considers a population of solutions. It selects one individual according to the running selection rule, operates on it with a neighbourhood operator, and evaluates whether to put it back into the population according to a particular replacement rule. The new

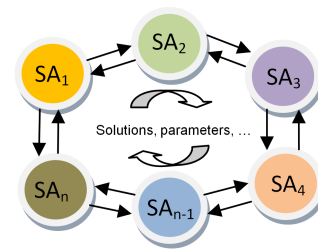


Fig. 4: Multiple cooperating SA processes

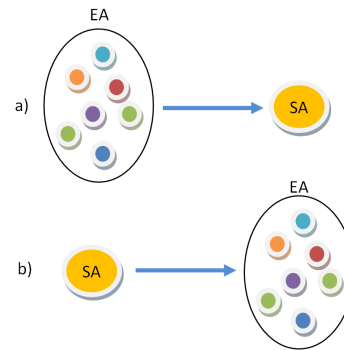


Fig. 5: EAs and SAs in a pipeline fashion

individual provided by the neighbourhood function is adopted according to the Metropolis acceptance criterion. The ESA algorithm can be seen as a population of multiple SA processes that exchange their current solutions at every iteration.

Sample-sort SA (SSSA) [23] maintains an array of samplers operating at static temperatures. At each iteration, each one firstly considers whether to accept any of the states of its neighbouring samplers. Then, each sampler performs a standard iteration of SA. *Coupled SA* (CSA) [35] considers a population of samplers where the acceptance probability of any sampler making an uphill move depends on the states of the other samplers.

2) *Relay Collaborative HMs-EA/SA*: In relay collaborative HMs, several metaheuristics are executed in a pipeline fashion. The output of each algorithm is supplied as the input to the next one (Figure 5). Depending on the order of execution, two kinds of basic approaches can be found:

a) *EA then SA*: In [38], the *highly hybrid GA+SA* (HHSAGA) generates a number of random initial solutions and runs a GA through a fixed number of iterations. After the GA ends, each individual in the final population is optimised by an SA (Figure 5.a). At the end of the execution of all the SA processes, a new population is generated using the solutions obtained by the SA processes. Then, the GA starts again. This cycle is repeated until the termination condition is reached. This scheme also appears in the *parallel heuristic SA+GA* (SAGA) presented in [39].

b) *SA then EA*: By way of contrast, it is possible to devise an algorithm that starts from SA and uses EAs to enrich the solutions found (Figure 5.b). The scheme that most directly fits this pattern is an EA that uses SA as a method to initialise the population. An implementation of this scheme called *GA-PSA* (*GA with population initialised by SA*) will be presented in Section IV.

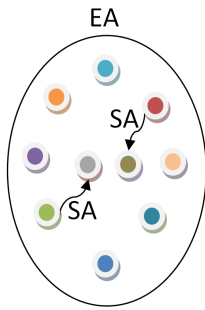


Fig. 6: MA with SA as local search procedure

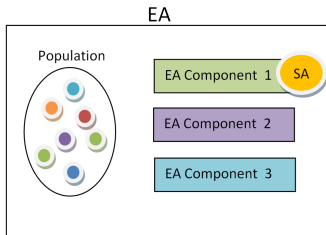


Fig. 7: EA with SA-based components

B. Integrative HMs-EA/SA

Integrative HMs address the functional composition of a single optimisation method. In this kind of HMs, a given function of a metaheuristic is replaced by another metaheuristic [3]. Integrative HMs can be also subdivided into teamwork or relay categories.

1) *Teamwork Integrative HMs-EA/SA*: In *teamwork integrative HMs*, one metaheuristic (subordinate) becomes a component of another population-based metaheuristic (master) [3]. In the case of HMs-EA/SA, we can find the following approaches:

a) *Memetic algorithms with SA as local search procedure*: *Memetic algorithms* (MAs) [56] combine an EA in charge of the global exploration with a local search procedure, which is executed within the EA run, looking for a synergy that takes benefits from both.

There are several instances of MAs in which the local search is performed by means of SA (Figure 6). *Annealing-GA* (AGA) [40] is an EA where the best solutions are optimised by SA processes, which perform cooling at each iteration and apply the Metropolis acceptance mechanism. Moreover, we can find several recent proposals that apply this scheme to solve real-world problems. In [31], a *GA-SA hybrid algorithm* (GASAHA) is proposed to optimise the parameters of a support vector machine. In each iteration, the GA operates on the population using three basic genetic operators to produce a new population. Afterwards, the GA applies the SA to the best individual of the GA population to further improve it. In [24], the *Genetic simulated annealing algorithm* (GSAA) deals with a similar problem. In this case, an SA process improves all the solutions. More examples of MAs based on SA can be found in [41] (*Improved GA-SA*, IGA-SA) and [42] (*Genetic SA algorithm based localization*, GSAAL).

b) *SA-based EA component*: Another teamwork integrative HM-EA/SA approach consists of defining EA components

by using principles of the SA algorithm (Figure 7). Specifically, the EA components that have been replaced or extended in the literature by the SA are the selection mechanism, the crossover and mutation operators, and the replacement strategy.

SA-based EA selection

In [43], a *hybrid GA uses a Boltzmann tournament selection* (HGA-BTS) function to provide asymptotic convergence. This selection function uses pairwise probabilistic acceptance and anti-acceptance mechanisms on three individuals from the population. The anti-acceptance competition takes place between two of the three selected individuals. The acceptance competition takes place between the winner of this competition and the other individual.

In [45], the number of offspring that an individual can contribute to the next generation is calculated by implementing Boltzmann scaling on the fitness function (*Hybrid GA with Boltzmann scaling*, HGA-BS) by varying the selective pressure as a function of the temperature.

In the *guided evolutionary SA* (GESA) algorithm [44], there are two levels of competition. In the first one, the children of the same family (i.e., generated from the same parents) compete with each other and only the one with the best fitness value survives. At the second level of competition, the best child is compared with its parents to find the members for the next generation. A Boltzmann probability is applied in order to decide whether the child will be accepted.

SA-based EA mutation and crossover

The approach proposed in [49] combines SA with GA by extending the mutation and crossover operators with SA (*hybrid GA with SAM/SAR*, HGA-SAM/R). Mutated and recombined solutions are accepted according to the standard SA acceptance condition.

In [48], the *genetic SA algorithm-based inverse algorithm* (GSAAIA) uses the SA technique to control the mutation operator in the EA. The member to be mutated is perturbed according to the Gaussian probabilistic distribution function and its variance is controlled by the SA technique.

Adaptive real-parameter SA-GA (ARSAGA) [47] with SA-based mutation operator samples a new solution from the neighbourhood of the solution generated by the crossover operator. The new solution is accepted according to the SA acceptance criterion. The same procedure is applied in the SAGACIA algorithm (*SA, GA, chemotaxis algorithm, integrated, algorithm*) [46] to accept new solutions during the phase called rough search, which can be seen as a mutation operator.

SA-based EA replacement

Parallel recombinative SA (PRSA) [50] iterates over a population of solutions, employing a crossover operator and an unary neighbourhood operator (mutation). Offspring replace their parents according to the acceptance mechanism of SA.

In [52], an algorithm called (*new population-oriented SA*, NPOSA) employs, in addition, a replacement strategy based on the SA acceptance criterion. However, each individual has its own local temperature according to the individual's rank,

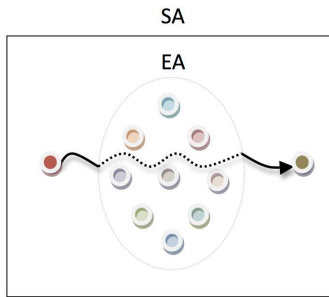


Fig. 8: SA with EA-based functions

settled by its cost value. If an individual finds that its own cost is higher than those of the other individuals, then it raises its local temperature to heighten the probability of uphill climbing in the cost space. Otherwise, it drops its temperature to lower this probability.

The *genetic SA* (GSA) is a parallel algorithm introduced in [51] that maintains small independent populations of solutions and performs periodic exchanges between the populations. Each subpopulation generates a direction and distance in the subpopulations grid to obtain a mating from another subpopulation. After that, the crossover and mutation operators are performed. A solution that replaces the resident candidate is selected from the current resident candidate and the offspring. An SA acceptance criterion is used to select this solution. Another parallel hybrid genetic SA approach that uses the SA acceptance mechanism to replace the solutions of the population is presented in [25] (*Parallel genetic SA*, PGSA).

More examples that follow this scheme to get HMs-EA/SA can be found in [26] (*multi-population parallel genetic SA algorithm*, MPGSAA) and [53] (*genetic SA algorithm*, GSA-MLE).

2) *Relay Integrative HMs-EA/SA*: This kind of HMs represents algorithms in which a given metaheuristic is embedded into a trajectory-based metaheuristic [3]. Specifically, for the HMs-EA/SA, an EA is used to perform one or more functions in an SA process (Figure 8). The *SA based on local genetic search* (SALGeS) [54] presents an EA designed specifically to play the role of the SA neighbourhood operator. In particular, a steady-state EA creates one single candidate solution at each iteration, by crossing over the current solution of the master SA and another one from the population. Afterwards, the master SA applies an acceptance mechanism to decide which solution becomes the new current solution, either the candidate solution or the current one. The other solution is inserted into the population by a replacement strategy.

The *GA-based multiple SA* (GAMSA) [37] extends the idea of SALGeS by considering the execution of multiple SA processes that share a unique steady-state EA. The existence of several SA processes promotes diversification by exploring different regions of the search space. On the other hand, the population of the EA allows the SA agents to communicate with one another in order to explore the search space. GAMSA can be seen as a population of SA processes that cooperate to explore the search space. In this sense, GAMSA can be classified as *teamwork collaborative*. On the other hand,

```

<HM-EA/SA> → <design-issues-EA/SA><implementation-issue>
<design-issues-EA/SA> → <hierarchical-EA/SA><flat>
<hierarchical-EA/SA> → <LRH-EA/SA>|<LTH-EA/SA>|<HRH-EA/SA>
|<HTH-EA/SA>|LRH(<metaheuristic>(<HM-EA/SA>))|
LTH(<metaheuristic>(<HM-EA/SA>))
<LRH-EA/SA> → LRH(<SA-metaheuristic>(<EA-metaheuristic>))|
LRH(<LRH-EA/SA>(<metaheuristic>))
<LTH-EA/SA> → LTH(<EA-metaheuristic>(<SA-metaheuristic>))|
LTH(<LTH-EA/SA>(<metaheuristic>))|
LTH(<HTH-EA/SA>(<metaheuristic>))
<HRH-EA/SA> → HRH(<EA-metaheuristic>+<SA-metaheuristic>)|
HRH(<met-or-null>+HRH(<HM-EA/SA>+<met-or-null>))|
HRH(<SA-metaheuristic>+<EA-metaheuristic>)
<HTH-EA/SA> → HTH(<EA-metaheuristic>,<SA-metaheuristic>)|
HTH(<metaheuristic>,<HM-EA/SA>)
<EA-metaheuristic> → EA|
LTH(<EA-metaheuristic>(<metaheuristic>))|<HTH>
<SA-metaheuristic> → SA|
LRH(<SA-metaheuristic>(<metaheuristic>))
<met-or-null> → <metaheuristic> | null
<metaheuristic> → LS|TS|SA|GA|ES|GP|GH|AC|SS|
NM|CLP|<HM-EA/SA>

```

Fig. 9: Grammar for HMs-EA/SA

each SA process in GAMSA is an HM-EA/SA that can be individually classified into the *integrative relay* category.

C. Grammar for HMs-EA/SA

The HMs-EA/SA are a specialised category of the HMs, and therefore any HM-EA/SA should be recognisable by the grammar proposed by Talbi [3]. In this section, we present a specialisation of this grammar that recognizes only the HMs-EA/SA (Figure 9). Undefined non-terminal and terminal symbols can be consulted in Talbi's grammar.

Notice that we have decided to consider the parallel evolution of a set of agents (<HTH>) as a kind of evolutionary method (<EA-metaheuristic>). Besides this, we have included the terminal symbol *null*, which does nothing, in order to simplify the <HRH-EA/SA> definition.

III. EXPERIMENTAL FRAMEWORK

A. Test Problems

In this section, we detail the test problems that were used for the upcoming empirical studies. The test suite is composed of 27 binary combinatorial optimisation problems, 13 of which were artificial problems and the remaining 14 were obtained from real-world applications. Table II outlines their name, number of bits (D), a value (f^*) that stands for either the fitness value of the global optimum, known best solution, or upper bound presented in the literature, and reference. All of them have been formulated as maximisation problems. BQP and Maxcut instances can be obtained from the corresponding files from the *BiqMacLibrary*¹, and Multiple knapsack problems, from the *SAC – 94Suite*².

B. Running Conditions

In order to perform a fair comparison between different search methods, we will run every algorithm with the same

¹<http://biqmac.uni-klu.ac.at/>

²<http://elib.zib.de/pub/Packages/mp-testdata/ip/sac94-suite/>

TABLE II: Tackled test problems

Prob.	Name	D	f^*	Ref.
1	Royal road problem (400, 8)	400	1	[57]
2	Trap problem	36	220	[58]
3	Deceptive problem	39	390	[59]
4	Bipolar deceptive problem	396	1	[60]
5	Overlapping deceptive problem	399	1	[60]
6	M-Sat(100,1200,3)	100	1	[61]
7	M-Sat(100,2400,3)	100	1	[61]
8	NkLand(48,4)	48	1	[62]
9	NkLand(48,12)	48	1	[62]
10	HIFF(2, 5, true)	32	192	[63]
11	HIFF(3, 4, false)	81	211	[63]
12	PPeaks(10,100)	100	1	[62]
13	PPeaks(100,100)	100	1	[64]
14	PPeaks(50,150)	150	1	[64]
15	PPeaks(50,200)	200	1	[64]
16	BQP(bqp50-1)	50	2098	[65]
17	BQP(bqp500-1)	500	116586	[65]
18	BQP(be120.3.3)	120	Not known	[65]
19	BQP(be200.8.5)	200	Not known	[65]
20	Maxcut(pm1s_80.6)	80	73	[66]
21	Maxcut(w09_100.2)	100	2738	[66]
22	Maxcut(g05_100.5)	100	1436	[66]
23	Maxcut(pw05_100.6)	100	8217	[66]
24	Maxcut(ising2.5-250_5555)	250	7919449	[66]
25	Multiple knapsack p. (weish03)	30	4115	[67]
26	Multiple knapsack p. (pet5)	28	12400	[67]
27	Multiple knapsack p. (pb4)	29	95168	[67]

budget of fitness evaluations. Each run of a search algorithm on a test problem will perform at most 10^5 fitness evaluations. The performance measure is the average of the best fitness values found over 50 independent runs, because we are interested in the regular performance of the compared algorithms. Moreover, the evolution of the best solution found throughout the whole execution will be studied as well.

Non-parametric tests have been used to compare the results of different optimisers [68] given they do not require explicit conditions for being conducted. In particular, mean ranking for each algorithm is firstly computed according to the Friedman test [69]. This measure is obtained by computing, for each problem, the ranking r_j of the observed result for algorithm j , assigning to the best of them the ranking 1, and to the worst the ranking J (J is the number of algorithms). Then, an average measure is obtained from the rankings of this method for all the test problems. Secondly, the *Iman and Davenport* test [70] is applied to check the existence of performance differences between all the considered algorithms. Finally, the *Holm* test [71], is used to detect performance differences between the best ranked algorithm and the remainder. Moreover, we have used the *Wilcoxon matched-pairs signed-ranks* test to perform pairwise comparisons.

IV. COMPARATIVE STUDY OF THE HMS BASED ON SA AND EAS

A representative set of the algorithms revised in Section II is chosen to study their behaviour experimentally. This set has been built combining recent proposals and those that best fit the general scheme of each category. Parameters related to the SA process are fixed according to the study performed in [54], for those algorithms whose original publications were not clear: 100 iterations per cooling event, 0.99 as the cooling factor, geometric cooling, and logistic acceptance mechanism. Regarding the initial temperature, two random solutions are firstly generated. We set a desired probability p_d of accepting the worst solution from the best one. Then, we compute the corresponding T_0 value, according to the applied acceptance criterion. The p_d value is set to 0.4. We have followed the recommendations of the original publications for the remaining parameters. Next, we specify the rest of the parameter settings of the HMS-EA/SA considered, grouped attending to their corresponding category. Moreover the specification according to the grammar presented in Section II-C is detailed for each category or algorithm.

Teamwork Collaborative HMS-EA/SA (<HTH-EA/SA>) (Section II-A1)

- **Multiple EAs and SAs:** we have developed an algorithm following the guidelines in Section II-A1 (Figure 3) that is called *DCHCSA*. It considers two CHC algorithms and two SA processes that are executed in parallel. The ring topology, alternating CHC and SA, is applied. Every certain number of fitness evaluations, the current solution of each SA is sent to the next CHC algorithm anticlockwise. This solution replaces the worst solution in the population of the CHC algorithm. At the same time, each CHC algorithm sends its best solution found so far to the next SA process anticlockwise. This solution becomes the new current solution of this SA process. The migration process takes place every 75 fitness evaluations of each algorithm.
- **Multiple SAs:** two approaches that belong to this category are considered in the study. *SSSA* [23] maintains 100 samplers. The neighbourhood size is set to 1, i.e., each sampler may accept any of the two states of its consecutive samplers (just one for the extreme samplers). *CSA* [35] has 5 samplers. Its parameter α , for the variance control procedure, is set to 0.05.

Relay Collaborative HMS-EA/SA (Section II-A2)

- **EA then SA (HRH(EA+SA)):** *HHSAGA* [38] considers a generational GA. Its population size is set to 60. Two-point-crossover is used. In order to adapt the algorithm to the experimental framework described, only two steps (one GA step and another SA step) are considered and fitness evaluations are shared between GA and SA. The GA stage consumes 75 percent of fitness evaluations and the SA stage the remaining 25 percent.
- **SA then EA (HRH(SA+EA)):** *GA-PSA* implements the approach presented in Section II-A2, in which the population of a GA is initialised by an SA (Figure 5.b). The

GA uses a population of 60 individuals. Mutation (one-flip) and crossover (two-point-crossover) are considered to evolve the individuals of the population. The method used by the SA to initialise the population is similar to that proposed in [40]. The population is filled with the solutions generated within the SA trajectory until it rejects the new solution proposed by the neighbourhood operator. At that moment, a new SA trajectory starts from another random solution. This process is repeated until the population is filled up.

Teamwork Integrative HMs-EA/SA (LTH(EA(SA))) (Section II-B1)

- **MA with SA as local search procedure:** AGA [40] uses a mutation probability equal to $1/(10 \cdot D)$. The population size is set to 60.
- **SA-based EA selection:** GESA [44] maintains a population of 100 individuals distributed into 10 families. The offspring are generated using two-step crossover.
- **SA-based EA mutation:** ARSAGA [47] sets the parameters as follows: population size equal to 40, crossover and mutation probability equal to 0.7 and 0.01, respectively, $\alpha = 0.9$, $\beta = 0.02$, and $N_{frozen} = 10$. Two-point-crossover is applied.
- **SA-based EA replacement:** PRSA [50] applies a mutation probability equal to $1/D$ and the population size is set to 60.

Relay Integrative HMs-EA/SA (LRH(SA(EA))) (Section II-B2)

- **EA-based SA component:** For this study we have implemented a simplified version of SALGeS [54] in which the individual from the population selected to perform the crossover is chosen randomly and the individual discarded from the current solution and the candidate one replaces the worst solution from the GA population (only if it is better). It considers a population size equal to 500 and a probability $p_y = 1/D$ associated with the crossover operator.

Teamwork Collaborative / Integrative Relay

- **GAMSA** [37] (LTH(HTH(SA,HTH(SA,...))(EA))): The number of simultaneous SA processes is set to 32, the maximum number considered in the original publication. The different SAs act sequentially, performing individual iterations, and use the same selection and replacement operators as the simplified SALGeS.

We have performed an experimental study comparing the previous HMs-EA/SA in the experimental framework described in Section III. In Figure 10, we can see graphically the Holm test results (the Iman-Davenport test finds significant performance differences between the considered algorithms because its statistical value, 117.166, is greater than its critical one, 1.822). The Holm test finds that the best results are achieved by GAMSA. It detects significant differences between GAMSA and CSA, ARSAGA, GESA, PRSA, SSSA, HHSAGA, GA-PSA, and AGA.

Table III summarises the results of applying the Wilcoxon test, with p -value = 0.05 and 0.1, to compare the results of

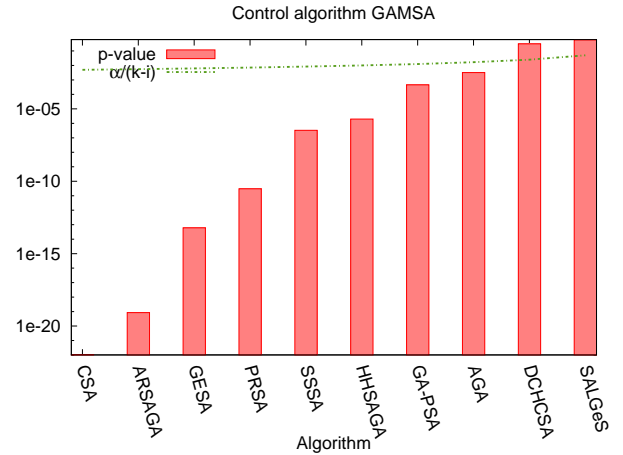


Fig. 10: Holm test results

TABLE III: GAMSA vs. other HMs-EA/SA (Wilcoxon's tests with p -value=0.05 and 0.1 and critical value = 107 and 119, respectively)

Algorithms	R^+	R^-	Sig. differences? 0.05/0.1
GAMSA vs DCHCSA	262	116	~/+
GAMSA vs SALGeS	259	113	~/+

GAMSA and the algorithms for which the Holm test does not detect significant differences (DCHCSA and SALGeS). The last column indicates whether GAMSA performs statistically better (+), worse (-), or without significant differences (~) to its competitor. According to these results, the Wilcoxon test determines that GAMSA is statistically better than DCHCSA and SALGeS with p -value=0.1. It is worth noting the good behaviour of DCHCSA, a simple HM-EA/SA variant (teamwork collaborative approach) derived from distributed GAs that, as far as we know, had not been implemented before

Table VI, in Appendix A, shows the averaged results for each test problem. Moreover t-test results, comparing GAMSA (the best performing HM-EA/SA) and the remaining approaches, indicate for each instance whether GAMSA results are statistically better (+), worse (-), or equal (~) to those of the other HMs-EA/SA. Regarding to these results, we see that:

- In 23 out of 27 instances, SALGeS and GAMSA reach the best solution found by any other HM-EA/SA approach.
- GAMSA obtains better results than PRSA, ARSAGA, GESA, HHSAGA, SSSA, and CSA for almost every test problem. For the remainder, the test performed does not show statistical differences between its results and those of its competitors.
- Only GA-PSA, AGA, DCHCSA, and SALGeS are able to significantly improve the GAMSA results in some instances. GA-PSA beats GAMSA on one problem, AGA on two, DCHCSA on four, and SALGeS on five.

Next, we study the averaged behaviour of the HM-EA/SA instances along the whole run by means of convergence

graphs, as follows (Figure 11):

- Taking into consideration the highest and lowest fitness values achieved by all the algorithms for each test problem, we have normalised every result, throughout all the runs, to the interval $[0, 1]$.
- Then, mean values over the 27 problems, have been obtained for each algorithm throughout the 10^5 evaluations.

In addition, Figure 12 shows the evolution of the averaged rankings over the 27 problems of all the algorithms, when their current best solutions according to the number of consumed evaluations are compared. Notice that Figures 11 and 12 depict two different kinds of information; in particular, averaged convergence graphs may be strongly affected by distant fitness values in a few functions, e.g. Royal Road; by contrast, rankings are robust with regards to distant fitness values, but may be strongly affected by small changes to many functions.

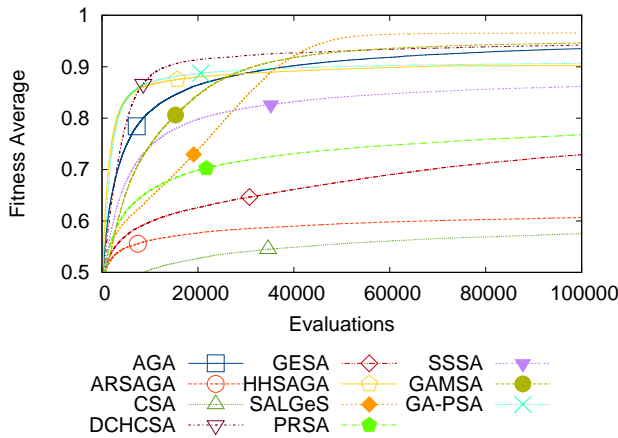


Fig. 11: Convergence graphs of the HMs-EA/SA

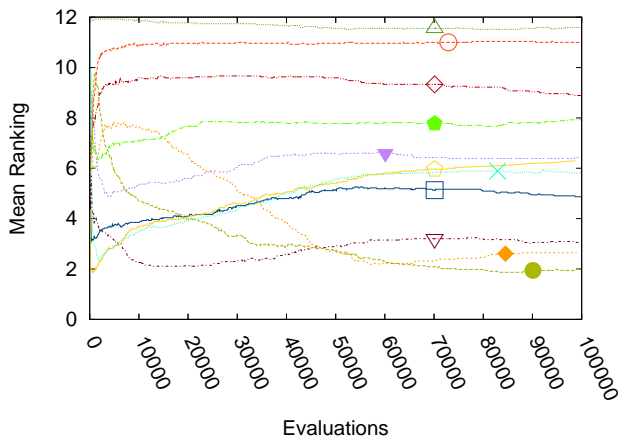


Fig. 12: Evolution of the ranking of the HMs-EA/SA

According to Figure 11, we may remark that:

- In the beginning of the execution, all the algorithms but PRSA, GESA, ARSAGA, CSA, and SALGeS move rapidly towards better solutions.
- Over the course of the 40,000 evaluations, SALGeS defeats all its competitors, considering the averaged fitness.

- In the intermediate stage of the execution, GAMSA catches all its competitors but SALGeS and outperforms them.
- In the latter stages, SALGeS seems to outperform GAMSA according to the convergence graph. This behaviour can be explained by the fact that, as we mentioned before, averaged convergence graphs may be strongly affected by distant fitness values in a few functions. However, according to the Wilcoxon test results for p -value=0.1 in Table III, we see that GAMSA outperforms to SALGeS considering a statistical study.

According to Figure 12, we observe the following:

- All the algorithms but GAMSA, SALGeS, GA-PSA, and HHSAGA more or less maintain their respective rankings from the beginning of the execution until the end.
- GAMSA and SALGeS show an improving trend from the beginning and reach the best rankings at the intermediate stages.
- GA-PSA and HHSAGA have their best rankings at the very beginning of the execution, but lose competitiveness gradually.
- At the end of the execution, GAMSA is the most competitive algorithm for all the problems in general.

V. PERFORMANCE OF HMs-EA/SA BY CATEGORIES

In this section, we study the behaviour of the HMs-EA/SA by categories in order to assess the success of each group of algorithms. In Figure 13, we can see the average ranking obtained by each algorithm and the category to which it belongs. HMs-EA/SA have been grouped by categories and the height of each column is proportional to the ranking. Therefore, the lower a column is, the better its associated algorithm is. Taking into account this figure, we can conclude the following:

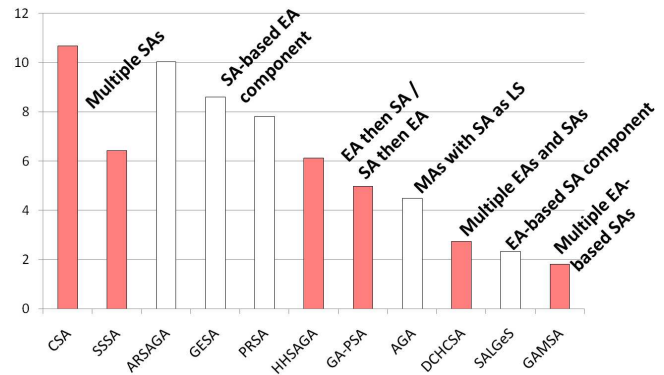


Fig. 13: Average rankings of the HMs-EA/SA versions

- The ‘multiple SAs’ category includes the CSA algorithm, which presents the poorest performance among all the studied algorithms. The other algorithm in this class, SSSA, appears in the group of the five algorithms with the worst average rankings. The three instances of the ‘SA-based EA component’ category (ARSAGA, GESA, and PRSA) are among the algorithms forming this group as well. A feature shared by these two categories is that

the hybridisation EA/SA is built employing incomplete formulations of either EA or SA. Specifically, an effort is made to incorporate principles of one of these metaheuristics in the other (population of solutions in SA, multiple SAs, and SA acceptance criterion in EA, SA-based EA component)

- Better performances are achieved by the algorithms in the other categories. In this case, they are conceived as HMs composed of self-contained EAs and SAs, and, cooperating somehow.
- The integrative approach is present in three out of the four best ranked HMs-EA/SA. They are AGA (MA with SA as local search procedure), SALGeS, and GAMSA (both belonging to EA-based SA component). In these HMs, one metaheuristic is specialised in to play a specific role inside the other metaheuristic. Specifically, in AGA, SA affords EA refined solutions and, in SALGeS and GAMSA, EA acts as an SA neighbourhood operator.
- The hybridisation technique used by SALGeS and GAMSA, that involves replacing some components in metaheuristics with customised EAs (*evolutionary components*) to develop the same work more effectively and with a relatively low computational cost, was also studied in [22]. As an example, the authors contributed an iterated local search algorithm with an evolutionary perturbation method, which is a micro EA that effectively explores in the neighbourhood of particular solutions. This algorithm turned out to be very competitive with state-of-the-art iterated local search metaheuristics for binary optimisation problems. The suitable results shown by GAMSA and SALGeS provide additional evidence for the effectiveness of this novel hybridisation paradigm.
- The group of the four best performing algorithms is completed by an HM-EA/SA belonging to the teamwork collaborative approach (DCHCSA). Interestingly, it has better results than the two algorithms representing the relay collaborative approach (HHSAGA and GA-PSA).

We can highlight that the ranking obtained when comparing the performance of the different HM-EA/SA approaches (Figure 13) may be used as a guideline for researchers attempting to tackle specific optimisation problems by building HMs-EA/SA, or even with other trajectory-based metaheuristics (tabu search, local search procedures, guided local search, etc.).

VI. A SYNERGY TEST

One of the most important aspects of studying the behaviour of an HM is to consider the synergy produced by the combination of the composing metaheuristics. In fact, exploring the complementary character of the different optimisation strategies involved in the hybrid metaheuristics is the main motivation behind the hybridisation, that is, hybrids are believed to benefit from synergy [72]. In order to assess the amount of synergy that appears when combining two or more components, or whether it does or does not appear, the usual practise involves the comparison between the hybrid algorithm and the sole usage of its components [73], [74], [75]. In this

section, we study the synergy produced by combining SA and EAs of previous HMs-EA/SA with regards to the following algorithms:

- *Standalone SA* (SA) [6], [7]. The parameter values are those detailed in Section IV.
- *Canonical generational GA* (CGGA). This considers a population of 60 individuals, binary tournament selection, one-flip mutation (probability 0.006), and two-point crossover (probability 1).
- *Canonical steady-state GA* (CSSGA). The parameters and operators are the same as in CGGA. In each iteration, two offspring that replace their parents are generated.
- *Cross-generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation* (CHC) [76]. This is an EA that involves the combination of a selection strategy with a very high selective pressure, and several components inducing diversity.

Table IV (column ‘Comp. on results’) summarises the results of applying the Wilcoxon test with p -value = 0.05, where the values of R^+ (associated to the HM-EA/SA) and R^- (associated to the single metaheuristic) of the test are specified. According to these results, we may conclude that:

- SALGeS and GAMSA are the only HMs that perform better than the sole application of SA, any canonical EA, and CHC. So, we can state that this hybrid approach (EA-based SA component) supplies a better performance, exploiting and uniting the advantages of the individual metaheuristics and, therefore, showing a profitable synergy.
- Considering the remaining hybridisations, only DCHCSA presents a positive result in terms of synergy. The other ones do not produce better results, for the considered experimental framework, than the sole application of one of their intrinsic metaheuristics. These results allow us to conclude that the simple combination of several metaheuristics does not ensure success. It is necessary to study the way the composing metaheuristics are combined in order to achieve a positive synergy between them.

Finally, it is interesting to analyse the computational time required for the HMs with regards to their standard counterparts. Table IV (column ‘Comp. on time’) shows the results of applying the Wilcoxon test, using as performance measure the average over 50 independent runs of the time employed to perform 10^5 fitness evaluation on each problem. The last column indicates whether the corresponding HMs requires statistically more time (–), less (+), or there is no significant differences (\sim) with regards to their standard counterparts. The Wilcoxon test results show that all the HMs, but HHSAGA, take statistically more time than SA to solve the considered problems. On the contrary, it is significant that there is no HMs exceeding the computational time required by the most computationally expensive counterparts, the evolutionary algorithms (CGGA, CSSGA, and CHC). In particular, we can highlight that SALGeS and GAMSA, which are the only ones that reach a profitable synergy and outperform the quality of the solutions provided by all their standard counterparts, do not exceed computational time required by their most expensive

counterparts, CGGA, CSSGA, and CHC.

VII. BEST PERFORMING HM-EA/SA vs. STATE-OF-THE-ART EAs FOR BINARY COMBINATORIAL PROBLEMS

In this section, we intend to assess the performance of GAMSAs, the best performing HM-EA/SA, with regards to other relevant EAs for binary combinatorial problems found in the literature. The experiments are carried out on the test suite detailed in Section III-A.

- CHC [76] was tested against different GAs, giving better results, especially for hard problems [77]. So, it has arisen as a reference point in the literature. Its population consists of 50 individuals.
- *Variable dissipative mating GA* (VDMGA) [78]. This is a steady-state EA, similar to CHC, in which the number of new chromosomes entering the population in each generation is controlled on-line by a threshold value, genetic diversity, and the population's state of convergence. The results in [78] show the superior performance of VDMGA when compared to other GAs. Its population size is set to 100. The mutation probability is set to 0.006.
- *Context-independent scatter search* (CISS) [79]. CISS is a proposal explicitly designed to tackle general binary optimisation problems. Its performance was compared with that of several general-purpose commercial optimisation tools, obtaining promising results. Population and reference set sizes have been set to 300 and 6, respectively. The other parameters have been set as in [79].
- *Sawtooth GA* (Saw-GA) [80]. This uses a variable population size and periodic partial reinitialisation of the population in the form of a saw-tooth function. For a wide range of problems, the performance of Saw-GA was statistically superior to a standard GA and a micro GA. The average population size is set to 80. Crossover and mutation probabilities are set to 0.85 and 0.05, respectively. Period and amplitude parameters are adjusted to 40 and 75, respectively.
- *Versatile quantum-inspired EA* (vQEA) [81]. vQEA is a novel EA approach based on quantum computing principles. It considers the quantum bit (Qbit) as the smallest information unit, which is defined by the probability at which the corresponding state (0 or 1) is likely to appear when it is collapsed, i.e., read or measured. vQEA considers a population of quantum individuals (a quantum individual is composed of a Qbit string, a realization, and an attractor) that evolve through quantum gate operations. The population of vQEA is divided into g groups each containing d individuals. Attractors are periodically synchronized between individuals in the same group and between different groups. vQEA considers a population of 10 individuals distributed into a unique group. The synchronization of attractors takes place every generation and $\Delta\Theta$, associated to quantum gate, is set to $\pi/100$.

Table V summarises the results of applying the Wilcoxon test for p -value = 0.05, where the values of R^+ (associated

TABLE IV: HMs-EA/SA vs. Standalone SA, Canonical GAs, and CHC (Comparison on results and time) (Wilcoxon's test with p -value=0.05 and critical value = 107)

HMs	Single Metah.	Comp. on results			Comp. on time		
		R^+	R^-	Diff	R^+	R^-	Diff
CSA	SA	0	378	-	64	314	-
	CGGA	0	378	-	321	57	+
	CSSGA	100	278	-	358	20	+
	CHC	0	378	-	309	69	+
ARSAGA	SA	0	378	-	24	354	-
	CGGA	0	278	-	262	116	~
	CSSGA	16	362	-	60	318	-
	CHC	0	378	-	277	101	+
GESA	SA	0	378	-	38	340	-
	CGGA	0	378	-	298	80	+
	CSSGA	4	374	-	378	0	+
	CHC	0	378	-	301	77	+
PRSA	SA	0	378	-	27	351	-
	CGGA	10	368	-	288	90	+
	CSSGA	0	378	-	279	99	+
	CHC	1	377	-	277	101	+
SSSA	SA	0.5	377.5	-	48	330	-
	CGGA	33.5	344.5	-	329	49	+
	CSSGA	378	0	+	378	0	+
	CHC	1	377	-	367	11	+
HHSAGA	SA	8	370	-	116	262	~
	CGGA	88	290	-	378	0	+
	CSSGA	377	1	+	378	0	+
	CHC	46	332	-	378	0	+
GA-PSA	SA	7.5	370.5	-	68	310	-
	CGGA	0	378	-	359.5	18.5	+
	CSSGA	158.5	219.5	~	210	168	~
	CHC	52.5	325.5	-	355	23	+
AGA	SA	30.5	347.5	-	81	297	-
	CGGA	237.5	140.5	~	339	39	+
	CSSGA	378	0	+	323	55	+
	CHC	129.5	248.5	~	345	33	+
DCHCSA	SA	206.5	171.5	~	32	346	-
	CGGA	330.5	47.5	+	276	102	+
	CSSGA	378	0	+	306	72	+
	CHC	328.5	49.5	+	325	53	+
SALGeS	SA	287.5	90.5	+	27	352	-
	CGGA	374.5	3.5	+	174	204	~
	CSSGA	378	0	+	192	186	~
	CHC	343.5	34.5	+	160	218	~
GAMSA	SA	277.5	101.5	+	27	351	-
	CGGA	352.5	25.5	+	188	190	~
	CSSGA	378	0	+	214	164	~
	CHC	333.5	44.5	+	199	179	~

TABLE V: GAMSA vs. state-of-the-art EAs (Wilcoxon's test with p -value=0.05 and critical value = 107)

Algorithms	R^+	R^-	Sig. differences?
GAMSA vs CHC	338.5	39.5	+
GAMSA vs VDMGA	286	92	+
GAMSA vs CISS	320	58	+
GAMSA vs Saw-GA	365.5	12.5	+
GAMSA vs vQEA	362.5	15.5	+

to GAMSA) and R^- of the test are specified. We notice that GAMSA obtains statistically better results than those of the other EAs (R^- values are lower than both R^+ ones and critical values).

Table VI, in Appendix A, shows the results of the optimisers when tackling each test problem. Moreover t-test results, comparing GAMSA (the best performing approach) and the other algorithms, show whether there are significant differences for each problem instance. Regarding to these results, we observe that:

- In 19 out of 27 instances, GAMSA reaches the best solution found by any algorithm studied in this section.
- Saw-GA and vQEA are able to reach better results than GAMSA in one instance (Deceptive and Royal Road respectively). CHC and CISS do the same in NKLand(48,12). Moreover, CHC outperforms GAMSA in Maxcut(ising2.5-250_5555) and CISS, in Overlapping Deceptive.
- VDMGA, the most competitive algorithm with regards to GAMSA, attains better results on five instances (Royal Road, HIFF(3,4,false), BQP(bqp500-1), Maxcut(ising2.5-250_5555), and Deceptive instances).

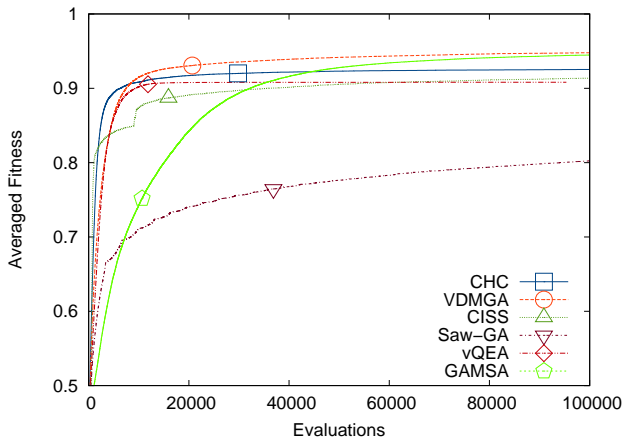


Fig. 14: Convergence graphs of state-of-the-art EAs and GAMSA

As we see in Figure 14, at the initial stages, GAMSA experiences lower convergence speed than most of its competitors, except Saw-GA. However, GAMSA continues improving its results at the intermediate and even at the final stages. This fact allows GAMSA to catch its competitors up and to overtake them at the latter stages. It is important to note that, according

to the convergence graph, though at the end of the execution VDMGA seems to reach similar solutions to GAMSA, the Wilcoxon test showed that GAMSA defeats VDMGA. This can be explained by the fact that the differences achieved by GAMSA are not too great, but it reaches positive differences more frequently. The latter causes the Wilcoxon test to detect significant differences.

In conclusion, this study shows that GAMSA really provides a competitive alternative in the combinatorial binary optimisation field, obtaining promising results on the considered testbed.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we provided an overview of the ways EAs and SA may be combined with each other to obtain HMs-EA/SA. We have organised the approaches found in the literature by proposing a taxonomy based on those introduced by Talbi and Raidl for HMs [3], [4]. Moreover, we have developed, to our knowledge, the first experimental study analysing a large spectrum of HM-EA/SA models from three points of view:

- We have compared the performance of the HM-EA/SA models, individually and by categories, extracting relevant conclusions regarding the categories presented in the presented taxonomy.
- We have performed a synergy test that has identified two HM-EA/SA models that really present synergistic properties.
- We have compared the best performing HM-EA/SA model with the state-of-the-art evolutionary algorithms for binary combinatorial optimisation.

Our study allowed us to draw an interesting conclusion: the hybridisation of EAs and SA becomes a prospective research area for finding more effective search algorithms.

Finally, we should remark that the research line focused on in this paper is indeed worthy of further studies. We will intend to explore three interesting avenues of research. Firstly, *multiobjective and constrained problems* are subject of great interest, existing in the literature a great number of proposals for dealing with this kind of problems based on EAs [82], [83], [84] and SA [85], [86]. Therefore, it is possible to adapt the design of HMs-EA/SA to this type of problems. Secondly, teamwork collaborative HMs-EA/SA are able to take advantage of parallel hardware (multicore processors, clusters, etc.) and software [87] that has become very affordable and widely available nowadays. This clearly favors the *implementation on parallel hardware* of HMs-EA/SA [25], [50], [51] that may lead to improved results due to the speed-up in the search process, which becomes a very appealing option for dealing with large-scale optimisation tasks. Finally, an important issue when combining different algorithms concerns the number of evaluations that each algorithm should consume throughout the run to create the conditions for the appearance of collaborative synergies among all the composing algorithms. Adaptive strategies that identify the best performing technique at each phase of the evolution with a minimum overhead [18], [19], [88] may be used to build *adaptive HMs-EA/SA* with the aim of allowing profitable

TABLE VI: Results of the binary optimisers on each test problem

Pr.	CHC	VDMGA	Saw-GA	CISS	vQEA	GAMSA
1	0.337+	1 -	0.192+	0.130+	0.536-	0.394
2	220 ~	220 ~	219~	210+	219+	220
3	378~	382-	385 -	378~	366+	378
4	0.878+	0.850+	0.853+	0.892+	0.886+	0.899
5	0.899+	0.896+	0.753+	0.915 -	0.894+	0.906
6	0.957+	0.957+	0.943+	0.955+	0.954+	0.959
7	0.937+	0.938+	0.926+	0.937+	0.936+	0.939
8	0.760+	0.765~	0.755+	0.761+	0.747+	0.766
9	0.746-	0.701+	0.713+	0.751 -	0.71+	0.741
10	164+	162+	155+	148+	136+	185
11	170~	176 -	155+	152+	154+	169
12	0.994+	1 ~	0.941+	0.997~	0.999~	1
13	0.990+	0.990+	0.929+	0.991+	0.988+	0.998
14	0.990+	0.992+	0.870+	0.995+	0.996+	0.999
15	0.990+	0.995+	0.831+	0.993+	0.993+	0.998
16	5174~	5176 ~	5159+	5176 ~	5168+	5176
17	117453~	120948 -	51714+	119056~	117402+	119264
18	11548+	11589+	9420+	11583+	11530+	11599
19	49852+	50428+	34427+	50407+	50162+	50465
20	68+	69+	61+	68+	65+	73
21	2623+	2647+	2267+	2681+	2552+	2716
22	1422+	1423+	1385+	1430+	1407+	1435
23	8153+	8168+	7923+	8190+	8102+	8204
24	7526181 -	7522897-	4949495+	7333148+	7249980+	7422326
25	4005+	4003+	4074~	4096 ~	3955+	4081
26	12396+	12398~	12385+	12400 ~	12393+	12400
27	118204 ~	118204 ~	118204 ~	118204 ~	118204 ~	118204

synergies to arise from the adjusted intervention of EAs and SA.

APPENDIX A RESULTS

Table VI and Table VII show the average fitness values obtained by the studied search algorithms. The best results for each problem are presented in boldface. Moreover, we have carried out a statistical analysis with p -value equal to 0.05 to measure the performance differences between GAMSA, the best performing HM-EA/SA, and the other algorithms for each problem separately. This statistical analysis is based on t -test, when normality and heteroscedasticity conditions are satisfied, and $Mann-Whitney$, otherwise [89], [69].

TABLE VII: Results of the HMs-EA/SA on each test problem

Pr.	PRSA	GA-PSA	ARSAGA	GESA	AGA	HHGASA	SSSA	CSA	DHCESA	SALGeS	GAMSA
1	0.366 ~	0.579 -	0.085 +	0.104 +	0.690 -	0.548 +	0.123 +	6.52e-02 +	0.338 ~	0.997 -	0.394
2	209 +	217 +	188 +	201 +	216 +	211 +	214 +	179 +	220 ~	220 ~	220
3	375 +	378 ~	363 +	376 +	370 +	374 +	373 +	365 +	382 -	379 -	378
4	0.860 +	0.883 +	0.826 +	0.875 +	0.893 +	0.893 +	0.875 +	0.813 +	0.88 +	0.895 +	0.899
5	0.768 +	0.864 +	0.671 +	0.777 +	0.894 +	0.896 +	0.845 +	0.635 +	0.896 +	0.9 +	0.906
6	0.933 +	0.955 +	0.916 +	0.932 +	0.953 +	0.953 +	0.952 +	0.912 +	0.957 +	0.958 +	0.959
7	0.921 +	0.936 +	0.906 +	0.917 +	0.936 +	0.935 +	0.932 +	0.903 +	0.938 +	0.939 ~	0.939
8	0.727 +	0.744 +	0.673 +	0.72 +	0.759 +	0.737 +	0.741 +	0.674 +	0.766 ~	0.766 ~	0.766
9	0.713 +	0.721 +	0.675 +	0.71 +	0.714 +	0.709 +	0.718 +	0.692 +	0.743 ~	0.737 ~	0.741
10	151 +	126 +	113 +	120 +	179 ~	128 +	160 +	116 +	187 ~	176 +	185
11	159 +	164 +	124 +	144 +	171 -	152 +	156 +	120 +	178 -	176 -	169
12	0.841 +	0.987 +	0.725 +	0.842 +	0.988 +	0.993 +	0.939 +	0.680 +	0.998 ~	0.989 ~	1
13	0.828 +	0.976 +	0.73 +	0.857 +	0.976 +	0.979 +	0.935 +	0.709 +	0.993 +	0.995 +	0.998
14	0.805 +	0.979 +	0.687 +	0.79 +	0.967 +	0.98 +	0.937 +	0.656 +	0.996 +	0.994 +	0.999
15	0.747 +	0.987 +	0.659 +	0.759 +	0.980 +	0.979 +	0.923 +	0.627 +	0.997 +	0.992 ~	0.998
16	4641 +	5164 +	3771 +	4325 +	5176 ~	5146 +	5152 ~	3378 +	5176 ~	5176 ~	5176
17	59834 +	95978 +	26523 +	46188 +	117034 +	111884 +	80116 +	16728 +	120215 -	121266 -	119264
18	8140 +	11588 ~	5022 +	7537 +	11597 ~	11408 +	10792 +	3958 +	11597 ~	11599 ~	11599
19	34193 +	49759 +	19042 +	27767 +	50311 +	50298 +	44338 +	14305 +	50410 +	50456 +	50465
20	51 +	68 +	34 +	52 +	70 +	67 +	68 +	31 +	70 +	72 ~	73
21	2122 +	2623 +	1564 +	2076 +	2675 +	2564 +	2529 +	1510 +	2667 +	2700 +	2716
22	1366 +	1420 +	1324 +	1366 +	1422 +	1414 +	1412 +	1319 +	1431 +	1433 +	1435
23	7819 +	8150 +	7514 +	7826 +	8125 +	8123 +	8074 +	7463 +	8187 +	8202 ~	8204
24	5153437 +	7285193 +	3230924 +	4975365 +	7369494 +	7275664 +	6546047 +	2812404 +	7545615 -	7563823 -	7422326
25	3915 +	3922 +	3732 +	3620 +	3965 +	3793 +	3975 +	3720 +	4053 +	4052 +	4081
26	12189 +	12094 +	12084 +	11934 +	11994 +	11913 +	12244 +	12090 +	12400 ~	12398 +	12400
27	116566 +	118204 ~	111258 +	112481 +	118204 ~	117156 ~	118204 ~	109247 +	118204 ~	118204 ~	118204

ACKNOWLEDGMENT

This work was supported by Research Projects TIN2008-05854 and P08-TIC-4173.

REFERENCES

- [1] F. Glover and G. Kochenberger, Eds., *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.
- [2] P. Siarry and Z. Michalewicz, Eds., *Advances in Metaheuristics for Hard Optimization*, ser. Natural Computing. Springer, 2008.
- [3] E. Talbi, "A taxonomy of hybrid metaheuristics," *J. Heuristics*, vol. 8, no. 5, pp. 541–564, 2002.
- [4] G. Raidl, "A unified view on hybrid metaheuristics," in *Hybrid Metaheuristics*, F. Almeida, M. B. Aguilera, C. Blum, J. M. Vega, M. P. Pérez, A. Roli, and M. Sampels, Eds., vol. LNCS 4030. Springer, 2006, pp. 1–12.
- [5] C. Blum, "Hybrid Metaheuristics – Guest Editorial," *Computers & Operations Research*, vol. 37, no. 3, pp. 430–431, 2010.
- [6] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by simulated annealing," *Sci.*, vol. 220, no. 4598, pp. 671–680, 1983.
- [7] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons, 1989.
- [8] D. Henderson, S. Jacobson, and A. Jacobson, "The theory and practice of simulated annealing," in *Handbook of Metaheuristics*. Kluwer Academic Publishers Group, 2003, pp. 287–319.
- [9] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: AMOSA," *IEEE Trans. Evol. Comput.*, vol. 12, no. 3, pp. 269–283, 2008.
- [10] P. Salamon, P. Sibani, and R. Frost, *Facts, Conjectures and Improvements for Simulated Annealing*, ser. Monographs on Mathematical Modeling and Computation. SIAM, 2002.
- [11] T. Bäck, D. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. Institute of Physics Publishers, 1997.
- [12] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.
- [13] D. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, 1995.
- [14] H. Beyer and H. Schwefel, "Evolution strategies—a comprehensive introduction," *Nat. Comput.*, vol. 1, no. 1, pp. 3–52, 2002.
- [15] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., 1989.
- [16] C. Grosan and A. Abraham, "Hybrid evolutionary algorithms: methodologies, architectures, and reviews," in *Hybrid Evolutionary Algorithms*, C. Grosan, A. Abraham, and H. Ishibuchi, Eds. Springer, 2007, pp. 1–17.
- [17] P. Preux and E. Talbi, "Towards hybrid evolutionary algorithms," *Int. Trans. Oper. Res.*, vol. 6, no. 6, pp. 557–570, 1999.
- [18] R. Mallipeddi and P. Suganthan, "Ensemble of constraint handling techniques," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 561–579, 2010.
- [19] J. Vrugt, B. Robinson, and J. Hyman, "Self-adaptive multimethod search for global optimization in real-parameter spaces," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 243–259, 2009.
- [20] Z. Lü, F. Glover, and J. Hao, "A hybrid metaheuristic approach to solving the UBQP problem," *Eur. J. Oper. Res.*, vol. 207, no. 3, pp. 1254–1262, 2010.
- [21] C. Cotta and A. Fernandez, "A hybrid GRASP - Evolutionary algorithm approach to Golomb ruler search," in *Parallel Problem Solving from Nature - PPSN VIII*, ser. LNCS, X. Yao, E. Burke, J. Lozano, J. Smith, J. MereloGuervos, J. Bullinaria, J. Rowe, P. Tino, A. Kaban, and H. Schwefel, Eds., vol. 3242, 2004, pp. 481–490.
- [22] M. Lozano and C. García-Martínez, "Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report," *Comput. Oper. Res.*, vol. 37, pp. 481–497, 2010.
- [23] D. Thompson and G. Bilbro, "Sample-sort simulated annealing," *IEEE Trans. Syst., Man, Cybern. B*, vol. 35, no. 3, pp. 625–632, 2005.
- [24] G. Liu, D. Zhou, H. Xu, and C. Mei, "Model optimization of svm for a fermentation soft sensor," *Exp. Syst. App.*, vol. 37, no. 4, pp. 2708 – 2713, 2010.
- [25] Z. Wang, Y. Wong, and M. Rahman, "Development of a parallel optimization method based on genetic simulated annealing algorithm," *Parallel Comput.*, vol. 31, no. 8-9, pp. 839–857, 2005.
- [26] H. Cheng, X. Wang, S. Yang, and M. Huang, "A multipopulation parallel genetic simulated annealing-based QoS routing and wavelength assignment integration algorithm for multicast in optical networks," *App. Soft Comput.*, vol. 9, no. 2, pp. 677–684, 2009.
- [27] L. Tang and X. Wang, "An improved particle swarm optimization algorithm for the hybrid flowshop scheduling to minimize total weighted completion time in process industry," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 6, pp. 1303–1314, 2010.
- [28] I. Obregon and A. Pawlovsky, "A hybrid SA-EA method for finding the maximum number of switching gates in a combinational circuit," *IEICE Electronics Express*, vol. 5, no. 18, pp. 756–761, 2008.
- [29] Y. Xu and R. Qu, "Solving multi-objective multicast routing problems by evolutionary multi-objective simulated annealing algorithms with variable neighbourhoods," *J. Oper. Res. Soc.*, vol. 62, pp. 313–325, 2011.
- [30] B. Firouzi, M. Sadeghi, and T. Niknam, "A new hybrid algorithm based on PSO, SA, and k-means for cluster analysis," *Int. J. Innov. Comput. Inf. Control*, vol. 6, no. 7, pp. 3177–3192, 2010.
- [31] W. Hong, Y. Dong, L. Chen, and C. Lai, "Taiwanese 3G mobile phone demand forecasting by SVR with hybrid evolutionary algorithms," *Exp. Syst. App.*, vol. 37, no. 6, pp. 4452 – 4462, 2010.
- [32] R. M'Hallah, "Minimizing total earliness and tardiness on a single machine using a hybrid heuristic," *Comput. Oper. Res.*, vol. 34, no. 10, pp. 3126 – 3142, 2007.
- [33] A. Tantar, N. Melab, and E. Talbi, "A grid-based genetic algorithm combined with an adaptive simulated annealing for protein structure prediction," *Soft Comput.*, vol. 12, no. 12, pp. 1185–1198, 2008.
- [34] S. Bhandarkar and H. Zhang, "Image segmentation using evolutionary computation," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 1–21, 1999.
- [35] S. Xavier-de-Souza, J. Suykens, J. Vandewalle, and D. Bollé, "Cooperative behavior in coupled simulated annealing processes with variance control," in *Symposium on Nonlinear Theory and its Applications*, 2006, pp. 114–119.
- [36] M. Aydin and V. Yigit, *Parallel Simulated Annealing*, ser. Parallel Metaheuristics: A New Class of Algorithms. Wiley, 2005, pp. 267–288.
- [37] F. Rodriguez-Diaz, C. Garcia-Martinez, and M. Lozano, "A GA-based multiple simulated annealing," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 195–201.
- [38] D. Chen, C. Lee, C. Park, and P. Mendes, "Parallelizing simulated annealing algorithms based on high-performance computer," *J. Global Optim.*, vol. 39, no. 2, pp. 261–289, 2007.
- [39] D. Brown, C. Huntley, and A. Spillane, "A parallel genetic heuristic for the quadratic assignment problem," in *Proceedings of the third international conference on Genetic algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 406–415.
- [40] F. Lin, C. Kao, and C. Hsu, "Applying the genetic approach to simulated annealing in solving some NP-hard problems," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 6, pp. 1752–1767, 1993.
- [41] X. Li and X. Wei, "An improved genetic algorithm-simulated annealing hybrid algorithm for the optimization of multiple reservoirs," *Water Res. Manag.*, vol. 22, pp. 1031–1049, 2007.
- [42] Q. Zhang, J. Wang, C. Jin, and Q. Zeng, "Localization algorithm for wireless sensor network based on genetic simulated annealing algorithm," in *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, 2008, pp. 1–5.
- [43] D. Goldberg, "A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing," *Complex Syst.*, vol. 4, pp. 445–460, 1990.
- [44] P. Yip and Y. Pao, "Combinatorial optimization with use of guided evolutionary simulated annealing," *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 290–295, 1995.
- [45] M. De la Maza and B. Tidor, "Increased flexibility in genetic algorithms: The use of variable boltzmann selective pressure to control propagation," in *Proc. of the ORSA CSTS Conference - Computer Science and Operations Research: New Developments in their Interfaces*, 1992, pp. 425–440.
- [46] B. Li and W. Jiang, "A novel stochastic optimization algorithm," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, pp. 193–198, 2000.
- [47] S. Hwang and R. He, "A hybrid real-parameter genetic algorithm for function optimization," *Adv. Eng. Inform.*, vol. 20, no. 1, pp. 7–21, 2006.
- [48] W. Han and P. Que, "Defect reconstruction of submarine oil pipeline from mfl signals using genetic simulated annealing algorithm," *J. Jpn. Petr. Inst.*, vol. 49, pp. 145–150, 2006.
- [49] D. Adler, "Genetic algorithm and simulated annealing: a marriage proposal," in *Proc. of the IEEE international conference on neural network*, 1993, pp. 1104–1109.

- [50] S. Mahfoud and D. Goldberg, "Parallel recombinative simulated annealing: A genetic algorithm," *Parallel Comput.*, vol. 21, no. 1, pp. 1–28, 1995.
- [51] H. Chen, N. Flann, and D. Watson, "Parallel genetic simulated annealing: a massively parallel simd algorithm," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 2, pp. 126–136, 1998.
- [52] H. Cho, S. Oh, and D. Choi, "A new evolutionary programming approach based on simulated annealing with local cooling schedule," in *Proc. of the Congress on Evolutionary Computation*, 1998, pp. 598–602.
- [53] Z. Yang, Z. Tian, and Z. Yuan, "GSA-based maximum likelihood estimation for threshold vector error correction model," *Comput. Stat. Data Anal.*, vol. 52, no. 1, pp. 109–120, 2007.
- [54] C. García-Martínez and M. Lozano, "Simulated annealing based on local genetic search," in *Proc. of the IEEE Int. Conf. Evolutionary Computation*, 2009, pp. 2569–2576.
- [55] F. Herrera and M. Lozano, "Gradual distributed real-coded genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 4, no. 1, pp. 43–63, 2000.
- [56] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy and design issues," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 474–488, 2005.
- [57] S. Forrest and M. Mitchell, "Relative building block fitness and the building block hypothesis," in *Foundations of Genetic Algorithms 2*, L. Whitley, Ed. Morgan Kaufmann, 1993, pp. 109–126.
- [58] D. Thierens, "Population-based iterated local search: restricting neighborhood search by crossover," in *Proc. of the Genetic and Evolutionary Computation Conf.*, ser. LNCS, K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burk, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, and A. Tyrrel, Eds., vol. 3103. Springer, 2004, pp. 234–245.
- [59] D. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: motivation, analysis, and first results," *Complex Syst.*, vol. 3, pp. 493–530, 1989.
- [60] M. Pelikan, D. Goldberg, and E. Cantú-Paz, "Linkage problem, distribution estimation, and bayesian networks," *Evol. Comput.*, vol. 8, no. 3, pp. 311–340, 2000.
- [61] K. Smith, H. Hoos, and T. Stützle, "Iterated robust tabu search for MAX-SAT," in *Proc. of the Canadian Society for Computational Studies of Intelligence Conf.*, ser. LNCS, J. Carbonell and J. Siekmann, Eds., vol. 2671. Springer, 2003, pp. 129–144.
- [62] S. Kauffman, "Adaptation on rugged fitness landscapes," *Lec. Sci. Complex.*, vol. 1, pp. 527–618, 1989.
- [63] R. Watson and J. Pollack, "Hierarchically consistent test problems for genetic algorithms," in *Proc. of the Congress on Evolutionary Computation*, vol. 2, 1999, p. 1413.
- [64] W. Spears, *Evolutionary Algorithms: The Role of Mutation and Recombination*. Springer, 2000.
- [65] J. Beasley, "Heuristic algorithms for the unconstrained binary quadratic programming problem," The Management School, Imperial College, Tech. Rep., 1998.
- [66] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. Miller and J. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.
- [67] D. Thierens, "Adaptive mutation rate control schemes in genetic algorithms," in *Proc. of the Congress on Evolutionary Computation*, 2002, pp. 980–985.
- [68] S. Garcia, 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," *J. Heuristics*, vol. 15, pp. 617–644, 2008.
- [69] J. Zar, *Biostatistical Analysis*. Prentice Hall, 1999.
- [70] R. Iman and J. Davenport, "Approximations of the critical region of the Friedman statistic," in *Communications in Statistics*, 1980, pp. 571–595.
- [71] S. Holm, "A simple sequentially rejective multiple test procedure," *Scand. J. Stat.*, vol. 6, pp. 65–70, 1979.
- [72] C. Blum, J. Puchinger, G. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," *App. Soft Comput.*, vol. 11, pp. 4135–4151, 2011.
- [73] C. Antonio, "A study on synergy of multiple crossover operators in a hierarchical genetic algorithm applied to structural optimisation," *Structural and Multidisciplinary Optimization*, vol. 38, no. 2, pp. 117–135, 2009.
- [74] F. Peng, K. Tang, G. Chen, and X. Yao, "Population-based algorithm portfolios for numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 782–800, 2010.
- [75] F. Herrera, M. Lozano, and A. Sánchez, "Hybrid crossover operators for real-coded genetic algorithms: An experimental study," *Soft Comput.*, vol. 9, no. 4, pp. 280–298, 2005.
- [76] L. Eshelman and J. Schaffer, "Preventing premature convergence in genetic algorithms by preventing incest," in *Int. Conf. on Genetic Algorithms*, R. Belew and L. Booker, Eds. Morgan Kaufmann, 1991, pp. 115–122.
- [77] D. Whitley, S. Rana, J. Dzubera, and E. Mathias, "Evaluating evolutionary algorithms," *Artif. Intell.*, vol. 85, pp. 245–276, 1996.
- [78] C. Fernandes and A. Rosa, "Self-adjusting the intensity of assortative mating in genetic algorithms," *Soft Comput.*, vol. 12, no. 10, pp. 955–979, 2008.
- [79] F. Gortazar, A. Duarte, M. Laguna, and R. Martí, "Context-independent scatter search for binary problems," Colorado LEEDS School of Business, University of Colorado at Boulder, Tech. Rep., 2008.
- [80] V. Koumousis and C. Katsaras, "A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 19–28, 2006.
- [81] M. Platel, S. Schliebs, and N. Kasabov, "Quantum-inspired evolutionary algorithm: A multimodel EDA," *IEEE Trans. Evol. Comput.*, vol. 13, no. 6, pp. 1218–1232, dec. 2009.
- [82] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.
- [83] C. Coello, G. Lamont, and D. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer-Verlag New York, Inc., 2006.
- [84] C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Comput. Meth. Appl. Mech. Eng.*, vol. 191, no. 11–12, pp. 1245–1287, 2002.
- [85] C. Pedamallu and L. Ozdamar, "Investigating a hybrid simulated annealing and local search algorithm for constrained optimization," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1230–1245, 2008.
- [86] A. Hedar and M. Fukushima, "Derivative-free filter simulated annealing method for constrained continuous global optimization," *J. Global Optim.*, vol. 35, no. 4, pp. 521–549, 2006.
- [87] S. Cahon, N. Melab, and E. Talbi, "ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics," *J. Heuristics*, vol. 10, no. 3, pp. 357–380, 2004.
- [88] A. LaTorre, S. Muelas, and J. Peña, "A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test," *Soft Comput.*, pp. 1–13, 2010.
- [89] D. Sheskin, *The Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 2000.

Francisco J. Rodriguez Biography text here.

PLACE
PHOTO
HERE

Carlos García-Martínez Biography text here.

Manuel Lozano Biography text here.

3. GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times

The journal paper associated to this part is:

- F.J. Rodríguez, C. Blum, C. García-Martínez, M. Lozano, GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times. *Annals of Operations Research*. In press. doi: 10.1007/s10479-012-1164-8.
 - Status: **Accepted**.
 - Impact Factor (JCR 2011): 0.840.
 - Subject Category: Operations Research & Management Science. Ranking 41 / 77 (Q3).

GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times

F.J. Rodriguez · C. Blum · C. García-Martínez ·
M. Lozano

© Springer Science+Business Media, LLC 2012

Abstract In this work, we tackle the problem of scheduling a set of jobs on a set of non-identical parallel machines with the goal of minimising the total weighted completion times. GRASP is a multi-start method that consists of two phases: a solution construction phase, which randomly constructs a greedy solution, and an improvement phase, which uses that solution as an initial starting point. In the last few years, the GRASP methodology has arisen as a prospective metaheuristic approach to find high-quality solutions for several difficult problems in reasonable computational times. With the aim of providing additional results and insights along this line of research, this paper proposes a new GRASP model that combines the basic scheme with two significant elements that have been shown to be very successful in order to improve GRASP performance. These elements are path-relinking and evolutionary path-relinking. The benefits of our proposal in comparison to existing metaheuristics proposed in the literature are experimentally shown.

Keywords Non-identical parallel machine scheduling problem with minimising total weighted completion times · Metaheuristics · GRASP · Path-relinking

1 Introduction

The *non-identical parallel machine scheduling with minimising total weighted completion times* (SNIM-WCT) considers a set J of n independent jobs that have to be processed on a set M of m parallel non-identical machines. Each job $j \in J$ has to be processed by exactly one of the m parallel machines and no machine can process more than one job at the same

F.J. Rodriguez (✉) · M. Lozano
Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain
e-mail: fjrodriguez@decsai.ugr.es

C. Blum
ALBCOM Research Group, Technical University of Catalonia, Barcelona, Spain

C. García-Martínez
Department of Computing and Numerical Analysis, University of Córdoba, Córdoba, Spain

time. A job j is processed on a given machine until completion, i.e., without pre-emption. The processing time of a job j on a machine i is a pre-defined, finite positive number p_{ij} . Furthermore, each job j has a non-negative integer weight w_j . The objective is to schedule the jobs in such a way that the sum of the weighted completion times of the jobs is minimised:

$$\text{Minimise } \sum_{i=1}^n w_j * C_j,$$

where C_j represents the completion time of job j for a given schedule. In the case of the identical parallel machine scheduling problem, each job has the same processing time regardless of the machine employed. By contrast, in the case of non-identical parallel machines, the processing time of a job depends on the selected machine, in two different ways:

- *Uniform parallel machines*: the processing time of job j on machine i is determined as $p_{ij} = p_j/s_i$, where p_j is the processing time of job j and s_i is the speed of machine i .
- *Unrelated parallel machines*: the values of p_{ij} are unrelated.

According to the standard notation proposed by Azizoglu and Kirca (1999a) and Allahverdi et al. (1999), the problems considered in this work are notated in the literature in the following manner: $Q_m/R_m || \sum w_j * C_j$. Bruno et al. (1974) and Lenstra et al. (1977) showed that the problem of minimising the total weighted completion time on two identical machines is NP-hard. So, we can conclude that the more general schemes considered in this work are also NP-Hard.

A mixed integer linear programming formulation for the SNIM-WCT problem is provided for the sake of completeness. Let $x_{ji}^t = 1$ if the job j is processed in the t th position (unit time) on machine i and 0, otherwise. And a variable C_{ji}^t denotes the completion time of the job j scheduled in the t th position on machine i . The model is stated by Azizoglu and Kirca (1999b) as:

$$\min \sum_j \sum_i \sum_t w_j \cdot C_{ji}^t \cdot x_{ji}^t \quad (1)$$

$$\text{subject to: } \sum_i \sum_t x_{ji}^t = 1 \quad \forall j, \quad (2)$$

$$\sum_j x_{ji}^t \leq 1 \quad \forall t, i, \quad (3)$$

$$C_{ji}^t = \sum_{r=1}^n \sum_{s=1}^{t-1} p_{ir} \cdot x_{ri}^s + p_{ij} \quad \forall j, t, i, \quad (4)$$

$$x_{ji}^t \in \{0, 1\} \quad \forall j, t, i. \quad (5)$$

It is important to note that we can find multiple real-life applications of the scheduling on parallel machines problem in different fields. These fields include human resources (Rosenbloom and Goertzen 1987), production management (Buxey 1989; Dodin and Chan 1991; Pendharkar and Rodger 2000), mail facilities (Jarrah et al. 1992), robotised systems (Rochat 1998), sport tournaments (Croce et al. 1999), and chemical processes (Brucker and Hurink 2000).

The *greedy randomised adaptive search procedure* (GRASP) (Feo and Resende 1989; Feo and Resende 1995) is a multi-start two-phase metaheuristic for combinatorial optimisation basically consisting of a solution construction phase and an improvement phase. The

solution construction mechanism builds an initial solution using a greedy randomised procedure, whose randomness allows solutions to be obtained in different areas of the solution space. Each solution is randomly produced step-by-step by uniformly adding one new element from a restricted candidate list to the current solution. The improvement phase then takes the incumbent solution and performs local perturbations in order to get a locally optimal solution with respect to some predefined neighbourhood. Different local search algorithms can be defined according to the chosen neighbourhood. The two-phase process of GRASP is iterative, that is, it continues until a user-defined termination condition—such as the maximum allowed CPU time or the maximum number of iterations—is met. The best solution generated during this iterative process is kept as the overall result. Due to its simplicity, GRASP is often used for real-world applications (Resende and Ribeiro 2003).

Path-relinking (Glover 1996) is an enhancement to the basic GRASP procedure, which consists of exploring trajectories that connect high-quality solutions. Generally, path-relinking may lead to significant improvements in solution quality (Aiex et al. 2003; Aiex et al. 2005; Laguna and Marti 1999; Ribeiro et al. 2002). The use of path-relinking within a GRASP procedure, as an intensification strategy applied to each locally optimal solution, was first proposed by Laguna and Marti (1999). In this work, we propose a *GRASP+PR* algorithm that combines the basic GRASP scheme with path-relinking to deal with the SNIM-WCT problem.

The remainder of this paper is structured as follows. In Sect. 2, we present an overview of the existing research on the SNIM-WCT problem. In Sect. 3, we describe the proposed GRASP+PR. In Sect. 4, we present an empirical study, which is designed to: (1) analyse the influence of our algorithms' parameters and their values, (2) understand the algorithms' behaviour, and (3) compare the results with those of other approaches from the literature. In Sect. 5, we discuss conclusions and further work. Finally, in Appendix a list of abbreviations used in this work is provided (Table 9).

2 Literature review

Scheduling problems deal, in general, with the allocating of resources over time to perform a set of tasks that are parts of certain processes, such as computational and manufacturing processes; see, for example, Blazewicz et al. (2007), for an extensive revision of scheduling problems.

As mentioned before, in this work we focus on the problem of scheduling on parallel machines. Since the introduction of this problem by McNaughton (1959), it has received much attention and many papers have been published in this area. For an in-depth review, interested readers may consult the reviews by Cheng and Sin (1990) and Mokotoff (2001), the chapter devoted to scheduling on parallel processors in Blazewicz et al. (2007) and a recent special issue on computational intelligence in scheduling (Kendall et al. 2010). It is interesting to note that the majority of the studies have concentrated on the case of identical parallel machines (Baker and Merten 1973; Belouadah and Potts 1994; Elmaghraby and Park 1974; Sarin et al. 1988), despite the fact that non-identical parallel machine schedules have more practical relevance. In what follows, for the sake of conciseness, we mainly focus on the non-identical parallel machine problem considering the total weighted completion time criterion.

The research efforts to deal with the SNIM-WCT problem have focused on three main research lines: exact procedures, approximation algorithms through solving relaxations of the problem, and metaheuristic procedures. Concerning exact procedures, since the considered problem is NP-hard, there are few works that present exact algorithms to solve the

SNIM-WCT problem. Azizoglu and Kirca (1999a) designed a branch and bound algorithm to solve the $Q||w_jC_j$ problem. Computational results indicate that the proposed branch-and-bound algorithm is capable of solving instances with up to two machines and 25 jobs or three machines and 20 jobs, within a reasonable time. The same authors (Azizoglu and Kirca 1999b) developed another branch and bound algorithm for the $R||w_jC_j$ problem using a depth first search strategy. Computational experiments are performed with randomly generated instances of the same maximal size as mentioned above. Finally, Chen and Powell (1999) proposed a decomposition approach for solving the SNIM-WCT problems exactly. The decomposition approach first formulates these problems by means of integer programming, and then reformulates the integer programs as a set partitioning problem strategy. The proposed algorithm can solve instances with up to 20 machines and 100 jobs within a reasonable time.

Exact algorithms are generally limited to problem instances of moderate size, due to the exponential increase in CPU time and memory when the problem size increases. Therefore, in practice, approximate algorithms are necessary to find (not necessarily optimal) solutions to the considered problem. In the first place, we can find approximate algorithms that, in general, try to find optimal solutions to relaxations of the problem, such as linear relaxations and convex quadratic programming relaxations (Hall et al. 1997; Hall et al. 1996; Phillips et al. 1997; Schulz and Skutella 1997; Schulz and Skutella 2002; Skutella 2001). In Li and Yang (2009), we find a detailed review of this kind of approximate algorithms and models for the non-identical parallel machines scheduling problem. Secondly, there are different approaches based on metaheuristics. Weng et al. (2001), for example, reviewed six existing heuristics and provided a new *heuristic algorithm* for the SNIM-WCT problem. The computational experiments performed considered four, six, eight, 10 and 12 machines, as well as 40, 60, 80, 100 and 120 jobs. Later, Vredeveld and Hurkens (2002) presented two types of neighbourhood functions. The first function is called the *jump neighbourhood*. It consists of selecting a job j and a machine i so that job j is not scheduled on machine i . Then job j is moved to machine i . The second one is called *swap neighbourhood*. For this neighbourhood, two jobs j and k must be selected and assigned to different machines. The corresponding neighbouring solution is obtained by interchanging the machine allocations of the two selected jobs. These two neighbourhood functions are applied in two metaheuristics based on a local search:

- *Multistart iterative method*. This procedure iteratively applies a first improvement local search to randomly generated solutions.
- *Tabu search*. Just like any other tabu search method, this algorithm improves the previously defined local search by storing some information about solutions visited in past iterations in a so-called tabu list in order to avoid cycling.

Zaidi et al. (2010) presented four new metaheuristics to tackle the SNIM-WCT problem:

- A *generational genetic algorithm* that starts by initialising chromosomes randomly and applies selection by linear ranking. Selected individuals produce offspring by means of uniform crossover, and, finally, mutation is applied to the produced offspring.
- A *differential evolution algorithm*. This evolutionary algorithm generates trial vectors from an initial population of solutions in vector form. At each step, differential evolution mutates existing vectors by adding weighted random vector differentials to them. If the fitness of the trial vector is better than that of the target vector, the trial vector replaces the target vector in the next generation.

– Finally, Zaidi et al. (2010) provide a *variable neighbourhood descent* method which is incorporated into the two evolutionary algorithms described above. The two resulting *hybrid approaches* explore the neighbourhood of the high-quality solutions obtained by the genetic algorithm or differential evolution by means of variable neighbourhood descent.

Recently, Lin et al. (2011b) presented another *genetic algorithm* approach to deal with unrelated parallel machines scheduling using three different performance criteria. In particular, the proposed approach initialises the population adding some solutions generated by heuristics methods. The remaining ones are generated randomly to provide enough diversity. Roulette wheel selection is used to choose a new population with respect to a fitness-proportional probability distribution. The crossover and mutation schemes are those proposed by Cheng et al. (1995). Elitism is considered by removing two chromosomes and adding the best two previous chromosomes to the new generation if they are not selected through the roulette-wheel-selection process. The experimental study performed compares the proposed genetic algorithm with a set of heuristics. Results show that the proposed algorithm outperforms the competing heuristics.

Finally, it is important to note that metaheuristics have also been widely used to solve the problem of scheduling on unrelated parallel machines considering other performance criteria. Tabu search, simulated annealing and genetic algorithms were applied in Glass et al. (1994) to deal with the problem of minimising the maximum makespan in the context of unrelated parallel machines. The same problem has been tackled recently by means of an iterated greedy metaheuristic (Fanjul-Peyro and Ruiz 2010) and an ant colony optimisation algorithm, in which sequence-dependent setup times are considered (Arnaout et al. 2010). Iterated greedy and a fast simple local search algorithm have also been employed to solve the unrelated parallel machines scheduling problem with the goal of makespan minimisation, considering optional machines and job selection (Fanjul-Peyro and Ruiz 2012). In addition, in Fanjul-Peyro and Ruiz (2011), a set of metaheuristics based on a size-reduction of the unrelated parallel machines problem with the goal of makespan minimisation has been studied in order to produce solutions of very good quality in a short amount of time.

Chen and Chen (2009) presented a hybrid metaheuristic combining tabu search and variable neighbourhood descent to minimise the total weighted tardiness concerning unrelated parallel machines. The same problem but considering the total tardiness criteria and sequence- and machine-dependent setup times in the presence of due date constraints was tackled with an iterated greedy algorithm in Lin et al. (2011a). In Feng and Lau (2008), a new metaheuristic (squeaky wheel optimisation) was proposed to solve the parallel machines scheduling problems with non-uniform sequence-dependent setup times, job processing times, due dates, and earliness/tardiness penalty weights. The cases considering uniform and identical parallel machines have also been tackled with metaheuristics. Anghinolfi and Paolucci (2007) tackled the problem of minimising the total weighted tardiness of uniform parallel machines by means of a hybrid metaheuristic based on simulated annealing, tabu search and variable neighbourhood search. Finally, tabu search was proposed by Waligora (2009) to minimise the maximum makespan in the context of identical parallel machines.

3 GRASP+PR for the SNIM-WCT problem

In this section, we develop a GRASP+PR model to tackle the SNIM-WCT problem. Firstly, in Sect. 3.1, we provide a general overview of basic GRASP and methods for its enhancement. Then, in Sect. 3.2, we present the proposed GRASP+PR for the SNIM-WCT problem. Finally, in Sects. 3.3, 3.4, 3.5, 3.6 and 3.7, we provide detailed descriptions of the different

algorithmic components of the proposal: the greedy randomised construction of solutions, the improvement procedure, path-relinking, the update of the set of elite solutions, and evolutionary path-relinking, respectively.

3.1 Basic GRASP and enhancements

Basic GRASP is an iterative process in which each iteration consists of two phases: solution construction and solution improvement. The construction phase generates a feasible solution, whose neighbourhood is explored by the improvement procedure. The best solution obtained over all iterations is returned as the result.

In the solution construction phase, GRASP makes use of a greedy function in a randomised way. At each step, the current partial solution may be extended by adding one component from a set E of opportunely defined solution components. Greediness is used to choose a set of the most promising solution components from E . In GRASP, this set is known as the *restricted candidate list* (RCL). In order to generate this set, each solution component $e \in E$ is evaluated by means of a greedy function $g(\cdot)$, where $g(e)$ denotes the greedy value of e . In a value-based construction, the restricted candidate list is then defined as follows:

$$\{e \in E : g_* \leq g(e) \leq g_* + \alpha \cdot (g^* - g_*)\},$$

where $g_* = \min\{g(e) : e \in E\}$, $g^* = \max\{g(e) : e \in E\}$, and α is a parameter satisfying $0 \leq \alpha \leq 1$. After the generation of the candidate list, one solution component from this set is chosen uniformly at random.

The path-relinking method consists of exploring trajectories that connect an initial solution and a guiding solution. This is done by iteratively introducing attributes of the guiding solution into the initial solution. At each step, all moves that incorporate attributes of the guiding solution are analysed and the best of these moves is chosen. The general idea is that a trajectory connecting two high-quality solutions may contain even better solutions. For the incorporation of path-relinking into GRASP, the basic GRASP scheme is extended with a set P of so-called elite solutions. Note that P is used to store high-quality solutions found during the search process. Path-relinking is then generally applied to the solution generated in the current iteration and one solution from P . In *forward path-relinking* the initial solution is chosen as the worst of these two solutions. On the contrary, *backward path-relinking* interchanges the roles of both solutions, assigning the role of the initial solution to the better solution of the two. The main advantage of the latter approach over forward path-relinking comes from the fact that, in general, there are more high-quality solutions near the better solution. Experiments performed by Aiex et al. (2005) and Resende and Ribero (2003) have shown that backward path-relinking usually outperforms forward path-relinking.

Resende and Werneck (2004) introduced *evolutionary path-relinking* as a post-processing phase for GRASP. The basis of evolutionary path-relinking is a population of solutions. The set P of elite solutions generated by GRASP is taken as the initial population. The population management of evolutionary path-relinking is the same as that employed by steady state evolutionary algorithms, therefore, the size of the population does not change over time. For the generation of the next population, path-relinking is applied to a set of pairs of solutions chosen from the current population. Each resulting offspring solution is tested for membership in the population of the next generation.

3.2 GRASP+PR

The GRASP+PR algorithm proposed to tackle the SNIM-WCT problem combines the basic GRASP scheme with backward path-relinking as an intensification method, performed

```

Input:  $T_{max}, freqVND, freqEvol, maxElite$ 
Output:  $s^*$ 
1  $s^* \leftarrow NULL;$ 
2  $iter \leftarrow 0;$ 
3  $P \leftarrow \emptyset;$ 
4 while computation time limit  $T_{max}$  not reached do
5    $s \leftarrow GreedyRandomizedSolution();$ 
6   if  $iter \% freqVND == 0$  then
7      $s \leftarrow VariableNeighborhoodDescent(s);$ 
8   else
9      $s \leftarrow SimpleLocalSearch(s);$ 
10  end
11  if  $|P| > 2$  then  $s \leftarrow PathRelinking(s, P)$  end;
12   $UpdateEliteSet(s, P);$ 
13  if  $s^* == NULL$  then
14     $s^* \leftarrow s;$ 
15  else
16    if  $f(s) < f(s^*)$  then  $s^* \leftarrow s$  end;
17  end
18   $iter \leftarrow iter + 1;$ 
19  if  $iter \% freqEvol == 0$  then  $EvolutionaryPathRelinking(P)$  end;
20 end
21  $EvolutionaryPathRelinking(P);$ 

```

Fig. 1 GRASP+PR

throughout the execution of the algorithm. Moreover, evolutionary path-relinking is applied both during the execution of GRASP, every pre-defined number of iterations, and as a post-optimisation method after the termination of GRASP. The general scheme of the proposed GRASP+PR algorithm is shown in Fig. 1.

The GRASP+PR procedure requires the input of values for four parameters. T_{max} denotes the computation time limit, $freqVND$ determines the frequency at which variable neighbourhood descent is applied to the incumbent solution s , $freqEvol$ determines the frequency of applying evolutionary path-relinking as an intensification method at the end of an iteration, and $maxElite$ is the maximum number of solutions in P , the set of elite solutions. At the start of the algorithm, P is initialised to the empty set, and the best found solution s^* points to $NULL$. Moreover, the iteration counter $iter$ is initialised to zero.

At each iteration, the following actions are taken. First, a solution is constructed by function $GreedyRandomizedSolution()$ (see Sect. 3.3). Second, depending on variable $freqVND$, either a simple local search procedure or a more sophisticated variable neighbourhood descent method is applied to the incumbent solution s . Both functions— $SimpleLocalSearch(s)$ and $VariableNeighborhoodDescent(s)$ —are detailed in Sect. 3.4. Next, in case the set P of elite solutions already contains more than two solutions, backward path-relinking is applied in function $PathRelinking(s, P)$ (see Sect. 3.5). After that, procedure $UpdateEliteSet(s, P)$ is responsible for updating the current set P of elite solutions with solution s generated by path relinking (see Sect. 3.6). At the end of each iteration, depending on $freqEvol$, evolutionary path-relinking is applied in function $EvolutionaryPathRelinking(P)$ to the set P of

elite solutions (see Sect. 3.7). The algorithm ends by applying the function `Evolutionary-PathRelinking(P)` once more, and by returning the best solution found during the search process (s^*). This way of using evolutionary path-relinking was proposed in Aiex et al. (2005), Resende et al. (2010), Resende and Werneck (2004). A detailed description of all the above-mentioned functions is provided in the following sections.

3.3 Greedy randomised solution construction

In this section, a description of function `GreedyRandomizedSolution()` (see line 5 of the algorithm in Fig. 1) is given. For the purpose of constructing solutions we have used the solution construction mechanism of the heuristic proposed in (Weng et al. 2001). This choice is motivated by the fact that this heuristic was reported as obtaining the best solutions from among several heuristics studied in the same work.

At this point it is important to clarify the structure of a feasible solution. Remember that a solution is obtained by assigning each job $j \in J$ (where J is the set of n jobs) to exactly one machine $i \in M$ (where M is the set of m machines).

Accordingly, the construction of a solution simply consists of assigning each job to exactly one machine. In our case this is done as follows. First, the *processing time of a machine* $i \in M$ is denoted by t_i . These machine processing times are initialised to zero at the start of a solution construction. At each solution construction step, set $\bar{J} \subseteq J$ denotes the set of so-far unassigned jobs. Each of these jobs may be assigned to any of the m machines. The greedy value $g(i, j)$ for the assignment of a job $j \in \bar{J}$ to a machine $i \in M$ is determined as follows:

$$g(i, j) = \frac{t_i + p_{ij}}{w_j}$$

At each construction step, the restricted candidate list *RCL* is formed by the possible assignments that have the best greedy values. The assignment that is finally chosen at a certain construction step is selected uniformly at random from *RCL*. The procedure to construct a greedy randomised solution ends when all jobs are assigned to machines. At this point a complete solution s is returned. The detailed pseudocode of the procedure is shown in Fig. 2.

3.4 Improvement methods

The improvement phase of the proposed GRASP+PR algorithm consists of functions `VariableNeighborhoodDescent(s)` and `SimpleLocalSearch(s)` (see lines 7 and 9 of the algorithm in Fig. 1). The aim of both functions is to find a better solution close to the solutions generated by the greedy randomised solution construction. The variable neighbourhood descent method executed in `VariableNeighborhood-Descent(s)` function was proposed by Zaidi et al. (2010). It uses two different neighbourhoods. The first neighbourhood is defined via *insertion moves*, where each possible move is generated by randomly selecting one job and assigning this job to a different (arbitrarily chosen) machine. The second neighbourhood works with *swap moves*. Each swap move consists of randomly selecting two jobs from two different machines and swapping the jobs. As it is more time-consuming than the simple local search method outlined below, the variable neighbourhood descent procedure is only used every *freqVND* iterations. In the remaining iterations, function `SimpleLocalSearch(s)` executes a simple local search method based on insertion moves.

<p>Input: Output: s</p> <pre> 1 $\bar{J} \leftarrow J;$ 2 Let s be an empty solution; 3 while $\bar{J} \neq \emptyset$ do 4 $g_* \leftarrow \min\{g(i, j) : i \in M, j \in \bar{J}\};$ 5 $g^* \leftarrow \max\{g(i, j) : i \in M, j \in \bar{J}\};$ 6 $RCL \leftarrow \{(i, j) : g_* \leq g(i, j) \leq g_* + \alpha \cdot (g^* - g_*)\};$ 7 $(k, l) \leftarrow$ Choose randomly from RCL; 8 Assign job l to machine k in solution s; 9 $\bar{J} \leftarrow \bar{J} \setminus \{l\};$ 10 end </pre>

Fig. 2 Greedy randomised solution construction

3.5 Path-relinking

As mentioned before, path-relinking is an intensification strategy that generates new solutions by exploring trajectories that connect high-quality solutions. These trajectories are generally produced by iteratively introducing features from a so-called guiding solution into an initial solution. This iterative process stops once the trajectory has reached the guiding solution. In this work we chose the option of assigning the role of guiding solution to the worse solution of the two solutions chosen for path-relinking. This scheme is known as backward path-relinking.

In the following we give a description of the path-relinking function $\text{Path-Relinking}(s, P)$ from line 11 of the algorithm in Fig. 1. First, in addition to solution s , which is an input parameter of function $\text{PathRelinking}(s, P)$, a second solution s' must be chosen in order to be able to apply path-relinking. This solution is chosen from the set of elite solutions P . For this purpose a distance measure $\text{Dist}(s_1, s_2)$ is introduced which gives a measure of the distance between two solutions s_1 and s_2 . More specifically, $\text{Dist}(s_1, s_2)$ is defined as the number of jobs that are not assigned to the same machine in s_1 and s_2 . Then, $s' \in P$ is chosen as the solution with the largest distance from s . Moreover, we define set $\Delta(s_1, s_2) \subseteq J$ as the set of jobs that are not assigned to the same machine in s_1 and s_2 . Henceforth we refer to the jobs in $\Delta(s_1, s_2)$ as solution features. Next, the worse of the two solutions s and s' is denoted by s_g —that is, the guiding solution—while the remaining solution is denoted by s_c , the current solution. At this moment, an iterative process is initiated which selects at each step exactly one job $j^* \in \Delta(s_c, s_g)$ and assigns this job in solution s_c to machine $\text{mach}(j^*, s_g)$, where $\text{mach}(j^*, s_g)$ refers to the machine to which job j^* is assigned in solution s_g . The resulting solution is denoted by $\text{mov}(s_c, s_g, j^*)$. Job j^* is chosen as follows:

$$j^* = \text{argmin}\{f(\text{mov}(s_c, s_g, j)) : j \in \Delta(s_c, s_g)\}$$

The complete pseudocode of the designed path-relinking procedure can be seen in Fig. 3.

3.6 Management of the set of elite solutions

The application of the path-relinking function $\text{PathRelinking}(s, P)$ (see the algorithm in Fig. 1) produces a solution s as output. This solution is considered as a candidate to be

```

Input:  $s, P$ 
Output:  $s_{pr}$ 
1  $s' \leftarrow \operatorname{argmax}\{\operatorname{Dist}(s, \hat{s}) : \hat{s} \in P\};$ 
2  $s_c \leftarrow \operatorname{argmin}\{f(s), f(s')\};$ 
3  $s_g \leftarrow \operatorname{argmax}\{f(s), f(s')\};$ 
4  $s \leftarrow s_g;$ 
5 while  $|\Delta(s_c, s_g)| > 1$  do
6    $j^* \leftarrow \operatorname{argmin}\{f(\operatorname{mov}(s_c, s_g, j)) : j \in \Delta(s_c, s_g)\};$ 
7    $s_c \leftarrow \operatorname{mov}(s_c, s_g, j^*);$ 
8   if  $f(s_c) < f(s)$  then
9      $s \leftarrow s_c;$ 
10  end
11 end

```

Fig. 3 Path-relinking

inserted into the set of elite solutions P . If the number of solutions in P is less than its maximum size (maxElite), s is simply inserted into P . By contrast, if the elite set has reached its maximum size, s replaces the solution \hat{s} which is most similar to s of all the solutions $s' \in P$ with a worse objective function value than s . For this purpose the distance measure $\operatorname{Dist}(\cdot, \cdot)$, as introduced in Sect. 3.5, is used. The procedure described in this section is implemented by function $\operatorname{UpdateEliteSet}(s, P)$ from line 12 of the algorithm in Fig. 1.

3.7 Evolutionary path-relinking

Evolutionary path-relinking (see function $\operatorname{EvolutionaryPathReLinking}(P)$ from lines 19 and 22 of the algorithm in Fig. 1) applies path-relinking to all pairs of solutions contained in the set P of elite solutions. Each application of the path-relinking function produces a solution s_{pr} which is then considered for inclusion in P . For this purpose, function $\operatorname{UpdateEliteSet}(\cdot, \cdot)$ as described in Sect. 3.6 is used. The complete pseudocode of the evolutionary path-relinking procedure is shown in Fig. 4. Note that function $\operatorname{PathReLinking}(s, s')$ implements exactly the same procedures as outlined in Sect. 3.5, except that the two solutions for the application of path-relinking are already given as input parameters.

4 Experimental comparison

In this section, we describe the experiments carried out in order to study the behaviour of the GRASP+PR model presented in the previous section. Firstly, we detail the experimental setup and the statistical methods applied (Sect. 4.1). Then we present and analyse the results obtained from different experiments carried out with the proposed GRASP+PR. Our aims are: (1) to analyse the influence of parameter α (see Fig. 3) on the behaviour of the algorithm (Sect. 4.2), (2) to carry out a comparison between our proposal and existing metaheuristics from the literature (Sect. 4.3), and (3) to investigate the way path-relinking affects GRASP+PR performance (Sect. 4.4).

```

Input:  $P$ 
Output:
1  $P' \leftarrow P;$ 
2 foreach  $s \in P$  do
3   foreach  $s' \in P$  such that  $f(s') > f(s)$  do
4      $s_{pr} \leftarrow \text{PathRelinking}(s, s');$ 
5      $\text{UpdateEliteSet}(s_{pr}, P');$ 
6   end
7 end
8  $P \leftarrow P';$ 

```

Fig. 4 Evolutionary path-relinking

4.1 Experimental setup

Our own algorithm (GRASP+PR) as well as all competitor algorithms have been implemented in C++ and the source code has been compiled with gcc 4.4.1. All experiments were conducted on a computer with a 3.2 GHz Intel Xeon processor with 2 GB of RAM running Fedora Linux V11. As mentioned before, in this work we consider problem instances with both uniform and unrelated parallel machines. We considered problem instances from 11 different combinations of the number of jobs (n) and the number of machines (m). These 11 instance types are shown in the first two columns of Table 1. Moreover, the same table shows—in the 3rd column—the maximum CPU time allotted for each instance type ($2n$ seconds). For each of the 11 instance types ten problem instances were randomly generated, which is a common choice in recent works dealing with this or related problems (Fanjul-Peyro and Ruiz 2010; Zaidi et al. 2010), in the following manner:

- Concerning the SNIM-WCT problem with uniform parallel machines, the speed of the m machines and the weights of the n jobs were chosen uniformly at random from $\{1, \dots, 10\}$. In addition, the job processing times (p_j) were chosen uniformly at random from $\{1, \dots, 100\}$.
- Concerning the SNIM-WCT problem with unrelated parallel machines, the weights of the n jobs were selected uniformly at random from $\{1, \dots, 10\}$ and the processing time of job j on machine i (p_{ij} , $i = 1, \dots, n$ and $j = 1, \dots, m$) was chosen uniformly at random from $\{1, \dots, 100\}$.

In total, this amounts to 110 problem instances for each problem type (that is, uniform or parallel machines).

Non-parametric tests (Garcia et al. 2008) have been used to compare the results of the different optimisation algorithms under consideration. The only condition to be fulfilled for the use of non-parametric tests is that the algorithms to be compared should have been tested under the same conditions (that is, the same set of problem instances, the same stopping conditions, the same number of runs, etc.). Specifically, we have considered two alternative methods based on non-parametric tests to analyse the experimental results:

- The first method is the application of Iman and Davenport’s test (1980) with Holm’s method (1979) as a post-hoc procedure. The first test may be used to see whether there are significant statistical differences among the algorithms in a certain set of tested algorithms. If such differences are detected, then Holm’s method is employed to compare the best algorithm (that is, the control algorithm) with the remaining ones.

Table 1 11 instance types considered concerning the SNIM-WCT problem with uniform and unrelated parallel machines. The last table column provides the maximum CPU time limit for each instance type (in seconds)

Number of jobs (n)	Number of machines (m)	Time limit (s)
20	5	40
	10	40
50	5	100
	10	100
	20	100
100	5	200
	10	200
	20	200
200	5	400
	10	400
	20	400

Table 2 Parameters values

Parameter	Value
Elite set size (<i>maxElite</i>)	10
Frequency evolutionary path-relinking (<i>freqEvol</i>)	20
Frequency VND (<i>freqVND</i>)	20

– The second method is the utilisation of Wilcoxon’s matched-pairs signed-ranks test (1945). With this test, the results of two algorithms may be directly compared.

4.2 Study of parameter α

In this section, we present a study of the influence of parameter α which is associated with the generation of the restricted candidate lists. Note that with rather low values of α the resulting restricted candidate lists are rather small, which means that the generated solutions are rather close to the greedy solution. On the other hand, with rather high values of α , the size of the restricted candidate lists is rather large, and the solution construction is more biased towards exploration.

In order to study the influence of α , we applied GRASP+PR with five different settings ($\alpha = \{0.02, 0.05, 0.1, 0.2, 0.5\}$) once to each of the 220 problem instances, considering the SNIM-WCT problem with uniform and with unrelated machines. The values for the remaining three parameters were fixed after tuning by hand as shown in Table 2.

Then for each of the 22 problem types—11 concerning uniform parallel machines, and 11 considering unrelated parallel machines—the average result for each of the five algorithm versions was computed. Based on these average results, a ranking of the five algorithm versions was computed for each problem type. More specifically, the algorithm version with the best average result for a certain problem type was assigned rank 1, while the algorithm version with the worst average result obtained rank 5. Finally, we computed for each algorithm version the average rank (over the ranks for the 22 problem types). These average ranks are shown in the form of bar plots for all five algorithm versions in Fig. 5. The algorithm version with the smallest value for α —that is, $\alpha = 0.02$ —has obtained the best average rank. This means that solutions close to the greedy solution are a good starting point for the exploration

Fig. 5 Average rankings obtained by five versions of GRASP+PR, as defined by five different settings for parameter α

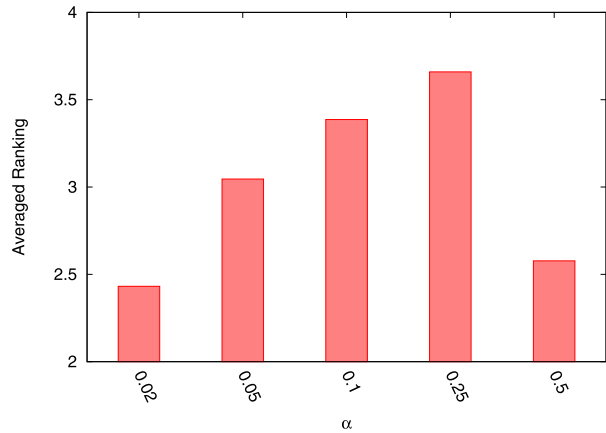


Table 3 Comparison, using Holm's test, of four GRASP+PR versions ($\alpha \in \{0.05, 0.1, 0.2, 0.5\}$) with the control algorithm using $\alpha = 0.02$

α (alg. version)	z	p -value	$alpha/i$	Diff.?
0.25	2.574349	0.010043	0.0125	yes
0.1	2.002271	0.045256	0.016667	no
0.05	1.287174	0.198033	0.025	no
0.5	0.095346	0.92404	0.05	no

of promising areas of the search space. Therefore, the setting of $\alpha = 0.02$ will be used for all experiments presented in subsequent sections.

In order to study the statistical significance of these results, we applied Iman and Davenport's test (the level of significance considered was 0.05). The results show the existence of significant differences among the five different algorithm versions: the statistical value 2.81 is greater than the critical value of 2.48. With regard to this result, we have compared the control algorithm—GRASP+PR with $\alpha = 0.02$, which is the best ranked algorithm—with the other four GRASP+PR versions, by means of Holm's test. Tables 3 provides all corresponding values (z , p -value, and $alpha/i$) concerning a significance level of 0.05. The last column indicates whether Holm's test finds statistical differences between the control algorithm and the corresponding algorithm version. If the corresponding p -value is smaller than the adjusted alpha, the test detects significant differences between them, which means that the control algorithm is better. Although Holm's test does not detect significant differences between version using $\alpha = 0.02$ and the remainder (except that using $\alpha = 0.25$), we will use $\alpha = 0.02$ because it obtained the best averaged ranking.

4.3 Comparison to existing approaches

In this section, we compare GRASP+PR to different approaches found in the literature for tackling the SNIM-WCT problem. More specifically, we considered the following approaches (see also Sect. 2):

- Iterative multistart method (**MultiS**) (Vredeveld and Hurkens 2002).
- Tabu search (**Tabu**) (Vredeveld and Hurkens 2002).

Table 4 Results of the studied algorithms averaged over the 10 instances of each of the 11 instance types. These results concern the case of uniform parallel machines

n	m	GRASP+PR	GA+H	DE	DE+VND	GA	GA+VND	Tabu	MultiS
200	20	28166	30230.3	44700	34236.6	44410.9	29724.3	28183	28178
200	10	53613.4	57721.5	75544.7	59772.6	76638.3	53620	53819.7	53664.3
200	5	121694.1	128153.2	140363.2	124379.4	146625.9	121671	122969.7	122003
100	20	6266.4	6684.3	8498.3	6846.5	8367.6	6281.4	6278.6	6272.8
100	10	9814	10488.1	11841.1	9841.2	12426.2	10151.2	9835.7	9819.6
100	5	18252.1	19382.3	19643.7	18408.7	21080.7	18662	18267.3	18252.3
50	20	2263.2	2378.5	3214.8	2279.4	3506.2	2280	2278.2	2260.7
50	10	4020.9	4250.7	4983.9	4046.3	5615.5	4047.8	4046.4	4031.1
50	5	7699.3	7966.8	8149.3	7712.1	8948.6	7708.3	7721.2	7698.9
20	10	1164.2	1223.5	1220.6	1169.8	1415.5	1172.2	1170.5	1164.2
20	5	1932.3	1988.7	1934.1	1933.8	2105	1942.5	1936.3	1932.3

- Genetic algorithm (**GA**) (Zaidi et al. 2010).
- Differential evolution (**DE**) (Zaidi et al. 2010).
- Hybrid approaches with variable neighbourhood descent (**GA+VND** and **DE+VND**) (Zaidi et al. 2010).
- A second version of a genetic algorithm (**GA+H**) (Lin et al. 2011b).

All these approaches were re-implemented in C++ using the same data structures as GRASP+PR. The parameter values used for each considered algorithm are those recommended in the original works. In order to assure a fair comparison, each algorithm was applied under the same conditions as GRASP+PR; that is, each algorithm was applied exactly once to each of the 220 problem instances. Moreover, the same CPU time limits were used as with GRASP+PR (see Table 1).

4.3.1 Case 1: uniform parallel machines

Table 4 presents the results of all algorithms averaged over the 10 instances for each of the 11 problem types. The best result for each instance type is indicated in bold. In addition, we have undertaken a comparative analysis by means of Wilcoxon's test. Table 5 summarises the results of this test by providing the values of R^+ (associated with GRASP+PR) and R^- (associated with the corresponding competitor). The last table column indicates whether Wilcoxon's test found statistical differences between GRASP+PR and the corresponding competitor algorithm. If $\min(R^+, R^-)$ is less than or equal to the critical value, the test has detected significant differences between the algorithms, which means that one algorithm outperforms the other. If this is the case and if $R^- = \min(R^+, R^-)$, then GRASP+PR is statistically better than the corresponding competitor algorithm.

The results of the algorithms and the additional statistical analysis allows us to make the following observations:

- The proposed GRASP+PR statistically outperforms all competing algorithms (see Table 5).
- Concerning the results shown in Table 4, GRASP+PR obtains the best average result in 8 out of 11 instances types.

Table 5 GRASP+PR versus competitors using Wilcoxon's test (significance level $\alpha = 0.05$, critical value = 10; significance level $\alpha = 0.1$, critical value = 13) concerning the case of uniform parallel machines

Competitor	$R+$	$R-$	Diff.? $\alpha = 0.05/0.1$
GA+H	66	0	yes/yes
DE	66	0	yes/yes
DE+VND	66	0	yes/yes
GA	66	0	yes/yes
GA+VND	64	2	yes/yes
Tabu	66	0	yes/yes
MultiS	53.5	12.5	no/yes

Table 6 Results of the studied algorithms averaged over the 10 instances of each of the 11 instance types. These results concern the case of unrelated parallel machines

n	m	GRASP+PR	GA+H	DE	DE+VND	GA	GA+VND	Tabu	MultiS
200	20	17002	17332.6	156913.1	89286.5	149463.9	17029.8	17066.8	17053
200	10	52933.2	53739.9	253703.8	73739	256803.9	52986.6	53039.2	52986.3
200	5	182120	183357.4	459479.4	182161.4	483989.6	182201.7	182710.9	182189.1
100	20	5602.5	5877.6	41668	9087.1	42760.8	9099.1	5657.1	5621.1
100	10	14921.4	15171.1	57545.7	14965.6	66673.8	19645.2	14992.5	14935.8
100	5	45961.5	46406.9	90685.7	46012.2	116467.4	46012.2	46014.5	45967.9
50	20	1833.4	1923.1	10295.8	1855.9	12807.6	1855.9	1858.5	1834.1
50	10	4611.2	4726	11581.2	4626.5	17043	4626.5	4654.8	4611.2
50	5	12625.1	12781.5	14407.9	12634.4	28506.6	12665.2	12642.6	12625.1
20	10	1298.9	1332.2	1315.2	1300.8	3881.5	1303	1298.9	1298.9
20	5	2512.2	2569.7	2512.2	2512.2	4641.9	2529.8	2512.2	2512.2

- MultiS and GA+VND are the best competitors of GRASP+PR. Especially for smaller problem instances, MultiS is able to match the results of GRASP+PR.
- The differences between the performance of DE and GA and their respective hybridizations with VND allow us to conclude the importance of local search procedures in order to reach high-quality solutions.

4.3.2 Case 2: unrelated parallel machines

The same experimental evaluation was repeated concerning the 110 problem instances for the case of unrelated parallel machines. Table 6 presents the results of all algorithms averaged over the 10 instances for each of the 11 problem types. The best result for each instance type is again indicated in bold. As in the case of uniform parallel machines, we have undertaken a comparative analysis by means of Wilcoxon's test. Table 7 summarises the results of this test.

The following observations can be made with respect to the results of the algorithms and the additional statistical analysis:

- The proposed GRASP+PR outperforms, again, the competing algorithms from the literature in a statistically significant way (see Table 7).

Table 7 GRASP+PR versus competitors using Wilcoxon's test (significance level $\alpha = 0.05$, critical value = 10; significance level $\alpha = 0.1$, critical value = 13) concerning the case of unrelated parallel machines

Competitor	R^+	R^-	Diff.? $\alpha = 0.05/0.1$
GA+H	66	0	yes/yes
DE	65.5	0.5	yes/yes
DE+VND	65.5	0.5	yes/yes
GA	66	0	yes/yes
GA+VND	66	0	yes/yes
Tabu Search	64.5	1.5	yes/yes
Multistart	61	5	yes/yes

Table 8 GRASP+PR versus pure GRASP. Results of Wilcoxon's test (significance level $\alpha = 0.05$, critical value = 10; significance level $\alpha = 0.1$, critical value = 13)

Competitor	R^+	R^-	Diff.? $\alpha = 0.05/0.1$
GRASP (Uniform par. machines)	55	11	no/yes
GRASP (Unrelated par. machines)	53	13	no/yes

- It is worth noting that GRASP+PR is able to achieve the best average performance for all 11 instance types. For the largest problem instances the results of GRASP+PR are not matched by any of the competing algorithms.
- As in the uniform instances, there is a great difference between the performance of DE and GA and their respective hybridizations with VND. Again, the local search procedure helps to significantly improve the quality of the obtained solutions.

4.4 Influence of path-relinking on GRASP+PR

The experimental results presented in this section aim at studying the influence of path-relinking on the performance of GRASP+PR. For this purpose we have repeated all experiments with pure GRASP, that is, the algorithm proposed in Sect. 3 without the path-relinking components. GRASP parameters are the same as those studied in Sect. 4.2 for GRASP+PR in order to focus exclusively on the effect of path-relinking on the behaviour of GRASP+PR. The Table 8 summarises the results of applying Wilcoxon's test with p -value = 0.05 and p -value = 0.1 both for the case of uniform and unrelated parallel machines, where the values of R^+ (associated with GRASP+PR) and R^- (associated with pure GRASP) are specified. Wilcoxon's test determines that GRASP+PR is statistically better than pure GRASP for both problem versions. This allows us to conclude that the combination of basic GRASP with path-relinking and evolutionary path-relinking helps us to achieve solutions of higher quality than those obtained by the pure algorithm version.

5 Conclusions

This paper has proposed a GRASP+PR algorithm for the SNIM-WCT problem, which combines the basic GRASP scheme with other elements such as path-relinking and evolutionary path-relinking. Moreover, we have performed a comparative study between the proposed

algorithm and the previously existing metaheuristics for this kind of problem. This study has shown that the GRASP+PR algorithm provides the best performance from among the studied algorithms. In addition, elements such as the greedy randomised initial solutions and path-relinking, which distinguish GRASP+PR from compared metaheuristics, have proven to be quite useful increasing the quality of the solutions and achieving high-quality solutions in a reasonable time. So, we can conclude from the experiments performed that this algorithm stands out as an excellent alternative to the existing methods for the SNIM-WCT problem.

We believe that the GRASP+PR algorithm presented in this paper is a significant contribution, worthy of future study. We will intend to explore two interesting avenues of research. Firstly, to adapt the GRASP+PR approach for its application to other variants of scheduling problems on parallel machines. Secondly, to build hybrid metaheuristics combining the proposed GRASP+PR with other salient metaheuristics for the SNIM-WCT problem.

Acknowledgements This work was supported by grant TIN2011-24124 of the Spanish government and by grant P08-TIC-4173 of the Andalusian regional government.

Appendix: Abbreviations

Table 9 List of abbreviations

Abbreviation	Description
CPU	Control Processing Unit
DE	Differential Evolution
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
MultiS	Iterative MultiStart method
PR	Path Relinking
RAM	Random Access Memory
RCL	Restricted Candidate List
SNIM-WCT	Scheduling on Non-Identical parallel Machines with Minimising the total Weighted Completion Times
VND	Variable Neighbourhood Descent

References

- Aiex, R., Binato, S., & Resende, M. (2003). Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing*, 29(4), 393–430.
- Aiex, R., Resende, M., Pardalos, P., & Toraldo, G. (2005). GRASP with path relinking for three-index assignment. *INFORMS Journal on Computing*, 17(2), 224–247.
- Allahverdi, A., Gupta, J., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, 27(2), 219–239.
- Anghinolfi, D., & Paolucci, M. (2007). Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Computers & Operations Research*, 34(11), 3471–3490.
- Arnaout, J. P., Rabadi, G., & Musa, R. (2010). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, 21, 693–701.
- Azizoglu, M., & Kirca, O. (1999a). On the minimization of total weighted flow time with identical and uniform parallel machines. *European Journal of Operational Research*, 113(1), 91–100.

- Azizoglu, M., & Kirca, O. (1999b). Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. *IIE Transactions*, 31(2), 153–159.
- Baker, K., & Merten, A. (1973). Scheduling with parallel machines and linear delay costs. *Naval Research Logistics Quarterly*, 20, 793–804.
- Belouadah, H., & Potts, C. (1994). Scheduling identical parallel machines to minimize total weighted completion time. *Discrete Applied Mathematics*, 48(3), 201–218.
- Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., & Weglarz, J. (2007). *International handbooks on information systems. Handbook on scheduling: models and methods for advanced planning*. Secaucus: Springer.
- Brucker, P., & Hurink, J. (2000). Solving a chemical batch scheduling problem by local search. *Annals of Operations Research*, 96(1), 17–38.
- Bruno, J., Coffman, E., & Sethi, R. (1974). Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17(7), 382–387.
- Buxey, G. (1989). Production scheduling: practice and theory. *European Journal of Operational Research*, 39, 17–31.
- Chen, C. L., & Chen, C. L. (2009). Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, 43(1), 161–169.
- Chen, Z. L., & Powell, W. (1999). Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing*, 11(1), 78–94.
- Cheng, R., Gen, M., & Tozawa, T. (1995). Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithms. *Computers & Industrial Engineering*, 29(1–4), 513–517.
- Cheng, T., & Sin, C. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47(3), 271–292.
- Croce, F. D., Tadei, R., & Asioli, P. (1999). Scheduling a round robin tennis tournament under courts and players availability constraints. *Annals of Operations Research*, 92, 349–361.
- Dodin, B., & Chan, K. H. (1991). Application of production scheduling methods to external and internal audit scheduling. *European Journal of Operational Research*, 52(3), 267–279.
- Elmaghraby, S., & Park, S. (1974). Scheduling jobs on a number of identical machines. *AIIE Transactions*, 6(1), 1–13.
- Fanjul-Peyro, L., & Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, 207(1), 55–69.
- Fanjul-Peyro, L., & Ruiz, R. (2011). Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Computers & Operations Research*, 38(1), 301–309.
- Fanjul-Peyro, L., & Ruiz, R. (2012). Scheduling unrelated parallel machines with optional machines and jobs selection. *Computers & Operations Research*, 39(7), 1745–1753.
- Feng, G., & Lau, H. (2008). Efficient algorithms for machine scheduling problems with earliness and tardiness penalties. *Annals of Operations Research*, 159, 83–95.
- Feo, T., & Resende, M. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2), 67–71.
- Feo, T., & Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2), 109–133.
- Garcia, S., Molina, D., Lozano, M., & Herrera, F. (2008). 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*, 15, 617–644.
- Glass, C. A., Potts, C. N., & Shade, P. (1994). Unrelated parallel machine scheduling using local search. *Mathematical and Computer Modelling*, 20(2), 41–52.
- Glover, F. (1996). Tabu search and adaptive memory programming—advances, applications and challenges. In *Interfaces in computer science and operations research* (pp. 1–75). Norwell: Kluwer Academic.
- Hall, L. A., Shmoys, D. B., & Wein, J. (1996). Scheduling to minimize average completion time: off-line and on-line algorithms. In *Proceedings of the seventh annual ACM-SIAM symposium on discrete algorithms, SODA '96* (pp. 142–151). Philadelphia: Society for Industrial and Applied Mathematics.
- Hall, L., Schulz, A., Shmoys, D., & Wein, J. (1997). Scheduling to minimize average completion time: off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22(3), 513–544.
- Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6, 65–70.
- Iman, R., & Davenport, J. (1980). Approximations of the critical region of the Friedman statistic. *Communications in Statistics. Theory and Methods*, 9(6), 571–595.
- Jarrah, A. I. Z., Bard, J. F., & de Silva, A. H. (1992). A heuristic for machine scheduling at general mail facilities. *European Journal of Operational Research*, 63(2), 192–206.

- Kendall, G., Tan, K., Burke, E., & Smith, S. (2010). Preface for the special volume on computational intelligence in scheduling. *Annals of Operations Research*, 180, 1–2.
- Laguna, M., & Marti, R. (1999). GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11(1), 44–52.
- Lenstra, J., Rinnooy-Kan, A., & Brucker, P. (1977). Complexity of machine scheduling problems. In B. K. P. L. Hammer, E. L. Johnson & G. Nemhauser (Eds.), *Studies in integer programming, annals of discrete mathematics* (Vol. 1, pp. 343–362). Amsterdam: Elsevier.
- Li, K., & Yang, S. L. (2009). Non-identical parallel-machine scheduling research with minimizing total weighted completion times: models, relaxations and algorithms. *Applied Mathematical Modelling*, 33(4), 2145–2158.
- Lin, S. W., Lu, C. C., & Ying, K. C. (2011a). Minimization of total tardiness on unrelated parallel machines with sequence- and machine-dependent setup times under due date constraints. *The International Journal of Advanced Manufacturing Technology*, 53, 353–361.
- Lin, Y., Pfund, M., & Fowler, J. (2011b). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Computers & Operations Research*, 38(6), 901–916.
- McNaughton, R. (1959). Scheduling with deadlines and loss functions. *Management Science*, 6(1), 1–12.
- Mokotoff, E. (2001). Parallel machine scheduling problems: a survey. *Asia-Pacific Journal of Operational Research*, 18(2), 193–242.
- Pendharkar, P., & Rodger, J. (2000). Nonlinear programming and genetic search application for production scheduling in coal mines. *Annals of Operations Research*, 95(1), 251–267.
- Phillips, C., Stein, C., & Wein, J. (1997). Parallel machine scheduling problems: a survey. *SIAM Journal on Discrete Mathematics*, 10(4), 573–598.
- Resende, M., Marti, R., Gallego, M., & Duarte, A. (2010). GRASP and path relinking for the max-min diversity problem. *Computers & Operations Research*, 37(3), 498–508.
- Resende, M., & Ribeiro, C. (2003). Greedy randomized adaptive search procedures. In F. Glover & G. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 219–249). Norwell: Kluwer Academic.
- Resende, M., & Ribeiro, C. (2003). A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41, 104–114.
- Resende, M., & Werneck, R. (2004). A hybrid heuristic for the p -median problem. *Journal of Heuristics*, 10(1), 59–88.
- Ribeiro, C., Uchoa, E., & Werneck, R. (2002). A hybrid GRASP with perturbations for the steiner problem in graphs. *INFORMS Journal on Computing*, 14(3), 228–246.
- Rochat, Y. (1998). A genetic approach for solving a scheduling problem in a robotized analytical system. *Journal of Heuristics*, 4, 245–261.
- Rosenbloom, E., & Goertzen, N. (1987). Cyclic nurse scheduling. *European Journal of Operational Research*, 31, 19–23.
- Sarin, S., Ahn, S., & Bishop, A. (1988). An improved branching scheme for the branch and bound procedure of scheduling n jobs on m parallel machines to minimize total weighted flowtime. *International Journal of Production Research*, 26(7), 1183–1191.
- Schulz, A., & Skutella, M. (1997). Random-based scheduling: new approximations and LP lower bounds. In *Proceedings of the first international symposium on randomization and approximation techniques in computer science (Random'97)* (pp. 119–133). Berlin: Springer.
- Schulz, A. S., & Skutella, M. (2002). Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics*, 15, 450–469.
- Skutella, M. (2001). Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48, 206–242.
- Vreddevel, T., & Hurkens, C. (2002). Experimental comparison of approximation algorithms for scheduling unrelated parallel machines. *INFORMS Journal on Computing*, 14(2), 175–189.
- Waligora, G. (2009). Tabu search for discrete-continuous scheduling problems with heuristic continuous resource allocation. *European Journal of Operational Research*, 193(3), 849–856.
- Weng, M., Lu, J., & Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics*, 70(3), 215–226.
- Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, 1, 80–83.
- Zaidi, M., Jarboui, B., Loukil, T., & Kacem, I. (2010). Hybrid meta-heuristics for uniform parallel machine to minimize total weighted completion time. In *Proc. of 8th international conference of modeling and simulation (MOSIM'10)*.

4. An Iterated Greedy Algorithm for the Large-Scale Unrelated Parallel Machines Scheduling Problem

The journal paper associated to this part is:

- F.J. Rodríguez, M. Lozano, C. Blum, C. García-Martínez, An Iterated Greedy Algorithm for the Large-Scale Unrelated Parallel Machines Scheduling Problem. **Computers & Operations Research**. **Submitted on second revision**.
 - Status: **Submitted on second revision**
 - Impact Factor (JCR 2011): 1.720.
 - Subject Category: Computer Science, Interdisciplinary Applications. Ranking 33 / 99 (Q2).
 - Subject Category: Engineering, Industrial. Ranking 7 / 43 (Q1).
 - Subject Category: Operations Research & Management Science. Ranking 10 / 77 (Q1).

An Iterated Greedy Algorithm for the Large-Scale Unrelated Parallel Machines Scheduling Problem

F.J. Rodríguez^a, M. Lozano^a, C. Blum^b, C. García-Martínez^c

^a*Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain*

^b*ALBCOM Research Group, Technical University of Catalonia, Barcelona, Spain*

^c*Department of Computing and Numerical Analysis, University of Córdoba, Córdoba, Spain*

Abstract

In this work, we tackle the problem of scheduling a set of jobs on a set of unrelated parallel machines with minimising the total weighted completion times as performance criteria. The iterated greedy metaheuristic generates a sequence of solutions by iterating over a constructive heuristic using destruction and construction phases. In the last few years, iterated greedy has been employed to solve a considerable number of problems. This is because it is based on a very simple principle, it is easy to implement, and it often exhibits an excellent performance. Moreover, scalability for high-dimensional problems becomes an essential requirement for modern optimisation algorithms. This paper proposes an iterated greedy model for the above-mentioned scheduling problem to tackle large-size instances. The benefits of our proposal in comparison to existing metaheuristics proposed in the literature are experimentally shown.

Keywords: Iterated greedy, unrelated parallel machines, large-scale optimisation, scheduling

1. Introduction

The *unrelated parallel machine scheduling with minimising total weighted completion times* problem (UPMS) considers a set of n jobs that have to be processed on m unrelated parallel machines so that the sum of the weighted completion times of the jobs is minimised. Different real-world applications of scheduling on parallel machines can be found in the literature, covering a

wide variety of fields such as human resources [1], production management [2, 3, 4], mail facilities [5], robotized systems [6], sport tournaments [7], and chemical processes [8].

The problem of minimising the total weighted completion time on two identical machines is NP-hard [9, 10]. Since the introduction of this problem by McNaughton in [11], different approaches to solve it have been presented. In the first place, there are exact procedures based on branch and bound [12, 13] and mathematical programming [14]. However, these approaches show limitations to solve large problem instances (branch and bound is limited to a maximum of 25 jobs and 5 machines, while the largest instances solved by mathematical programming have 100 jobs and 10 machines). Therefore, approximate algorithms are the most common way to tackle this problem. Within this latter category, we distinguish two completely different schemes: 1) approximate algorithms that, in general, try to find optimal solutions to relaxations of the problem (a review of the different proposals that fit this scheme can be found in [15]) and 2) algorithms based on metaheuristics, including different approaches such as local search methods [16], genetic algorithms [17], tabu search [16], and a greedy randomized adaptive search procedure [18].

Nowadays, large scale optimisation problems have been one of the most interesting trends in the last years for what concerns research on evolutionary algorithms and metaheuristics. This is due to the fact that many real-world problems are of large size. Unfortunately, the performance of many available optimisation algorithms deteriorates rapidly as the dimensionality of the search space increases. Thus, scalability for high-dimensional problems becomes an essential requirement for modern optimisation algorithms. In particular, we can find metaheuristics to face parallel machine scheduling problems that incorporate techniques for dealing with large size instances [19]. However, for the UPMS problem, we can observe that the previous studies has focused on studying the performance of the proposed metaheuristics on small/medium-size instances (up to 200 jobs) [17, 16, 18].

In this paper, we propose an *iterated greedy* (IG) algorithm [20, 21, 22] that allows reaching high-quality solutions especially for large-size instances, in scenarios with up to 1000 jobs and 50 machines. IG is a very simple and effective metaheuristic recently developed for combinatorial optimisation problems that follows a very simple principle, is easy to implement and can show excellent performance; in fact, it has exhibited state-of-the-art performances for a considerable number of problems [23, 24, 25, 26]. IG algorithms

try to improve iteratively a solution by removing elements from this solution and completing the resulting partial solution using a constructive procedure. Moreover, IG algorithms may make use of an improvement phase that takes the incumbent solution after destruction and reconstruction and performs local perturbations in order to find a better solution close to the incumbent one.

The remainder of this paper is structured as follows. In Section 2, we present the UPMS problem in detail and give an overview of the existing research on metaheuristics for this problem. In Section 3, we describe the proposed IG. In Section 4, we perform an experimental analysis of the proposal that has the following aims: 1) to study the IG behaviour depending on the parameters and settings associated and 2) to compare its results with state-of-the-art metaheuristics for the UPMS problem. Finally, in Section 5, conclusions and further work are presented.

2. The UPMS problem

The UPMS problem can technically be described as follows. Given is a set J of n jobs. Each job $j \in J$ has to be processed on exactly one machine i from a set M of m parallel machines. Note that a job may be processed by only one machine at a time, and a machine can process at most one job at a time. If a job j is processed on a machine i , it will take a positive integral processing time p_{ij} whose value is determined arbitrarily. Furthermore, each job has a non-negative integer weight w_j . The objective is to schedule the jobs so that the sum of the weighted completion times of the jobs is minimised:

$$\text{Minimise } \sum_{i=1}^n w_j \cdot C_j,$$

where C_j is the time at which the job j is completed.

It is important to note that the jobs assigned to a specific machine are processed in non-decreasing order with respect to the ratio between processing time (p_{ij}) and weight (w_j). This order is known as the *weighted shortest processing time* (WSPT) order. According to [27], sequencing the jobs in each machine following the WSPT ordering produces an optimal scheduling for this machine. A mixed integer linear programming formulation for the UPMS problem is provided for the sake of completeness. Let $x_{ij} = 1$ if the job j is processed on machine i and 0, otherwise. The model is stated by Vredeveld et al. [16] as:

$$\min \sum_{i=1}^n w_j \cdot C_j \quad (1)$$

$$\text{subject to: } \sum_i x_{ij} = 1 \quad \forall j, \quad (2)$$

$$C_j = \sum_{i=1}^n x_{ij} \cdot (p_{ij} + \sum_{k \prec i j} (p_{ik} \cdot x_{ik})) \quad \forall j, \quad (3)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j. \quad (4)$$

Constraint (3) ensures that the completion time of a job is its own processing time plus the processing times of its predecessors (according to the WSPT rule) on the machine on which it is scheduled and constraint (2) ensures that each job is scheduled on exactly one machine. According to the standard notation proposed by Azizoglu et al. [28] and Allahverdi et al. [29], the family of problems considered in this work is notated in the literature in the following manner: $R_m || \sum w_j * C_j$.

Different approaches considering a wide variety of metaheuristics have been presented so far to deal with the UPMS problem. In the first place, Weng et al. [30] presented seven *constructive procedures* for the UPMS problem and performed an experimental comparison between them. The computational experiments performed consider 4, 6, 8, 10 and 12 machines, and 40, 60, 80, 100 and 120 jobs. Three other constructive procedures were presented by Li et al [17]. One of them was a completely new development, whereas the other two were adaptations of those proposed by Alidaee et al [31] and Koulamas [32].

Later, Vredeveld et al. [16] presented two metaheuristics based on local search. These local search procedures make use of two types of different neighbourhood functions. The first function is called the *jump neighbourhood*. It consists in selecting a job j and a machine i such that job j is not scheduled on machine i . Then job j is moved to machine i . The second one is called *swap neighbourhood*. For this neighbourhood, two jobs j and k must be selected and assigned to different machines. The corresponding neighbouring solution is obtained by interchanging the machine allocations of the two selected jobs. These two neighbourhood functions were applied in the two metaheuristics based on local search proposed in [16]:

- *Multistart iterative method*. This procedure iteratively applies a first-

improvement local search to randomly generated solutions.

- *Tabu search.* This algorithm improves local search by storing information about solutions visited in past iterations in a so-called tabu list in order to avoid cycling, that is, returning to already visited solutions over and over again.

Recently, Li et al [17] presented a *genetic algorithm* approach to deal with unrelated parallel machine scheduling using three different performance criteria. In particular, the proposed approach initialises the population adding some solutions generated by heuristic methods. The remaining ones are generated randomly to provide enough diversity. Roulette wheel selection is used for choosing a new population with respect to a fitness-proportional probability distribution. The crossover and mutation schemes are the ones proposed by Cheng et al. [33]. Elitism is considered by removing two chromosomes and adding the best two previous chromosomes into the new population if they are not selected through the roulette wheel process. The experimental study performed compares the proposed genetic algorithm with a set of heuristics. Results show that the proposed algorithm outperforms the competing heuristics.

Finally, Rodriguez et al. [18] proposed a greedy randomized adaptive search procedure (GRASP) to solve the problem of scheduling on unrelated and uniform parallel machines. The proposed algorithm is a multi-start two-phase metaheuristic basically consisting of a solution construction phase and an improvement phase. The solution construction mechanism builds an initial solution using a greedy randomized procedure. This solution is improved by means of a variable neighbourhood descent method. Moreover, the proposed metaheuristic combines the basic GRASP scheme with backward path-relinking as an intensification method. Computational experiments consider instances with up to 200 jobs and compare the proposed algorithm with five different metaheuristics proposed previously in the literature. This study concludes that the proposed GRASP significantly outperforms the competing algorithms.

Moreover, it is important to note that metaheuristics have also been widely used for solving the problem of scheduling on unrelated parallel machines with various performance criteria. Tabu search, simulated annealing and genetic algorithms were applied in [34] to deal with the problem of minimising the makespan for unrelated parallel machines. The same problem

has been recently tackled by means of an iterated greedy metaheuristic by Fanjul-Peyro et al. [23]. Chen et al. [35] presented an hybrid metaheuristic combining tabu search and variable neighbourhood descent to minimise the total weighted tardiness on unrelated parallel machines.

Uniform and identical parallel machines scheduling problems have also been tackled with metaheuristics. Anghinolfi [36] faced the problem of minimising the total weighted tardiness on uniform parallel machines by means of a hybrid metaheuristic based on simulated annealing, tabu search and variable neighbourhood search. Zaidi et al. [37] presented four new metaheuristics to deal with the minimisation of the total weighted completion times on uniform parallel machines: a generational genetic algorithm, a differential evolution method, and two hybrid metaheuristics incorporating a variable neighbourhood descent method into the genetic algorithm and the differential evolution method. Finally, tabu search was proposed by Waligora [38] to minimise the makespan on identical parallel machines.

3. IG for the UPMS problem

In this section, we develop an IG algorithm to tackle the UPMS problem. The proposed IG tries to improve iteratively a solution through four different stages: destruction, construction, improvement, and the acceptance criterion. IG starts by generating an initial solution by means of a constructive procedure. Then, it applies iteratively the above-mentioned four-step process until a predefined stopping condition is satisfied. In the first place, some elements of the current solution are dropped (destruction). Then, the resulting partial solution is reconstructed using a constructive procedure (construction). A local improvement phase is applied to the reconstructed solution in order to improve its quality (improvement). Finally, IG chooses the new current solution between the current solution and the solution obtained from the improvement procedure (acceptance criterion).

Firstly, in Sections 3.1 and 3.2, we detail the constructive and destructive procedures of the proposed IG. Then, in Section 3.3, we detail the local improvement phase. Finally, in Section 3.4, we provide a complete description of the general scheme of the algorithm.

3.1. Constructive procedure

In this section, we outline the constructive procedure that is employed in the initialisation and the construction steps of the IG algorithm. The

constructive procedure extends a partial solution by adding one component from a set E of opportunely defined solution components. For the UPMS problem, this set E is composed of all the pairs (\bar{j}, i) , where \bar{j} is a job that belongs to the set of unscheduled jobs (\bar{J}) , $i \in M$. At each step, each solution component $e \in E$ is evaluated, searching for the most promising elements from E , by means of a heuristic $h(\cdot)$, where $h(e)$ denotes the heuristic value of e . Then, the element $e_* = (\bar{j}_*, i_*)$ such that $e_* = \operatorname{argmin}\{h(e) : e \in E\}$ is selected to be added to the current partial solution. Particularly, for the UPMS problem, the job \bar{j}_* will be scheduled on machine i_* , deleting \bar{j}_* from the set \bar{J} . This process is repeated until set \bar{J} is empty. The complete pseudocode of the constructive procedure can be found in Figure 1.

<p>Input: $J, M, h(\cdot)$ Output: s</p> <pre style="margin: 0;"> 1 $\bar{J} \leftarrow J;$ 2 $E = \bar{J} \times M;$ 3 while $\bar{J} \neq \emptyset$ do 4 $e_* = (\bar{j}_*, i_*) \leftarrow \operatorname{argmin}\{h(e) : e \in E\};$ 5 Assign job \bar{j}_* to machine i_* in solution $s;$ 6 $\bar{J} \leftarrow \bar{J} \setminus \{\bar{j}_*\};$ 7 $E = E \setminus \{(\bar{j}_*, i) : i = 1, \dots, m\};$ 8 end</pre>
--

Figure 1: Constructive procedure

For the UPMS problem, we can find in the literature different heuristics to determine at each step of the constructive procedure the job $\bar{j}_* \in \bar{J}$ and the machine $i_* \in M$ in which \bar{j}_* will be scheduled:

- Heuristic 1 ($h1$) [30]. In the first place, jobs are sorted in increasing order according to their weighted processing time $(p_j/w_j : j = 1, \dots, n)$, where $p_j = (\sum_{i=1}^m p_{ij})/m$. At each step, the job from \bar{J} with the lowest weighted processing time (\bar{j}_*) is assigned to the machine i_* , where

$$i_* = \operatorname{argmin}\{t_i + p_{i\bar{j}_*} : i = 1, \dots, m\},$$

being t_i the completion time of the last job scheduled on the machine i .

- Heuristic 2 (*h2*) [30]. The jobs are sorted in the same way as in *h1*. Moreover, at each step the job from $\bar{\mathcal{J}}$ with the lowest weighted processing time (\bar{j}_*) is assigned to the machine i_* determined as follows:

$$i_* = \operatorname{argmin}\{p_{i\bar{j}_*} : i = 1, \dots, m\}.$$

- Heuristics 3 (*h3*) and 4 (*h4*) [30] are the same as *h1* and *h2*, respectively, except that the jobs are sorted according to their weighted processing time p_j/w_j , where $p_j = \min\{p_{ij} : i = 1, \dots, m\}$.
- Heuristic 5 (*h5*) [30] assigns at each step the job $\bar{j}_* \in \bar{\mathcal{J}}$ to the machine $i_* \in M$ such that:

$$(i_*, \bar{j}_*) = \operatorname{argmin}\{(t_i + p_{ij})/w_j : i = 1, \dots, m; j = 1, \dots, n\}.$$

- Heuristic 6 (*h6*) [17] employs two steps. In the first one, a job $\bar{j}_* \in \bar{\mathcal{J}}$ is selected according to:

$$\bar{j}_* = \operatorname{argmin}\{t_i + p_{ij}/w_j : i = 1, \dots, m; j = 1, \dots, n\},$$

then, machine i_* on which it will be processed is determined as follows:

$$i_* = \operatorname{argmin}\{t_i + p_{i\bar{j}_*}/w_{\bar{j}_*} : i = 1, \dots, m\}.$$

- Heuristic 7 (*h7*) [31] selects, in the first place, the machine $i_* \in M$ with the earliest starting time, i.e., $i_* = \operatorname{argmin}\{t_i = \min\{t_i : i = 1, \dots, m\}\}$. After that, it assigns the job from $\bar{\mathcal{J}}$ with the lowest weighted processing time, $\bar{j}_* = \operatorname{argmin}\{p_{i_*j}/w_j : j = 1, \dots, n\}$, on machine i_* .
- Heuristic 8 (*h8*) [32] selects the machine as *h7*. However, the selected job $\bar{j}_* \in \bar{\mathcal{J}}$ is determined such that $\bar{j}_* = \operatorname{argmin}\{p_{i_*j} : j = 1, \dots, n\}$.

3.2. Destructive procedure

In this section, we describe the destructive procedure performed by the proposed IG. This destruction procedure is performed in an iterative way, removing an element each time. The process stops when the number of elements included in set $\bar{\mathcal{J}}$ is equal to $n_d\%$ of the elements of the current solution. The most simple and extended strategy to perform the IG destructive

procedure consists in randomly choosing the elements to be deleted. However, as it was showed by Fanjul et al. [23] for the case of scheduling on parallel machines with minimising makespan, more elaborated destruction procedures can lead to better solutions. Based on this idea, we have studied two different destruction strategies:

- In the first strategy (D1), one machine is uniformly selected at random and a binary tournament is performed in order to select the job to be deleted from this machine. Two jobs are randomly chosen and that with a lower p_{ij}/w_j ratio will be dropped. This destructive strategy tries to move the jobs scheduled at the beginning in each machine, which are delaying the jobs scheduled after them, to a more suitable machine.
- The second strategy (D2) also randomly selects a machine. However, it defines a different strategy to choose a job. In particular, this strategy selects the jobs whose processing times are shorter in other machines, and, consequently, could be completed earlier in other machines ([23]). Once a machine i is selected, the job k that has the largest processing time difference with respect to the other machines is deleted:

$$k = \operatorname{argmax}\{p_{ij} - p_{lj} : l \in M; l \neq i\}.$$

3.3. Improvement phase

The improvement phase aims to find a better solution close to the solution generated by the destruction and construction procedures. In the following, we revise different local search procedures for the UPMS problem that can be used to perform this task:

- First-improvement local search based on jump moves (F-Jump) [16]. Given a solution s , a neighbour is generated by randomly selecting one job and assigning this job to a different (arbitrarily chosen) machine. The first neighbour of the current solution yielding an improvement in the objective function is selected as the new current solution. This process is repeated until no improvement is possible.
- First-improvement local search based on swap moves (F-Swap) [16]. This local search procedure works in the same way as the one mentioned above, except that the neighbour of the current solution s is

generated by interchanging two randomly selected jobs from two different machines.

- A variable neighbourhood descent (VND) method was proposed in [37] to solve the problem of scheduling on uniform parallel machines. It uses both jump and swap moves to generate the neighbourhood of the current solution. In the first place, the neighbours are generated by means of jump moves and then, once there is no possible improvement, swap moves are applied. If a swap move leads to a better solution, jump moves are applied again. It also considers the first-improvement local search scheme. It is important to note that this procedure is more time-consuming than the two local search methods outlined above. In order to reduce its computational cost, we modified the VND method by avoiding the return to jump moves after applying swap moves. This way, it is more suitable for dealing with large-size instances.

3.4. General scheme of the proposed IG

The general scheme of the proposed IG is presented in Figure 2. It starts by generating an initial solution ($\text{Initialise}(s_0, \bar{J}, h)$) by means of the constructive procedure described in Section 3.1. In this case, the constructive procedure—using heuristic $h(\cdot)$ —generates a complete solution s starting from an empty solution (s_0). Then, the IG algorithm performs an iterative process until a maximum computation time limit t_{max} is reached. Each iteration consists of the following steps:

- In the destruction phase ($\text{Destruction}(s, n_d, \bar{J})$), $n_d\%$ of the elements of the current solution s are deleted, resulting in a partial solution s_d .
- The construction phase ($\text{Construction}(s_d, \bar{J}, h)$) makes use of the constructive procedure described in Section 3.1 to complete the partial solution s_d with the jobs from \bar{J} . The resulting candidate solution is labelled s_c .
- In the improvement phase ($\text{Improvement}(s_c)$), a local search procedure is performed in order to find better solutions near the current solution s_c .
- Finally, an acceptance criterion ($\text{AcceptCriterion}(s_p, s)$) decides whether the solution returned by the improvement phase (s_p) will become the

new current solution. The most used acceptance criteria are the following ones:

- ‘Replace if better’ acceptance criterion (RB). The new solution (s_p) is accepted only if its objective function value is better than the one of s [39].
- ‘Random walk’ acceptance criterion (RW). An IG algorithm using the RB acceptance criterion may lead to stagnation of the search due to insufficient diversification [22]. In order to avoid this, different acceptance criteria consider replacing the current solution by the new solution although its objective function value is worse. In [22], a simulated annealing acceptance criterion is employed, whereas in [40], the improved solution s_p becomes the new current solution independently of its objective function value. In this work, we have considered a new acceptance criterion to choose the new current solution. A uniform random number p_a between 0 and 1 is generated. If $p_a \leq 0.5$, s_p becomes the new current solution, independently of its objective function value. Otherwise, s_p is discarded.

<p>Input: t_{max}, n_d, h Output: s</p> <pre style="margin: 0;"> 1 $\bar{J} \leftarrow J;$ 2 Let s_0 be an empty solution; 3 $s \leftarrow \text{Initialise}(s_0, \bar{J}, h);$ 4 while <i>computation time limit t_{max} not reached</i> do 5 $s_d \leftarrow \text{Destruction}(s, n_d, \bar{J});$ 6 $s_c \leftarrow \text{Construction}(s_d, \bar{J}, h);$ 7 $s_p \leftarrow \text{Improvement}(s_c);$ 8 if $\text{AcceptCriterion}(s_p, s)$ then 9 $s \leftarrow s_p;$ 10 end 11 end</pre>

Figure 2: IG scheme

4. Computational experiments

In this section, we present the experiments performed in this work to analyse the behaviour of the IG algorithm presented in the previous section. Firstly, we outline the experimental framework used in this study (Section 4.1); then, we show and analyse the results obtained from different experiments carried out with the proposed IG. These experiments have two main goals: 1) to find suitable values for the IG parameters and understand their influence on the behaviour of the algorithm (Section 4.2) and 2) to perform a comparison between the proposed IG and state-of-the-art metaheuristics from the literature (Section 4.3).

4.1. Experimental framework

All algorithms under consideration have been implemented in C++ and the source code has been compiled with gcc 4.6. The experiments were conducted on a computer with a 2.8 GHz Intel® Core™ i7-930 processor¹ (8MB cache, 4 cores and 8 threads) with 12 GB of RAM running Fedora™ Linux V15. Each execution of an algorithm is performed sequentially, using a unique thread. In this work, we have used instances with unrelated parallel machines considering 20 different combinations of the number of jobs (n) and the number of machines (m). These 20 instance types are outlined in the first two columns of Table 1. Moreover, in the same table we find—in the 3rd column—the maximum CPU time allotted to each instance type (n seconds). Seven sets of ten problem instances were randomly generated for each instance type following different ways of choosing the processing time of job j on machine i (p_{ij} , $i = 1, \dots, n$ and $j = 1, \dots, m$) and its weight (w_j).

- *UC1*: In the first set, the processing time of job j on machine i was chosen uniformly at random from $[1, 100]$. The weights of the n jobs were selected uniformly at random from $[1, 10]$. These kind of instances are known in the literature as uncorrelated [23].
- Moreover, it is frequent in the literature to find benchmarks where the processing times are job or machine correlated [23].
 - *JC*: Job correlated instances: processing time p_{ij} is determined by the expression $p_{ij} = b_j + d_{ij}$ where b_j values, associated to the

¹More information: <http://ark.intel.com/Product.aspx?id=41447>

job j , are selected uniformly at random from the interval $[1, 100]$ and d_{ij} are chosen uniformly at random from $[1, 20]$. The weights of the jobs were selected uniformly in the interval $[1, 10]$.

- *MC*: Machine correlated instances: processing time p_{ij} is generated by the expression $p_{ij} = a_i + c_{ij}$, being a_i , associated to the machine i , selected uniformly at random from $[1, 100]$ and c_{ij} selected uniformly at random from $[1, 20]$. The weights of the jobs are generated as in the *JC* instances.
- *UC2, UC3, UC4*: For uncorrelated instances, some recent works have concentrated on uncorrelated instances dealing with others intervals for the processing time values: $[10, 100]$ (*UC2* [41]), $[100, 200]$ (*UC3* [23]) and $[100, 120]$ (*UC4* [23]), among others. For the sake of completeness, we have included three different sets that use these intervals for choosing the processing time values. The weights of the jobs are generated as in *UC1*.
- *UC5*: In this work we additionally utilize a different interval for the weights of the jobs concerning uncorrelated instances. In particular, we use the interval $[1, 100]$ for selecting uniformly at random the weights of the jobs. The processing times are chosen as in *UC1*.

In order to assure the quality and the reliability of the results of our non-deterministic IG algorithm, we present the average results over ten instances generated randomly instead of executing ten times an algorithm over a unique instance, which is a common choice in recent works dealing with this problem [37] or other optimisation problems [42].

Non-parametric tests [43] have been used to compare the results of the different algorithms under consideration. The only condition to be fulfilled for the use of non-parametric tests is that the algorithms to be compared should have been tested under the same conditions (that is, the same set of problem instances, the same stopping conditions, the same number of runs, etc). On the contrary, in order to use the parametric tests, it is necessary to check the following conditions: independence, normality, and heteroscedasticity. It has been observed that these conditions are not always met in the data to be analysed [43]. In the first place, mean ranking for each algorithm is computed according to Friedman’s test [44, 45]. This measure is obtained by computing, for each problem, the ranking r_a of the observed result for

Number of jobs (n)	Number of machines (m)	Time limit (seconds)
200	5	200
	10	
	20	
	50	
250	5	250
	10	
	20	
	50	
300	5	300
	10	
	20	
	50	
500	5	500
	10	
	20	
	50	
1000	5	1000
	10	
	20	
	50	

Table 1: 20 *instance types* considered for each of the seven sets of instances. The last table column provides the maximum CPU time limit for each instance type (in seconds).

algorithm a , assigning to the best of them the ranking 1, and to the worst the ranking $|A|$ (A is the set of algorithms). An average measure is obtained from the rankings of these methods for all the test problems. Then, we have considered two alternative methods based on non-parametric tests to analyse the experimental results:

- The first method is the application of Iman and Davenport’s test [46] with Holm’s method [47] as a post-hoc procedure. The first test may be used to see whether there are significant statistical differences among the algorithms in a certain set. If such differences are detected, then the Holm’s method is employed to compare the best algorithm (that is, the control algorithm) against the remaining ones (Appendix Appendix A).
- The second method concerns the utilisation of the Wilcoxon’s matched-pairs signed-ranks test [48]. With this test, the results of two algorithms may be directly compared (Appendix Appendix B).

4.2. Study of the proposed IG with different parameters

In this section, we investigate the effect of the different parameters and the algorithmic components of IG on its behaviour: (1) the heuristic used in

the construction phase (Section 4.2.1), (2) the parameter n_d , associated to the destruction step, (3) the local search procedure (Section 4.2.3), (4) the acceptance criterion (Section 4.2.4), and (5) the destruction strategy used in the destruction step. The experimental framework used to perform this study is the one described in Section 4.1. For the parameter study we have considered a subset of the instances described in the previous section in order to avoid overfitting. We have considered instances belonging to UC1, JC, and MC with a number of jobs between 250 and 500.

4.2.1. Study of the heuristics for the constructive procedure

In this section, we have compared the performance of the proposed IG using the different heuristics for the constructive procedure detailed in Section 3.1. The values for the different parameters are fixed in the following way: $n_d = 15\%$, F-jump as the improvement method, RW as the acceptance criterion and random selection of machines and jobs in the destruction step.

Table 2 shows the mean ranking—in the 2nd column—of the IG algorithm with a specific heuristic—as indicated in the 1st column—over the instances selected for performing the tuning. In the first place, Iman and Davenport’s test finds significant performance differences between the considered algorithms because the statistical value (26.67) is greater than the critical one (2.047) for a level of significance $\alpha = 0.05$. Then, we applied Holm’s test in order to find significant performance differences between the best ranked configuration (IG with the heuristic h6) and the other ones. The third column shows whether Holm’s test finds significant differences between the best ranked algorithm and the corresponding one (+), or if there are no significant differences (\sim).

As shown in Table 2, IG with heuristic h6 gets the best ranking and Holm’s test finds significant performance differences with regard to h1, h3, h7, and h8. We choose heuristic h6 for all remaining experiments, because the best ranked configuration employs this setting.

4.2.2. Study of parameter n_d

In this section, we study the influence of the value of parameter n_d on the performance of the IG proposed for the UPMS problem. This parameter determines the number of jobs that will be unscheduled in the destructive step with the aim of allocating them later to possibly different machines. The remaining parameters are fixed with the following values: F-jump as

Heuristic of the construction step	Ranking	Holm
h6	2.83	Winner
h2	2.90	~
h4	3.06	~
h5	3.44	~
h3	5.17	+
h7	6.06	+
h1	6.19	+
h8	6.41	+

Table 2: Comparison, using Holm’s test, of eight IG versions using different heuristics.

the improvement procedure, RW as the acceptance criterion, and random selection of machines and jobs in the destructive phase.

We have performed the study considering four different values for n_d : 15%, 25%, 50%, and 75%. First, Iman and Davenport’s test finds significant performance differences between the considered configurations for a level of significance $\alpha = 0.05$ (its statistical value, 10.56, is greater than its critical one, 2.69). In Table 3, we present the results of Holm’s test—3rd column—and the mean ranking—2nd column.

Value of parameter n_d	Ranking	Holm
15%	1.66	Winner
75%	2.53	+
25%	2.64	+
50%	3.17	+

Table 3: Comparison, using Holm’s test, of four IG versions with different values for n_d .

According to the results of Holm’s test in Table 3, the best ranked configuration is the one that destroys 15% of elements from the current solution during the destructive step, exhibiting statistically significant differences with respect to IG with all the remaining n_d values. Clearly, the best results are achieved by removing a small percentage of elements from the current solution. We should point out that we set $n_d = 15\%$ for the rest of our experimentation, because IG has exhibited the best performance with this setting.

4.2.3. Study of the performance of improvement methods

The objective of this section is to determine the improvement method (from the ones presented in Section 3.3) that provides the best performance when used within the proposed IG. In order to deal with this objective, we have evaluated the performance in terms of solution quality. For this study, we employ RW acceptance criterion and random selection of machines and jobs in the destructive step.

In the first place, we have applied Iman and Davenport’s test (level of significance $\alpha = 0.05$) that has found significant performance differences between the considered algorithms (the Iman-Davenport value is 52.06 and the critical one is 3.13). Then, we have compared the best ranked algorithm, IG with F-jump, with the other IG variants by means of the Holm’s test.

Improvement method	Ranking	Holm
F-jump	1.17	Winner
VND	2.13	+
F-swap	2.70	+

Table 4: Comparison, using Holm’s test, of three IG versions with different local improvement methods.

The results are presented in Table 4—in the 3rd column—and show that F-jump provides a statistically better performance with respect to F-swap and VND. For this reason, we employ F-jump as improvement method for the rest of our experimentation.

4.2.4. Study of the acceptance criterion

The acceptance criterion determines the starting solution for the next iteration of the IG algorithm. In this section we compare the behaviour of the proposed IG algorithm with respect to the acceptance criterion employed, using random selection of machines and jobs in the destructive phase.

We have undertaken this comparative analysis by means of Wilcoxon’s test, which compares the results of two IG versions with different acceptance criteria. Table 5 contains the results for a level of significance $\alpha = 0.05$ (critical value = 208), by providing the values of $R+$ (associated with IG using the RW acceptance criterion) and $R-$ (associated with IG using the RB acceptance criterion). If $R-$ is smaller than $R+$ and the critical value of the test, IG-RW is statistically better than IG-RB; if $R+$ is inferior to $R-$ and

the critical value, IG-RW is statistically worse than its competitor; if neither $R+$ nor $R-$ is smaller than the critical value, the Wilcoxon’s test does not find statistical differences. The last column indicates whether IG-RW performs statistically better (+), worse (−), or without significant differences (\sim) than IG-RB.

	R+	R-	Wilcoxon’s test result
IG-RB vs IG-RW	574	92	+

Table 5: IG-RW vs. IG-RB (Wilcoxon’s test with $\alpha = 0.05$ and critical value = 208).

According to the results of Wilcoxon’s test presented in Table 5, the use of the RW acceptance criterion leads to the best results for the IG algorithm. These results reaffirm the conclusions of previous studies [40, 22] and suggest that RW is possibly a good alternative also for solving other optimisation problems. We choose the RW acceptance criterion for all remaining experiments.

4.2.5. Study of the performance of different destruction strategies

In this section, we study the behaviour of the proposed IG depending on the strategy used during the destruction step. We have performed this study by means of Wilcoxon’s test in order to perform a pairwise comparison between the results of IG with random selection of machines and jobs in the destruction step (DR) and those of the IG models that employ destruction strategies D1 and D2.

Table 6 provides the values of $R+$ (associated with IG using the destruction strategies D1 and D2) and $R-$ (associated with IG using the destruction strategy DR) (level of significance $\alpha = 0.05$). In this table, the results for the UC instances and the JC+MC instances are treated separately.

According to the results of Wilcoxon’s test as shown in Table 6, using more elaborate strategies such as D1 and D2 clearly improves the performance of the proposed IG for what concerns the JC+MC instances. However, when considering the UC instances, Wilcoxon’s test does not show significant performance differences for destruction strategy D1 and indicates that for this kind of instances D2 provides a poorer performance than DR. Taking these results into account, we consider that IG-RW-D1 provides the best performance over all the remaining alternatives.

JC+MC instances			
	R+	R-	Critical value = 81
IG-RW-D1 vs IG-RW-DR	220	80	+
IG-RW-D2 vs IG-RW-DR	253	47	+
UC instances			
	R+	R-	Critical value = 13
IG-RW-D1 vs IG-RW-DR	58	20	~
IG-RW-D2 vs IG-RW-DR	0	78	-

Table 6: Comparison between the destruction strategies by means of Wilcoxon’s test.

4.3. IG-RW-D1 vs. state-of-the-art metaheuristics for the UPMS problem

In this section we compare the IG-RW-D1 approach with different metaheuristics found in the literature for tackling the UPMS problem. More specifically, we considered the following approaches (Section 3):

- Iterative multistart method (**MultiS**) [16].
- Tabu search (**Tabu**) [16].
- Genetic algorithm (**GA**) [17].
- GRASP + path relinking (**GRASP**) [18].

All these approaches were implemented in C++ using the same data structures as the proposed IG. The parameter values used for each algorithm are the ones recommended in the original works. For the IG-RW-D1 algorithm, the parameters are selected according to the experimental study performed in Section 4.2: heuristic h6, $n_d = 15\%$, and F-jump. In order to assure a fair comparison, each algorithm was applied under the same conditions as IG-RW-D1, that is, each algorithm was applied exactly once to each of the 1400 problem instances (see Table 1). Moreover, the same CPU time limits as for IG-RW-D1 were used. In addition, we provide the results obtained by IBM-ILOG-CPLEX, using the model stated in Section 2, to be used as a reference point in the experimental study. The results that are presented for CPLEX were obtained with a computation time limit of 2 hours for each problem instance.

4.3.1. Case 1: UC1 instances

Table 7 presents the results of all algorithms for the UC1 instances. Note that these results are averaged over the 10 instances for each of the 20 problem types. Hereby, a problem type is defined by specific values for n and m . Table 7 provides two different measures: (1) the average result and (2) the average relative percentage deviation (RPD), which is calculated as follows:

$$RPD = \frac{C_{met}(i) - C_{cplex}(i)}{C_{cplex}(i)} \cdot 100,$$

where $C_{met}(i)$ is the average value obtained by a given metaheuristic for the instances of type i , and $C_{cplex}(i)$ is the average of the solution value obtained by CPLEX within a computation time limit of 2 hours. The best result concerning the metaheuristics is indicated—both concerning the average solution quality and the RPD—in each table row in boldface.

n	m	CPLEX	GA		GRASP		MultiS		Tabu		IG-RW-D1	
		Sol.	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD
1000	50	*	82551	-	158604.6	-	1593174	-	1593174	-	81542	-
	20	372192.4	373949.1	0.47	576481	55.00	3577016	981.80	3577016	965.26	372005.4	-0.05
	10	1286840.3	1291135	0.33	2034004	58.16	8229562	543.43	8229562	539.73	1286803	-0.00
	5	4501612.8	4506558	0.11	5280587	17.31	17162120	276.16	17162120	281.35	4501648	0.00
500	50	24248.3	23393.3	-3.52	25653.5	5.71	151931.7	484.77	151931.7	526.62	22854.3	-5.74
	20	97140.7	98024.1	0.91	97082.9	-0.06	476167.7	340.77	476167.7	390.27	96985.4	-0.16
	10	326886	329057.8	0.66	327037.8	0.05	1232319	283.66	1232319	277.55	326867	-0.01
	5	1090245	1093097	0.26	1090345	0.01	3170840	190.20	3170840	191.07	1090260	0.00
300	50	10848.8	9910.1	-8.63	9678.1	-10.76	11844.8	27.75	11844.8	9.07	9621.5	-11.29
	20	36561.5	37179.7	1.693	36524.1	-0.10	38132.4	3.19	38132.4	4.31	36489.3	-0.19
	10	117841.1	118791.6	0.81	117856.2	0.01	120852	7.15	120852	2.55	117822.7	-0.02
	5	389826.5	391163.3	0.34	389854.5	0.01	444678.4	10.80	444678.4	14.04	389831.8	0.00
250	50	7858.5	7258.3	-7.62	7042.6	-10.37	7352.3	-7.09	7352.3	-6.46	6998.5	-10.92
	20	27594.4	28026.6	1.56	27577.4	-0.06	27672.7	0.28	27672.7	0.28	27554.9	-0.14
	10	80333.9	81287.4	1.187	80341.3	0.01	80408.2	0.09	80408.2	0.09	80333.3	-0.00
	5	284705.8	286048.8	0.47	284719.2	0.01	284846.6	0.08	284846.6	0.05	284711.6	0.00
200	50	5795.9	5138.5	-11.22	5006.6	-13.49	5062.7	-12.61	5062.7	-12.51	4989.4	-13.79
	20	17026.4	17345	1.87	16995.8	-0.18	17066.8	0.06	17066.8	0.24	16983	-0.25
	10	52918.1	53676.7	1.437	52925	0.01	53039.2	0.11	53039.2	0.23	52919.6	0.00
	5	182100	183400.4	0.71	182113.6	0.01	182166	0.02	182166	0.04	182108.6	0.00
Global Av. RPD				-0.956		5.329		164.770		167.568		-2.240

Table 7: Results of the studied algorithms for the UC1 instances.

The results of the algorithms allow us to make the following observations:

- The proposed IG-RW-D1 obtains the lowest *global average RPD* (see last table row) among all methods compared in this study. It is worth

highlighting that IG-RW-D1, according to the value of global average RPD, obtains on average better results than CPLEX with significantly lower runtimes. The complexity of the problem is shown by the fact that CPLEX is not able to find the optimal solution for any instance.

- IG-RW-D1 obtains the best average result for all instance types. This is especially noteworthy for what concerns the complex instances with a large number of jobs and machines. In fact, CPLEX is not able to find a feasible solution for the instances with 1000 jobs and 50 machines (as indicated by the asterisk). On the contrary, for large instances IG-RW-D1 is able to reach high quality solutions that mostly improve over those obtained by CPLEX.
- It is worth noting that, apart from IG-RW-D1, only GA performs (on average) better than CPLEX. The performance of GRASP and, specially, of MultiS and Tabu degrades significantly for what concerns the instances with more than 300 jobs.

4.3.2. Case 2: JC and MC instances

The same experimental evaluation was repeated concerning the 400 instances from sets JC and MC. The detailed average results and RPDs of the studied metaheuristics over the 10 instances for each of the 20 problem types are shown in Tables 8 and 9, respectively.

The following observations can be made with respect to the results as shown in Tables 8 and 9:

- The proposed IG-RW-D1 achieves the best global average RPD for both instance sets (JC and MC). As suggested by Fanjul et al. [23] for the problem of minimising the makespan in the context of unrelated parallel machines, this kind of instances seems to represent a more challenging problem than uncorrelated ones. In fact, the number of instances in which IG-RW-D1 and the other metaheuristics are able to reach better average results than CPLEX is considerably reduced in comparison to the uncorrelated case. However, it is important to highlight that, despite this fact, IG-RW-D1 obtains a better average result than CPLEX for the JC instances (the global average RPD value is -0.17). Moreover, for what concerns the MC instances, the global average RPD is lower than 1%.

n	m	CPLEX	GA		GRASP		MultiS		Tabu		IG-RW-D1	
		Sol.	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD
1000	50	*	1681082	-	1860562	-	2384658	-	2354086	-	1676338	-
	20	2812214.7	4020891	0.03	4397002	6.67	5690978	28.87	5658121	28.38	3993860	-0.72
	10	7992969.8	8041254	0.60	8696201	8.80	11322570	41.66	11366880	42.21	8000019	0.09
	5	16468641.0	16510720	0.26	17208400	4.49	22368410	35.82	22468270	36.43	16470900	0.01
	50	455696.3	441278.1	-3.11	456530.1	0.24	503967.8	10.66	497768.5	9.29	440223.3	-3.34
500	20	990823.3	1004954.4	1.43	1001098.5	1.04	1122845	13.33	1153897	16.46	994813.4	0.40
	10	1946721.1	1968328	1.11	1953564	0.35	2386413	22.59	2368174	21.64	1950641	0.20
	5	3960537.6	3981339	0.53	3962971	0.06	4960288	25.24	4896540	23.63	3961860	0.03
	50	180707.9	179281.2	-0.78	184901	2.33	179148.7	-0.85	179130.7	-0.86	179103	-0.88
	20	376898.4	384228.5	1.95	383863.7	1.85	378403.8	0.40	378783.3	0.50	380244.1	0.89
300	10	721510.6	732777.5	1.56	725082.2	0.49	728089.6	0.91	727883.3	0.88	724190.2	0.07
	5	1454399.9	1466592	0.84	1455911	0.10	1489475	2.41	1488205	2.33	1455348	0.07
	50	133957.5	131809.1	-1.60	136706.9	2.06	130900.3	-2.28	130869.6	-2.30	131725.6	-1.66
	20	269330.4	275372	2.24	274606	1.96	269692.7	0.13	269789.6	0.17	272281.6	1.10
	10	508920.1	518192.9	1.82	511824.3	0.57	509635.9	0.14	509635.9	0.14	511335.1	0.47
250	5	1017577.1	1028106	1.03	1018244	0.07	1019075	0.15	1018609	0.10	1018525	0.09
	50	93210.3	91365.1	-1.97	94318	1.19	90742.6	-2.64	90848.7	-2.53	91397.3	-1.94
	20	179713.8	183398.7	2.05	180621.4	0.51	179780.5	0.04	180057.3	0.19	181828.6	1.18
	10	333166.1	340022.6	2.06	333605.4	0.13	333502.1	0.10	333566.5	0.12	335165.1	0.60
	5	662227.1	670125.7	1.19	662420.4	0.03	662448.5	0.03	662454.4	0.03	663023.5	0.12
Global Av. RPD				0.59		1.73		9.30		9.31		-0.17

Table 8: Results of the studied algorithms for the JC instances

n	m	CPLEX	GA		GRASP		MultiS		Tabu		IG-RW-D1	
		Sol.	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD
1000	50	*	1059816.4	-	1229773	-	2791200	-	2742746	-	1016939.3	-
	20	2427184.9	2775194	23.95	2946319	36.01	6885581	307.66	6847211	305.28	2565055	9.56
	10	4837163.3	5567272	15.14	5563880	15.17	14032600	191.16	14075400	192.17	4985071	3.01
	5	12205493.2	13417679	10.25	12664642	3.98	29683830	150.67	29829520	151.77	12243061	0.34
	50	256643.2	275730.4	7.50	271501.1	5.93	430804.1	67.87	429214.9	67.47	250819.9	-2.24
500	20	613633.6	688743.6	12.32	629461.8	0.59	945266.5	54.70	1021257.5	67.07	617157.2	0.58
	10	1109898.1	1260246	13.59	1112615.9	0.26	2145814	94.69	2053756	85.90	1111212.5	0.12
	5	2316674.1	2556277	10.84	2317241	0.03	4509104	101.77	4475050	99.97	2316942	0.01
	50	155277.3	161737.5	4.17	156473.6	0.77	155540.6	0.17	154242.6	-0.68	155080.2	-0.12
	20	326842.2	356135.9	8.99	331031.2	1.29	331163.6	1.38	331663.8	1.48	329766.9	0.90
300	10	540214.8	597591.7	10.70	541631.4	0.27	582245.9	8.07	573157.1	6.33	541385.4	0.22
	5	1045814.3	1125015.4	7.72	1046170.6	0.03	1189685.7	13.97	1235179	19.27	1046159.3	0.03
	50	90587.8	94908.8	4.80	90478.5	-0.10	88473.7	-2.31	88071.3	-2.76	89862.1	-0.78
	20	217191.9	239618.3	10.34	219469.5	1.05	217265.2	0.04	217262.9	0.03	219497.1	1.06
	10	508912.6	552872.8	9.13	510777.7	0.38	509260	0.07	509650.6	0.16	511137.1	0.44
250	5	930673.6	1003699.6	8.34	931087.9	0.05	960528.1	3.37	954203.7	2.85	931297.6	0.07
	50	68725.9	72127	4.96	69643.7	1.34	67531.2	-1.73	67547.7	-1.70	69196.7	0.69
	20	162849	176181.6	8.23	164387.3	-0.15	162610.3	-0.14	162692.7	-0.09	164627.5	1.10
	10	344534.7	373137.2	8.69	344697.6	0.05	344464.6	-0.02	344492.1	-0.01	345944.5	0.40
	5	828483.3	862823.1	5.04	828611.8	0.01	828666.5	0.02	828723.1	0.03	829178.9	0.08
Global Av. RPD				9.72		3.52		52.18		52.34		0.81

Table 9: Results of the studied algorithms for the MC instances

- Apart from IG-RW-D1, the best results are achieved again by GA and GRASP, which highlights the importance of using heuristic information for solving this problem. However, for the JC and MC instances, MultiS and Tabu improve their performance with respect to the UC1 instances. However, they are both not competitive for the largest instances.
- The instances in which IG-RW-D1 obtains the best results are those with more than 250 jobs. It is important to note that the complexity of an instance is further increased when considering a larger number of machines. These instances represent a big challenge because the cost of a job not only depends on the machine to which it is allocated but also on the other jobs allocated to this machine. In order to minimise this cost, it is necessary to consider the scheduling of these jobs on other machines, resulting in a larger search space as the number of machines increases. In fact, we notice that CPLEX experiences the greatest difficulties in solving instances with a large number of machines, to such an extent that it is not able to resolve those instances with 1000 jobs and 50 machines. For this reason, the good results of IG-RW-D1 on these instances become highly significant (see, for example, the average RPD for the JC instances with 500 jobs and 50 machines or 1000 jobs and 20 machines). Moreover, IG-RW-D1 reaches the best average result for 11 out of 12 JC and MC instance types with more than 250 jobs.

4.3.3. Case 3: Instance sets UC2, UC3, UC4, and UC5

In this section, we analyse the behaviour of the studied algorithms when facing different intervals for processing times and weights concerning uncorrelated instances (sets UC2, UC3, UC4, and UC5). The detailed results of all algorithms for instances sets UC2, UC3, UC4, and UC5 are shown in Tables 10, 11, 12, and 13, respectively. Again, the results are presented as averages over the 10 instances for each of the 20 instance types.

Based on the results that are shown in these tables, we can make the following observations:

- The behaviour of IG-RW-D1 on the different instances is quite robust, reaching the best global average RPD for all four instance sets. Moreover, we should highlight that the global average RPD is lower than 0 in three out of four sets (UC2, UC3, and UC5), which means that

n	m	CPLEX	GA		GRASP		MultiS		Tabu		IG-RW-D1						
		Sol.	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD					
1000	50	*	453429.9	-	546201.8	-	1975134	-	1964045	-	449911.3	-					
	20	1271735.5	1277098	0.42	1628732	28.06	4833726	280.10	4780005	275.87	1271480	-0.02					
	10	3062495.3	3070287	0.25	4172726	36.29	9923652	224.05	9881675	222.68	3062433	0.00					
	5	7950232.4	7958324	0.10	10121672	27.34	20145360	153.42	20130850	153.23	7950292	0.00					
	50	122240.5	123282.8	0.85	138485.4	13.29	254528.4	108.20	245229.9	100.60	121648.7	-0.48					
500	20	325454.6	328010.2	0.78	333004.8	2.31	613012.4	88.36	610560.2	87.61	325421	-0.01					
	10	775560.5	778865.7	0.43	779495.2	0.51	1717337	121.47	1663783	114.58	775521.5	0.00					
	5	1949461.9	1953992	0.23	1949525	0.00	3891960	99.71	3903851	100.36	1949495	0.00					
	50	49271.6	49866.9	1.21	49169.4	-0.20	52138.9	5.82	51622.6	4.77	49040.4	-0.47					
	20	123300.9	124822.7	1.23	123339	0.03	124079	0.63	123666.7	0.30	123264.6	-0.03					
300	10	283481.9	285240.5	0.62	283522.9	0.01	285004.6	0.54	284417	0.33	283480.8	0.00					
	5	703612.4	705684.8	0.29	703631.1	0.00	736162.2	4.60	744223.9	5.74	703623.1	0.00					
	50	35491.9	35856.9	1.03	35423.4	-0.19	35743.3	0.71	35746.9	0.72	35297.3	-0.54					
	20	88415.9	90072	1.87	88470.4	0.06	88648.9	0.26	88648.9	0.26	88401	-0.02					
	10	194239.7	196144	0.98	194253.9	0.01	194374.4	0.07	194374.4	0.07	194238.3	0.00					
250	5	500819.7	502933.8	0.68	500844.1	0.00	500901.4	0.02	500901.4	0.02	500838.6	0.00					
	50	23926.8	24195.4	1.13	23814.9	-0.46	24039.3	0.47	24086.9	0.68	23666.1	-1.08					
	20	55539.7	56468.6	1.67	55538.9	0.00	55677	0.25	55753.8	0.39	55512.8	-0.05					
	10	124083.3	125445.3	1.10	124089	0.00	124143	0.05	124172.8	0.07	124078.9	0.00					
	5	317587.6	319098.6	0.48	317589.5	0.00	317636.4	0.02	317670.9	0.03	317593.4	0.00					
Global Av. RPD			0.809			5.636			57.302			56.226			-0.142		

Table 10: Results of the studied algorithms for the UC2 instances.

n	m	CPLEX	GA		GRASP		MultiS		Tabu		IG-RW-D1						
		Sol.	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD					
1000	50	*	3984904	-	4201080	-	5644122	-	5627207	-	3984770	-					
	20	9585440.5	9601832	0.17	10041829	4.76	13694150	42.86	13694000	42.86	9599480	0.15					
	10	19050140	19087210	0.19	19805210	3.96	27549150	44.61	27460940	44.15	19060170	0.05					
	5	38358062.5	38398500	0.11	39458960	2.87	53168930	38.61	53119970	38.48	38360640	0.01					
	50	1079109.4	1058084	-1.90	1103632	2.31	1182938	9.67	1174939	8.92	1058183	-1.90					
500	20	2440544.1	2448320	0.32	2491829	2.10	2757078	12.97	2748709	12.62	2447621	0.29					
	10	4782366.9	4798311	0.33	4842889	1.27	5845448	22.23	5890625	23.17	4789561	0.15					
	5	9543204.7	9561636	0.19	9553104	0.10	11831300	23.97	11843450	24.10	9545302	0.02					
	50	422833.3	419773.1	-0.72	433845.2	2.61	421202.9	-0.38	420989.5	-0.44	419747.1	-0.73					
	20	921061.4	924945.2	0.42	926082.8	0.55	922666.8	0.17	922946.6	0.20	924686.3	0.39					
300	10	1769001.3	1777483	0.48	1771032	0.11	1774620	0.32	1774552	0.31	1774309	0.30					
	5	3497177.7	3507447	0.29	3498301	0.03	3566101	1.97	3583271	2.46	3498964	0.05					
	50	303748.2	300458.9	-1.08	305999	0.74	300535.2	-1.06	300535.2	-1.06	300421	-1.09					
	20	643737.2	645955.3	0.34	644392.4	0.10	644584.8	0.13	644590.8	0.13	646003.4	0.35					
	10	1222891.9	1229416	0.53	1223587	90.08	1223840	90.11	1223840	90.11	1227566	0.38					
250	5	2410684.1	2419658	0.37	2411108	0.02	2412329	0.07	2413125	0.10	2412441	0.07					
	50	200391.1	197590	-1.40	197976.2	-1.20	197632.9	-1.37	197692.2	-1.34	197588.3	-1.40					
	20	408787.6	410011.9	0.30	408808.8	0.01	408717.6	-0.02	408977	0.05	410189.9	0.34					
	10	766083.3	771066.9	0.65	766611.5	0.07	766616.3	0.07	766751.3	0.09	769795.3	0.48					
	5	1500296.5	1506830	0.44	1500489	0.01	1500543	0.02	1500668	0.02	1502025	0.12					
Global Av. RPD			0.002			5.816			14.999			14.998			-0.103		

Table 11: Results of the studied algorithms for the UC3 instances.

n	m	CPLEX	GA		GRASP		MultiS		Tabu		IG-RW-D1	
		Sol.	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD
1000	50	*	4071939	-	4607063	-	7019613	-	6971324	-	4071032	-
	20	9951473.7	9991976	0.41	11464520	15.20	17021480	71.04	17004110	70.87	9959041	0.08
	10	20372680	20421070	0.24	23037960	13.08	34279380	68.26	34156180	67.66	20374550	0.01
	5	42914670	42959370	0.10	47001330	9.52	67517990	57.33	67493510	57.27	38634110	0.00
	50	1086830	1086342	-0.04	1179518	8.53	1336358	22.96	1339970	23.30	1085778	-0.10
500	20	2535361	2552478	0.68	2751157	8.51	3208921	26.57	3198752	26.17	2540576	0.21
	10	5121221.8	5145794	0.48	5277252	3.05	7024595	37.16	7041798	37.50	5122909	0.03
	5	10652000	10673610	0.20	10659560	0.07	14704310	38.05	14613490	37.19	10652360	0.00
	50	433830.9	432616.4	-0.28	438741.6	1.13	437591.9	0.87	436591.7	0.64	432544.8	-0.30
	20	958408.6	968057.4	1.07	960250.7	0.19	962368.4	0.41	962279.5	0.40	962721.2	0.45
300	10	1891772.6	1903944	0.64	1892705	0.05	1899072	0.39	1896938	0.27	1892871	0.06
	5	3893283	3907244	0.36	3893540	0.01	4013626	3.10	4027654	3.45	3893603	0.01
	50	311330.3	310129.2	-0.39	311322.3	0.00	310754.5	-0.18	310754.5	-0.18	309971.9	-0.44
	20	672272.6	680170.5	1.17	674050.2	0.26	674648.4	0.35	674648.4	0.35	676637.7	0.65
	10	672272.6	1317151	0.86	1306443	0.04	1306832	0.07	1306832	0.07	1307149	0.09
200	5	2696483	2709672	0.49	2698894	0.09	2699502	0.11	2700191	0.14	2698980	0.09
	50	204683.1	204624	-0.03	205422.9	0.36	205008.5	0.16	205254.9	0.28	204665.7	-0.01
	20	426070.1	431251.4	1.22	426946	0.21	427886.4	0.43	428385.1	0.54	429454.3	0.79
	10	820636.5	829402.7	1.07	821385.2	0.09	821373.8	0.09	821755.2	0.14	821667.2	0.13
	5	1688052.9	1697789	0.58	1688118	0.00	1688251	0.01	1688406	0.02	1688257	0.01
Global Av. RPD			0.464		3.179		17.219		17.162		0.093	

Table 12: Results of the studied algorithms for the UC4 instances.

n	m	CPLEX	GA		GRASP		MultiS		Tabu		IG-RW-D1	
		Sol.	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD	Sol.	RPD
1000	50	*	730567.2	-	8206747	-	14678010	-	14144790	-	720833.4	-
	20	3280547.9	3297686	0.52	19657160	498.76	35581280	985.11	35298260	976.22	3278131	-0.07
	10	11342840	11380820	0.34	37045240	227.01	74212030	554.36	72928710	543.05	11341330	-0.01
	5	39641030	39687760	0.23	78862350	99.37	154252300	289.75	152253900	284.69	39641340	0.12
	50	218092.8	207665.4	-4.78	1638327	651.75	1192197	445.82	1126065.1	416.31	202680.8	-7.06
500	20	856585	866907.2	1.21	2917333	239.87	3178859	271.78	3429552	300.90	855383.1	-0.14
	10	2884168.6	2904045	0.69	4885861	69.19	11292821	292.00	10994340	281.65	2883910	-0.01
	5	9591324.5	9619952	0.30	10367767	8.17	27263140	184.64	27682160	189.00	9591595	0.00
	50	94635.7	89250.6	-5.62	107165.2	13.26	102930.6	8.75	119149.8	26.16	86111.3	-8.93
	20	325405.5	330998.9	1.73	325510.3	0.03	335057.5	2.96	341159	4.83	324884	-0.16
300	10	1044770	1055481	1.02	1045415	0.06	1086566	3.99	1065131	1.96	1044740	0.00
	5	3441360.8	3457141	0.08	3441817	-0.37	3958431	14.53	3733453	7.95	3441397	-0.38
	50	70199.9	65739.4	-6.28	63073.7	-10.09	63559.9	-9.39	63740.1	-9.16	62767.2	-10.53
	20	246102.7	251679.7	2.27	246199.7	0.04	246544.1	0.18	246544.1	0.18	245690.1	-0.17
	10	709430.8	717402.6	1.13	709620.7	0.03	710006.3	0.08	710006.3	0.08	709350.8	-0.01
200	5	2517440.1	2530354	0.51	2517609	0.01	2518102	0.03	2518604	0.05	2517514	0.00
	50	50263.4	46597.3	-7.23	44819.7	-10.78	45028.1	-10.36	45047.2	-10.32	44585.5	-11.24
	20	150662.1	154223.6	2.35	150500.7	-0.11	150667.7	0.01	151266	0.41	150309.8	-0.23
	10	464978.8	472775.2	1.68	465113	0.03	465441.8	0.10	465929.6	0.21	464974.9	0.00
	5	1603202.8	1614497	0.70	1603431	0.01	1603445	0.01	1603625	0.03	1603274	0.00
Global Av. RPD			-0.482		94.012		159.703		158.641		-2.043	

Table 13: Results of the studied algorithms for the UC5 instances.

IG-RW-D1 is on average better than CPLEX for these instance sets. For what concerns the UC4 set, the average RPD is very close to zero. This lets us conclude that IG-RW-D1 is very competitive with CPLEX while using a much lower runtime.

- Moreover, IG-RW-D1 reaches the best average result for 61 out of 80 instance types. Especially noteworthy are the results for instances with a rather large number of jobs and/or machines.
- Other metaheuristics such as MultiS and Tabu exhibit large variations in their performance depending on the intervals considered, especially for large instances. This fact shows that dealing with different intervals can modify significantly the complexity of the problem for certain metaheuristics.
- We can conclude that the performance of IG-RW-D1 on uncorrelated instances is superior to the other metaheuristics independently of the intervals used for the processing time values and the weight values.

4.3.4. Global summary by means of statistical tests

So far, in this section, we have analysed the results of different algorithms using the average results obtained on the different types of instances. In this section, we want to confirm that the observed differences are statistically significant.

In order to perform a statistical analysis, we use the average result obtained by each metaheuristic for the 140 different instance types—that is, 20 instance types in each of the seven result tables—employed in previous sections. In the first place, we apply Iman and Davenport’s test to check the existence of performance differences between all the considered algorithms. Iman-Davenport finds significant performance differences between the considered algorithms because its statistical value, 67.41, is greater than its critical one, 2.39, with a significance level $\alpha = 0.05$. Then, we apply Holm’s test to compare the best ranked algorithm (IG-RW-D1, see Table 14) with the remaining algorithms.

As can be seen from the results of Holm’s test in Table 14, IG-RW-D1 obtains statistically better results than its competitors for the set of 140 different instance types contemplated in this work. In summary, this experimental analysis confirms that IG-RW-D1 is a very attractive alternative to the existing approaches for the UPMS problem.

Metaheuristic	Ranking	Holm
IG-RW-D1	1.64	Winner
GRASP	2.51	+
GA	3.49	+
MultiS	3.65	+
Tabu	3.71	+

Table 14: Comparison, using Holm’s test, over the whole set of instances.

5. Conclusions

This paper has proposed an IG algorithm (IG-RW-D1) to deal with the UPMS problem. The computational experiments performed show that: 1) the proposed IG-RW-D1 is very competitive with other state-of-the-art algorithms for the UPMS problem; specifically, significant improvements were obtained for large size problem instances (involving a large number of jobs or machines) and 2) IG-RW-D1 has been tested on a great number of different kinds of instances, proving to be very robust. Moreover, from the methodological point of view, IG-RW-D1 presents two novel elements that can help to improve the performance of IG on other optimisation problems. In the first place, the IG-RW-D1 acceptance criterion including randomness has proved to be quite useful to improve its performance, showing that this kind of acceptance criterion can help to improve the behaviour of IG algorithms for other problems. Secondly, the heuristic strategy for the destruction step has allowed improving the results of IG-RW-D1 on certain types of difficult instances. This way, we think that using more elaborated strategies in the destructive step than the classical random one should be consider when dealing with complex optimisation problems thorough IG algorithms.

We believe that the IG algorithm presented in this paper is a significant contribution, worthy of future study. We will intend to explore two interesting avenues of research. Firstly, to include memory-based mechanisms in IG models that can help to face large size instances. Secondly, to adapt our IG approach for its application to other variants of parallel machine scheduling problem dealing with constraints such as jobs release dates and due times.

6. Acknowledgements

This work was supported by grants TIN2007-66523 and TIN2011-24124 of the Spanish government and by grant P08-TIC-4173 of the Andalusian regional government. Moreover, Christian Blum acknowledges support from the *Ramón y Cajal* program of the Spanish Ministry of Economy and Competitiveness.

Appendix A. Holm’s Test

Iman and Davenport test is applied in the first place. The rejection of the null hypothesis does not involve the detection of the existing differences among the algorithms compared. It only inform us about the presence of differences among all samples of results compared. In order to conduct pairwise comparisons within the framework of multiple comparisons, we can proceed with a post-hoc procedure. In this case, a control algorithm is chosen (the configuration that reaches the best averaged ranking on the problems considered). Then, the post-hoc procedures proceed to compare the control algorithm with the remaining algorithms. In order to do this, Holm’s test is used. It computes p -value for each comparison. We will denote the p -values ordered by p_1, p_2, \dots, p_{k-1} in the way that $p_1 \leq p_2 \leq \dots \leq p_{k-1}$. Holm’s test compares then each p_i with $\alpha/(k-i)$, where α is 0.05 and k is the number of configurations compared, starting from the most significant p -value. If p_1 is below than $\alpha/(k-1)$, the corresponding hypothesis is rejected and it leaves us to compare p_2 with $\alpha/(k-2)$. If the second hypothesis is rejected, we continue with the process. As soon as a certain hypothesis can not be rejected, all the remaining hypotheses are maintained as supported. In this case, the hypothesis to be rejected is there is no significant difference between both configurations. It is important to note that p -value, for each comparison, is related to the averaged ranking obtained by this configuration over the problems considered. The lower p -value is, the worse the ranking obtained is.

Appendix B. Wilcoxon Matched-Pairs Signed-Ranks Test

Wilcoxon’s test is used for answering this question: *do two samples represent two different populations?* It is a non-parametric procedure employed in a hypothesis testing situation involving a design with two samples. It is the analogous of the paired t-test in non-parametrical statistical procedures; therefore, it is a pairwise test that aims to detect significant differences between the behaviour of two algorithms.

The null hypothesis for Wilcoxon’s test is $H_0 : \theta_D = 0$; in the underlying populations represented by the two samples of results, the average of the difference scores equals zero. The alternative hypothesis is $H_1 : \theta_D \neq 0$, but also can be used $H_1 : \theta_D > 0$ or $H_1 : \theta_D < 0$ as directional hypothesis.

In the following, we describe the tests computations. Let d_i be the difference between the performance scores of the two search algorithms on i -th out

of N functions. The differences are ranked according to their absolute values (We have previously normalised the results of every algorithm on each test problem into the interval $[0, 1]$, taking into consideration highest and lowest fitness values achieved on each test problem); average ranks are assigned in case of ties. Let R^+ be the sum of ranks for the functions on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$\begin{aligned}
 R^+ &= \sum_{d_i > 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i) \text{ and} \\
 R^- &= \sum_{d_i < 0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i = 0} \text{rank}(d_i)
 \end{aligned}
 \tag{B.1}$$

Let T be the smallest of the sums, $T = \min(R^+, R^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for N degrees of freedom (Table B.12 in [45]), the null hypothesis of equality of means is rejected.

The obtaining of the p-value associated to a comparison is performed by means of the normal approximation for the Wilcoxon T statistic (Sect. VI, Test 18 in [49]). Furthermore, the computation of the p-value for this test is usually included in well-known statistical software packages (SPSS, SAS, R, etc.).

References

- [1] Rosenbloom E, Goertzen N. Cyclic nurse scheduling. *European Journal of Operational Research* 1987;31:19–23.
- [2] Buxey G. Production scheduling: Practice and theory. *European Journal of Operational Research* 1989;39:17–31.
- [3] Dodin B, Chan KH. Application of production scheduling methods to external and internal audit scheduling. *European Journal of Operational Research* 1991;52(3):267–79.
- [4] Pendharkar P, Rodger J. Nonlinear programming and genetic search application for production scheduling in coal mines. *Annals of Operations Research* 2000;95(1):251–67.
- [5] Jarrah AIZ, Bard JF, de Silva AH. A heuristic for machine scheduling at general mail facilities. *European Journal of Operational Research* 1992;63(2):192–206.
- [6] Rochat Y. A genetic approach for solving a scheduling problem in a robotized analytical system. *Journal of Heuristics* 1998;4:245–61.
- [7] Croce FD, Tadei R, Asioli P. Scheduling a round robin tennis tournament under courts and players availability constraints. *Annals of Operations Research* 1999;92:349–61.
- [8] Brucker P, Hurink J. Solving a chemical batch scheduling problem by local search. *Annals of Operations Research* 2000;96(1):17–38.
- [9] Bruno J, Coffman E, Sethi R. Scheduling independent tasks to reduce mean finishing time. *Communications ACM* 1974;17(7):382–7.
- [10] Lenstra J, Rinnooy-Kan A, Brucker P. Complexity of machine scheduling problems. In: P.L. Hammer E.L. Johnson BK, Nemhauser G, editors. *Studies in Integer Programming*; vol. 1 of *Annals of Discrete Mathematics*. Elsevier; 1977, p. 343–62.
- [11] McNaughton R. Scheduling with deadlines and loss functions. *Management Science* 1959;6(1):1–12.

- [12] Azizoglu M, Kirca O. On the minimization of total weighted flow time with identical and uniform parallel machines. *European Journal of Operational Research* 1999;113(1):91–100.
- [13] Azizoglu M, Kirca O. Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. *IIE Transactions* 1999;31(2):153–9.
- [14] Chen ZL, Powell W. Solving parallel machine scheduling problems by column generation. *Inform Journal on Computing* 1999;11(1):78–94.
- [15] Li K, Yang SL. Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Applied Mathematical Modelling* 2009;33(4):2145–58.
- [16] Vredeveld T, Hurkens C. Experimental comparison of approximation algorithms for scheduling unrelated parallel machines. *Inform Journal on Computing* 2002;14(2):175–89.
- [17] Lin Y, Pfund M, Fowler J. Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Computers & Operations Research* 2011;38(6):901–16.
- [18] Rodriguez F, Blum C, García-Martínez C, Lozano M. GRASP with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times. *Annals of Operations Research* 2012;In Press.
- [19] Fanjul-Peyro L, Ruiz R. Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Computers & Operations Research* 2011;38(1):301–9.
- [20] Culberson JC, Luo F. Exploring the k-colorable landscape with iterated greedy. In: *Dimacs Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society; 1996, p. 245–84.
- [21] Jacobs L, Brusco M. A local-search heuristic for large set-covering problems. *Naval Research Logistics* 1995;42:1129–40.
- [22] Ruiz R, Stützle T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research* 2007;177(3):2033–49.

- [23] Fanjul-Peyro L, Ruiz R. Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research* 2010;207(1):55–69.
- [24] Framinan J, Leisten R. A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. *OR Spectrum* 2008;30:787–804.
- [25] Lin SW, Lee ZJ, Ying KC, Lu CC. Minimization of maximum lateness on parallel machines with sequence-dependent setup times and job release dates. *Computers & Operations Research* 2011;38(5):809–15.
- [26] Urlings T, Ruiz R, Stützle T. Shifting representation search for hybrid flexible flowline problems. *European Journal of Operational Research* 2010;207(2):1086–95.
- [27] Elmaghraby S, Park S. Scheduling jobs on a number of identical machines. *AIIE Transactions* 1974;6(1):1–13.
- [28] Azizoglu M, Kirca O. On the minimization of total weighted flow time with identical and uniform parallel machines. *European Journal of Operational Research* 1999;113(1):91–100.
- [29] Allahverdi A, Gupta J, Aldowaisan T. A review of scheduling research involving setup considerations. *Omega* 1999;27(2):219–39.
- [30] Weng M, Lu J, Ren H. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics* 2001;70(3):215–26.
- [31] Alidaee B, Rosa D. Scheduling parallel machines to minimize total weighted and unweighted tardiness. *Computers & Operations Research* 1997;24(8):775–88.
- [32] Koulamas C. Decomposition and hybrid simulated annealing heuristics for the parallel-machine total tardiness problem. *Naval Research Logistics* 1997;44(1):109–25.
- [33] Cheng R, Gen M, Tozawa T. Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithms. *Computers & Industrial Engineering* 1995;29(1-4):513–7.

- [34] Glass CA, Potts CN, Shade P. Unrelated parallel machine scheduling using local search. *Mathematical and Computer Modelling* 1994;20(2):41–52.
- [35] Chen CL, Chen CL. Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology* 2009;43(1):161–9.
- [36] Anghinolfi D, Paolucci M. Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Computers & Operations Research* 2007;34(11):3471–90.
- [37] Zaidi M, Jarboui B, Loukil T, Kacem I. Hybrid meta-heuristics for uniform parallel machine to minimize total weighted completion time. In: *Proc. of 8th International Conference of Modeling and Simulation (MOSIM10)*. 2010,.
- [38] Waligora G. Tabu search for discrete-continuous scheduling problems with heuristic continuous resource allocation. *European Journal of Operational Research* 2009;193(3):849 –56.
- [39] Ying KC, Cheng HM. Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. *Expert Systems with Applications* 2010;37(4):2848–52.
- [40] Lozano M, Molina D, García-Martínez C. Iterated greedy for the maximum diversity problem. *European Journal of Operational Research* 2011;214(1):31 –8.
- [41] Ghirardi M, Potts C. Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. *European Journal of Operational Research* 2005;165(2):457–67.
- [42] Gallego M, Laguna M, Marti R, Duarte A. Tabu search with strategic oscillation for the maximally diverse grouping problem. *Journal of Operational Research Society* 2011;In Press.
- [43] Garcia S, Molina D, Lozano M, Herrera F. 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* 2008;15:617–44.

- [44] Friedman M. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics* 1940;11(1):86–92.
- [45] Zar J. *Biostatistical Analysis*. Prentice Hall; 1999.
- [46] Iman R, Davenport J. Approximations of the critical region of the Friedman statistic. In: *Communications in Statistics*. 1980, p. 571–95.
- [47] Holm S. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 1979;6:65–70.
- [48] Wilcoxon F. Individual comparisons by ranking methods. *Biometrics* 1945;1:80–3.
- [49] Sheskin D. *The Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC; 2000.

5. Tabu Search with Strategic Oscillation for the Quadratic Minimum Spanning Tree

The journal paper associated to this part is:

- M. Lozano, F. Glover, C. García-Martínez, F.J. Rodríguez, R. Martí, Tabu Search with Strategic Oscillation for the Quadratic Minimum Spanning Tree. **IEE Transactions. Submitted on second revision.**
 - Status: **Submitted on second revision**
 - Impact Factor (JCR 2011): 0.856.
 - Subject Category: Operations Research & Management Science. Ranking 39 / 77 (Q3).
 - Subject Category: Engineering, Industrial. Ranking 22 / 43 (Q3).

Tabu Search with Strategic Oscillation for the Quadratic Minimum Spanning Tree

Manuel Lozano

Department of Computer Science and Artificial Intelligence,
University of Granada, Granada 18071, Spain, lozano@decsai.ugr.es

Fred Glover

OptTek Systems, Boulder (Co), USA, glover@opttek.com

Carlos García-Martínez

Department of Computing and Numerical Analysis,
University of Córdoba, Córdoba 14071, Spain, cgarcia@uco.es

Francisco Javier Rodríguez

Department of Computer Science and Artificial Intelligence,
University of Granada, Granada 18071, Spain, fjrodriguez@decsai.ugr.es

Rafael Martí

Department of Statistics and Operations Research
University of Valencia, Valencia, Spain, rmarti@uv.es

Abstract

The quadratic minimum spanning tree problem consists of determining a spanning tree that minimizes the sum of costs of the edges and pairs of edges in the tree. Many algorithms and methods have been proposed for this hard combinatorial problem, including several highly sophisticated metaheuristics. In this paper we present a simple tabu search (TS) for this problem that incorporates strategic oscillation (SO) by alternating between constructive and destructive phases. We show commonalities shared by this strategy and the more recently introduced methodology called iterated greedy search, and identify implications of their differences regarding the use of memory structures. Extensive computational experiments reveal that the proposed SO algorithm with embedded TS is highly effective for solving complex instances of the problem as compared to the best metaheuristics in the literature. We also introduce a hybrid method that proves similarly effective for problem instances that are both simple and complex.

Keywords. Tabu search, strategic oscillation, adaptive memory programming, Quadratic minimum spanning tree, iterated greedy.

1 Introduction

The Quadratic Minimum Spanning Tree Problem (QMSTP) has been widely studied in the literature due to its applications in a wide variety of settings, including transportation, telecommunication, irrigation, and energy distribution. The problem appears, for example, when transferring oil from one pipe to another in a situation where the cost depends on the type of interface between two pipes. The same pairwise interaction effect arises in the connection of aboveground and underground cables in a road network with turn penalties [23, 28].

We may define the QMSTP as follows. Let $G = (V, E)$ be an undirected graph where $V = \{v_1, \dots, v_n\}$ is the vertex set and $E = \{e_1, \dots, e_m\}$ is the edge set. Consider that each edge and each pair of edges has an associated cost. In mathematical terms, we have two cost functions: $w : E \rightarrow \mathbb{R}^+$ and $c : (E \times E) - \{(e, e), \forall e \in E\} \rightarrow \mathbb{R}^+$ where as in previous approaches [2, 19], we assume that $c(e_i, e_j) = c(e_j, e_i)$ for $i, j = 1, \dots, m$. The QMSTP consists of finding a spanning tree T of G with edge set $\xi(T) \subseteq E$ that minimizes:

$$\sum_{e_i \in \xi(T)} \sum_{\substack{e_j \in \xi(T) \\ e_i \neq e_j}} c(e_i, e_j) + \sum_{e \in \xi(T)} w(e).$$

The QMSTP is an extension of the well-known minimum spanning tree problem, where in addition to edge costs, we have costs associated with pairs of edges. The problem was first introduced by Assad and Xu [1, 25], who showed that it is NP-hard.

This paper has two objectives: the primary objective is to investigate the strategic oscillation proposal to alternate between constructive and destructive phases as a basis for creating a competitive method for the QMSTP, and the secondary objective is to compare memory-less with memory based designs. The remainder of this paper is organized as follows. In Section 2, we give a background of Tabu Search (TS) and Strategic Oscillation (SO) that sets the stage for the later specific algorithmic design we employ. We also refer to the more recent Iterated Greedy (IG) approach which has some resemblances to strategic oscillation, and which provides a basis for subsequent comparative testing. Section 3 gives an overview of particular metaheuristics that have previously been applied to the QMSTP. Section 4 describes our proposed methods based on TS, SO and IG. In Section 5, we present empirical studies, which are designed to: 1) analyze the influence of the parameters and settings of our methods, 2) compare the different designs, paying special attention to the influence of memory structures, 3) obtain an algorithm with a robust performance across test problems with different characteristics, and 4) compare its results with those of the best approaches from the literature. We finish with the associated conclusions in Section 6 and our proposals for further extensions.

2 Background of Tabu Search and Strategic Oscillation

Tabu Search is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. One of the main components of Tabu Search is its use of adaptive memory, which creates a highly flexible search behavior. Memory-based strategies which are the hallmark of tabu search approaches, are founded on a quest for “integrating principles,” by which alternative forms of memory are appropriately combined with effective strategies for exploiting them. The adaptive memory feature of TS allows the implementation of procedures that are capable of searching the solution space economically and effectively. Since local choices are guided by information collected during the search, TS contrasts with memoryless designs that heavily rely on semi-random processes that implement a form of sampling.

The structure of a neighborhood in tabu search goes beyond that used in local search by embracing the types of moves used in constructive and destructive processes (where the foundations for such moves are accordingly called constructive neighborhoods and destructive neighborhoods). Following basic TS principles, memory structures can be implemented within a constructive process to favor (or avoid) the inclusion of certain elements in the solution previously identified as attractive (or unattractive). Such expanded uses of the neighborhood concept reinforce a fundamental perspective of TS, which is to define neighborhoods in dynamic ways that can include serial or simultaneous consideration of multiple types of moves.

This dynamic neighborhood approach applies not only to the types of neighborhoods used in “solution improvement methods” (sometimes called “local search methods”) but also applies to constructive neighborhoods used in building solutions from scratch - as opposed to transitioning from one solution to another. Although it is commonplace in the metaheuristic literature to restrict the word “neighborhood” to refer solely to transitions between solutions as embodied in improvement methods, constructive neighborhoods have been proposed as an important ingredient of search processes from the very beginning of the TS methodology, as documented by Glover and Laguna [7]. Nevertheless, tabu search methods for exploiting constructive neighborhoods have rarely been applied in computational studies.

Our tabu search approach for the QMSTP is additionally based on the strategic oscillation methodology [7, 8]. Strategic oscillation (SO) is closely linked to the origins of tabu search, and operates by orienting moves in relation to a critical level, as identified by a stage of construction. In particular, we consider a constructive/destructive type of strategic oscillation, where constructive steps “add” elements and destructive steps “drop” elements. As described in Chapter 4 of [8], the alternation of constructive with destructive processes, which strategically dismantle and then rebuild successive trial solutions, affords an enhancement of such traditional constructive procedures.

More recently, constructive and destructive neighborhoods have been applied within a simplified and effective method known as Iterated Greedy (IG) [10], which generates a sequence of solutions by iterating over a greedy constructive heuristic using two main phases: *destruction* and *construction*. IG is a memory-less metaheuristic easy to implement that has exhibited state-of-the-art performance in some settings (see for example [4, 12, 21]). We identify links between this more recent constructive/destructive approach and strategic oscillation by first proposing an adaptation of the IG methodology to the QMSTP and then extending it to include short term memory structures to create a tabu search approach based on strategic oscillation.

3 Previous Metaheuristics Applied to the QMSTP

Although exact procedures, including branch and bound [1, 25] and Lagrangian relaxation [14], have been applied to the QMSTP, their success has been chiefly limited to problems that may be classified as small, containing from 6 to 15 vertices. By contrast, metaheuristics have been shown capable of finding high quality solutions to QMSTP problems over a wide range of sizes, from small to significantly larger than those handled by exact methods. Early genetic algorithms (GAs) implementations were made by Zhou and Gen [28], which proposed a GA based on the Prüfer number to encode a spanning tree. The algorithm was tested on QMSTP instances with up to 50 vertices and proved superior to two greedy algorithms proposed in [1, 25]. More recently, Soak, Corne, and Ahn [22] developed another GA approach that employed an edge-window-decoder encoding, (in which a tree is represented as a linear string of node identifiers, and these in turn are interpreted as a set of edges). This GA implementation was demonstrated to perform much better than GAs based on other encodings (including the Prüfer number representation) on Euclidean instances containing between 50 and 100 nodes. In related work, Gao and Lu [5] proposed another GA to address a QMSTP variant, the fuzzy QMSTP, which was formulated as an expected value model with chance-constrained programming and dependent-chance constrained programming. The authors suggested the use of Prüfer number representation for the GA.

The Artificial bee colony method is another bio-inspired metaheuristic that has been a tool of choice for dealing with the QMSTP [23], involving a swarm intelligence technique based on the analogy to the intelligent foraging behavior of honey bees. To investigate the effectiveness of this approach, the authors carried out experiments with problem instances of up to $n = 250$, finding that artificial bee colony obtained better solutions than those achieved by the GA-based methods of Zhou and Gen [28] and Soak et al [22].

Recently, Cordone and Passeri [2] proposed a tabu search implementation for the QMSTP based on exchanges. At each iteration, the method adds a new edge to the current spanning tree and removes one of the edges from the resulting loop. The tabu mechanism forbids recently removed edges to be added into the solution and recently added edges to be removed for a certain number of iterations. The algorithm was tested over a benchmark set of randomly generated instances with up to 30 vertices. However, no comparison with previous metaheuristics is presented.

In the context of adaptive memory programming (the use of memory structures within metaheuristics), Öncan and Punnen [14] presented a local search algorithm with tabu thresholding. The algorithm was tested on different benchmark problems with sizes ranging from 6 to 100 vertices. Empirical results evidenced that the method improves upon the greedy algorithms presented in [1, 25]. Finally, the most recent TS proposal is the highly effective Iterated Tabu Search approach, proposed by Palubeckis [19] using two phases: solution perturbation and TS. The method to perturb a solution (current spanning tree) relies on randomization: an edge for removal and a non-tree edge for replacement are randomly selected from a candidate list. At each iteration, the TS method examines all feasible exchange moves (replacement of an edge in the tree) and the best admissible non-tabu move is performed. The removed and added edges are labeled tabu for $\frac{m}{4}$ iterations. The TS method is run for a number of iterations taken from the interval $[I_{min}, I_{max}]$. Computational results for problem instances with up to

50 vertices indicated that its performance (in terms of both solution quality and computational time) surpasses those of the other metaheuristics included in their comparison, which consisted of multistart simulated annealing, GA, and GA with local search.

4 New Metaheuristic Adaptations

In this section, we explore the adaptation of the TS and IG methodologies to obtain high quality solutions to the QMSTP. We first describe in Section 4.1 the greedy constructive and destructive algorithms that will be applied in both methods. Then, in Section 4.2, we provide details of the improvement methods proposed in the literature for the QMSTP, which will be employed as the local search phase of our IG for the QMSTP. Moreover, we describe the short term memory structure that added to the local search creates the tabu search improvement method of our TS for the QMSTP. Section 4.3 gives the entire IG method, showing how its different elements interact; and finally, Section 4.4 completes the development by providing the general overview of the TS algorithm with the Strategic Oscillation strategy.

4.1 Constructive and Destructive Methods

We begin by introducing two greedy constructive algorithms, C1 and C2, and a destructive one, D, which may be used to build feasible solutions for the QMSTP. The two constructive methods accomplish their task by adding, at each construction step, exactly one edge to a current partial solution, which is a forest of G (disjoint union of trees). In order to describe C1 and C2, we firstly define the contribution of an edge $e_i \in E$ to the total cost of a forest F with edge set $\xi(F) \subseteq E$ as:

$$Q(e_i, F) = w(e_i) + \sum_{e_j \in \xi(F)} c(e_i, e_j), \quad i = 1, \dots, m.$$

C1 is an iterative process that is similar to Kruskal's algorithm [11]. At each iteration, the algorithm considers the set $S \subseteq E \setminus \xi(F)$ of all edges that can be used to feasibly augment the current forest F (i.e., those edges whose inclusion would not result in a cycle) and adds the feasible edge with the minimum contribution value across all elements in S , e_{min} . We can easily update $Q(e_i, F)$ for each e_i , $i = 1, \dots, m$ by adding the value $c(e_i, e_{min})$ to it. We should point out that given a spanning tree T , the objective function value z can be obtained from the Q values with the expression:

$$z = \sum_{e_i \in \xi(T)} Q(e_i, T).$$

The greedy constructive procedure C2 is based on the sequential fixing method proposed by Assad and Xu [1, 25]. It proceeds like C1 but employing the following greedy function to guide the iterative selection of edges (again, the lowest values are preferable):

$$Q_2(e_i, F) = Q(e_i, F) + \frac{n_1}{m_1} \cdot \sum_{\substack{e_j \in S \\ i \neq j}} c(e_i, e_j), \quad i = 1, \dots, m.$$

where $m_1 = |S| - 1$ and $n_1 = n - 1 - |\xi(F)|$. Note that Q_2 extends Q by further considering interaction costs with the candidates edges for the tree in S (in this way, C2 is more complicated than C1).

The greedy destructive algorithm is based on the reverse-delete algorithm (reverse version of Kruskal's algorithm). The algorithm starts with the original graph G and then it removes one edge at a time until there are only $n - 1$ selected edges remaining. Specifically, it deletes the edge e with the maximum contribution to the current graph that does not disconnect the graph, e_{max} . In this case, the Q values for each e_i ($i = 1, \dots, m$) are updated by subtracting $c(e_i, e_{max})$ from them. In this method, we follow a lexicographical order for tie-breaking.

4.2 Local Search Methods

In this section, we describe three improvement methods, based on exchanges, previously proposed for the QMSTP. Given a solution T , we exchange an edge $e_i \in \xi(T)$ with an edge $e_j \in E \setminus \xi(T)$ resulting on a tree T' with $\xi(T') = \xi(T) \cup \{e_j\} \setminus \{e_i\}$. Note that when we delete e_i from T , the tree is partitioned into two components and therefore, the edge e_j is selected from the set of edges connecting these two components of T in order to obtain a new feasible solution (tree T'). We observe that we can efficiently evaluate an exchange move without recomputing the objective function from scratch. Let z be the value of the objective function of solution T . The value z' of the new solution T' may be directly computed as:

$$z' = z - 2 \cdot Q(e_i, T) + w(e_i) + 2 \cdot Q(e_j, T) - w(e_j) - 2 \cdot c(e_i, e_j).$$

In this way, the move value of the solutions in the neighborhood of a given solution can be quickly evaluated, and once a move is chosen, the values $Q(e_k, T')$, $k = 1, \dots, n$, may be calculated as follows:

$$Q(e_k, T') = Q(e_k, T) - c(e_k, e_i) + c(e_k, e_j).$$

Based on the neighborhood defined by the exchanges above, we have studied three local search methods:

- *Best-improvement local search* (BL). This is the classical local search method in which at each iteration we explore the entire neighborhood and perform the best move. In other words, the *best choice* strategy selects the move with the largest move value among all the moves in the neighborhood. If it improves the current solution, we apply it and examine in the next iteration the neighborhood of the new solution. Otherwise, the method stops.
- *First-improvement local search* (FL) [19]. This method implements the so-called *first choice* strategy that scans the moves in the neighborhood in search for the first exchange yielding an improvement in the objective function. At each iteration, an edge is randomly selected from the current solution ($e_i \in \xi(T)$) to be removed from it. Then, we examine the edges out of the solution ($e_j \in E \setminus \xi(T)$) that can be added to $\xi(T) \setminus \{e_i\}$, producing a new solution (spanning tree). We randomly select an edge e_j and evaluate the associated move. If it is an improving move, we perform it; otherwise, we randomly select a new edge e_j until we find an improving move or all the e_j edges have been explored. In the latter case we simply proceed to the next iteration in which a new edge e_i is randomly selected for removal. The procedure terminates when no further improvement is possible.
- *ABC local search* (AL) [23]. This is a hybrid method between the BL and the FL described above. At each iteration, this local search randomly selects an edge from the current solution ($e_i \in \xi(T)$) to be removed from it. Then, it evaluates all the possible edges that can be added to $\xi(T) \setminus \{e_i\}$ and performs the best one if it improves the current solution. The procedure terminates when no further improvement is possible.

These three methods represent typical implementations of a local search process. They basically resort to randomization or exhaustive exploration to select a move in the neighborhood of a solution. Based on AL, we now propose a *short term memory tabu local search* (TLS) in which we add a memory structure and apply candidate list strategies for move selection.

For each edge in the solution, $e_i \in \xi(T)$, we can compute a measure, $m(e_i)$ of its contribution to the solution value:

$$m(e_i) = w(e_i) + \sum_{\substack{e_j \in \xi(T) \\ e_j \neq e_i}} c(e_i, e_j).$$

These measures are mapped onto probabilities for move selection. Specifically, at each iteration of TLS, we probabilistically select an edge in the solution to be removed. The probabilities are computed from the measure above, where the larger the contribution the larger the probability to be selected (the probability of selecting edge e_i is proportional to its contribution $m(e_i)$). Then, as in AL, we evaluate all the possible edges that can be added to $\xi(T) \setminus \{e_i\}$, but here we select the best one that is not tabu, regardless of whether it improves the current solution (i.e., the move is executed even when the move value is not positive, resulting in a deterioration of the current objective function value). Then, we update the measure values and perform a new iteration. The moved edge(s) become tabu for *TabuTenure* iterations, and therefore they cannot be selected for addition or removal during this time.

To implement our memory structure, we employ a one-dimensional array $tabu(e)$, initially set to $-TabuTenure$ to permit initial selections, to store the iteration number when edge e is moved. That is, if edge e_i is removed from the solution and e_j is added to the solution at iteration $iter$, then $tabu(e_i) = tabu(e_j) = iter$. Then, in a subsequent iteration $iter_2$, we say that an edge e is tabu (and cannot be added to or removed from the solution) if

$$iter_2 - tabu(e) < TabuTenure.$$

Note that although we use the same tenure value for both edges, an interesting variant is to use a different tenure value for e_i than for e_j . In our experiments, however, we determined that the effect of using different tenure values does not justify the increase in complexity related to calibrating an additional search parameter.

4.3 Iterated Greedy

From an initial solution the IG method alternates between destructive and constructive phases just as strategic oscillation does. During the destructive phase, some solution components are removed from a previously constructed solution. The construction procedure then applies a greedy constructive heuristic to reconstruct a solution. Once a newly reconstructed solution has been obtained, an acceptance criterion is applied to decide whether it will replace the incumbent solution. Additionally, an optional local search phase for improving the reconstructed solution can be applied for improved outcomes.

An outline of the proposed IG is depicted in Figure 1. It starts from a complete initial solution T (*Initialise()*; Step 1) and then iterates through a main loop which first generates a partial candidate solution F by removing a fixed number of edges from the complete candidate solution T (*Destruction-phase*(T, n_d); Step 4) and next reconstructs a complete solution T_c starting with F (*Construction-phase*(F); Step 5). In the local search phase (*Local-Search-phase*(T_c); Step 6), an improvement procedure is performed in order to find better solutions near the reconstructed solution. Before continuing with the next loop, an acceptance criterion (*AcceptCriterion*(T, T_i); Step 10) decides whether the solution returned by the local search procedure, T_i , becomes the new incumbent solution. The process iterates through these phases until a computation limit t_{max} is reached. The best solution, T_b , generated during the iterative process is kept to provide the final result.

In the algorithm in Figure 1 we apply the greedy destructive algorithm D (see Section 4.1) to obtain the initial solution *Initialise()*. Then we can apply either C1 or C2 constructive algorithms (Section 4.1) to implement the *Construction-phase*(F). These methods insert n_d edges into the forest resulting from the destruction phase, obtaining a feasible spanning tree. The *Destruction-phase*(T, n_d) removes n_d (a parameter of the algorithm) edges from the current solution. These edges are selected at random as in many previous IG algorithms [20, 21, 26]. Finally, we can apply any of the three local search algorithms described in Section 4.2 as the *Local-search-phase*(T_c).

We have considered two different following acceptance criteria in the scheme shown in Figure 1:

```

Input:  $G, t_{max}, n_d$ 
Output:  $T_b$ 
1  $T \leftarrow \text{Initialise}()$ ;
2  $T_b \leftarrow T$ ;
3 while  $t_{max}$  is not reached do
4    $F \leftarrow \text{Destruction-phase}(T, n_d)$ ;
5    $T_c \leftarrow \text{Construction-phase}(F)$ ;
6    $T_i \leftarrow \text{Local-Search-phase}(T_c)$ ;
7   if  $T_i$  is better than  $T_b$  then
8      $T_b \leftarrow T_i$ ;
9   end
10  if  $\text{AcceptCriterion}(T, T_i)$  then
11     $T \leftarrow T_i$ ;
12  end
13 end

```

Figure 1: Iterated greedy pseudocode

- ‘*Replace if better*’ acceptance criterion (RB). The new solution is accepted only if it provides a better objective function value [27].
- ‘*Random walk*’ acceptance criterion (RW). An IG algorithm using the RB acceptance criterion may lead to stagnation situations of the search due to insufficient diversification [20]. At the opposite extreme is the random walk acceptance criterion, which always applies the destruction phase to the most recently visited solution, irrespective of its objective function value. This criterion clearly favors diversification over intensification, because it promotes a stochastic search in the space of local optima.

4.4 Tabu Search with Strategic Oscillation

Our SO approach for the QMSTP implements a 1-sided oscillation that does not cross into infeasible space by adding more edges than necessary (which could be an interesting design to test). It thus differs from instances of SO that are organized relative to feasibility and infeasibility, involving a 2-sided oscillation that crosses that boundary.

An outline of the proposed SO is depicted in Figure 2. It starts from a complete initial solution T and then iterates through a main loop which first generates a solution T_i by a combination of a constructive procedure and a memory structure. Once a fully constructed solution is obtained (and a limiting computational time has not yet been reached), we pass to the Destructive Phase in which edges are removed from the complete candidate solution T_i . Once a Turn Around point is reached, the method goes to the next Constructive Phase.

Note that in the SO algorithm the meaning of a best move depends not only on problem context but on whether a Constructive or Destructive phase is being employed. The Turn Around point itself may oscillate by creating partial solutions containing different numbers of elements. The Constructive and Destructive processes can be interrupted at any point to apply an Improvement Process that may or may not include memory structures. Often Improvement Processes are applied only at the conclusion of a Constructive Phase, when a complete solution is obtained, but they may be applied at other points as well. Sometimes applying them earlier in a constructive process, for example, can remove deficient features that would otherwise be inherited by later construction stages and that would create undesirable solutions.

In our SO method for the QMSTP we consider a constructive phase with two parts. In the first one, we simply apply a greedy algorithm (C1 or C2, described in Section 4.1) but some elements (those labeled as tabu) are not allowed to become part of the constructed solutions. In the second part, we apply a short term tabu search (TLS) to improve the solution as described

```

Input:  $G, t_{max}, n_{tabu}$ 
Output:  $T_b$ 
1  $T \leftarrow \text{Initialise}()$ ;
2  $T_b \leftarrow T$ ;
3 while  $t_{max}$  is not reached do
    // Destructive Phase
4    $T_i \leftarrow T$ ;
5   while Current solution  $T_i$  has not reached a Turn
      Around point do
6     Select a best element to drop from  $T_i$  subject to (po-
      tential) tabu restrictions.
7     Drop it from the (partial) solution  $T_i$ 
8   end
    // Constructive Phase
9   while Current solution  $T_i$  is not complete do
10    Select a best element subject to tabu restrictions.
11    Add it to the partial solution
12  end
13   $T'_i \leftarrow \text{TLS Improvement Process}(T_i, n_{tabu})$ ;
14  if  $T'_i$  is better than  $T_b$  then
15     $T_b \leftarrow T_i$ ;
16  end
17  if AcceptCriterion( $T, T'_i$ ) then
18     $T \leftarrow T'_i$ ;
19  end
20 end

```

Figure 2: Strategic Oscillation pseudocode

in Section 4.2. In line with this, our destructive phase for the QMSTP also incorporates memory structures. As in the IG algorithm we randomly remove n_d elements from the current solution, but now, a number n_{tabu} of these elements, $n_{tabu} \leq n_d$, removed by the destruction phase, are labeled tabu and thus not allowed to be added to the solution in the following constructive phase. This strategy attempts to avoid looping over already visited solutions, allowing the algorithm to advance towards diversified promising solutions. Note that the *tabu tenure* in our memory structure is one entire iteration (destructive + constructive phases). This memory structure may become decisive to ensure that SO performs an effective search when dealing with complex problems (diversification is a key factor in this process). The level of diversity induced by this technique is controlled by the n_{tabu} parameter.

As customary in tabu search, we have implemented an *aspiration criterion* to overrule the tabu status under certain circumstances. Note that specific situations may arise where the construction of a feasible tree is not possible given the constraints imposed by this strategy. When such an outcome is detected, the reference to tabu status is directly disabled ($n_{tabu} = 0$) during the current cycle to allow the creation of the new solution. (A common alternative is to use a so-called “aspiration by default” which removes the tabu status from a certain number of the least tabu elements, measured by reference to their unexpired tabu tenures.)

5 Computational Experiments

This section describes the computational experiments that we performed to assess the performance of the IG and SO models presented in the previous sections. Firstly, we detail the experimental setup and the statistical methods applied (Section 5.1), then, we analyze the

results obtained from different experimental studies carried out with these algorithms. Our aim is: 1) to analyze the influence of the parameters and settings associated with IG, which is a memory-less based algorithm (Section 5.2), 2) to show the benefits of the use of memory structures in SO and TLS (Section 5.3), 3) to combine SO, IG, and ITS with the aim of obtaining a hybrid metaheuristic being able to show a robust operation for test problems with different characteristics (Section 5.4), and 4) to compare the results of the hybrid algorithm with those of other metaheuristic approaches for the QMSTP from the literature (Section 5.5).

5.1 Experimental Setup

The codes of all the studied algorithms have been implemented in C and the source code has been compiled with gcc 4.6. The experiments were conducted on a computer with a 3.2 GHz Intel® Core™ i7 processor with 12 GB of RAM running Fedora™ Linux V15. We considered three groups of benchmark instances for our experiments (in total constituting 83 instances), which are described below and available at <http://sci2s.ugr.es/qmst/QMSTPInstances.rar>.

- The first set of benchmarks, henceforth denoted by CP, is composed of 36 instances, ranging in size from 40 to 50 vertices, introduced by Cordone and Passeri (they are publicly available¹). For experimentation with these instances, the cutoff time for each run was 10 seconds.
- Öncan and Punnen [14] present a transformation scheme to obtain QMSTP instances from *quadratic assignment* problem (QAP) instances. They demonstrated that the optimum value of a QMSTP instance obtained with this transformation is equal to the optimum value of the corresponding QAP instance. Particularly, the authors offered two instance sets by transforming 15 QAP instances (from $n = 24$ to 60) by Nugent et al. [13] and 14 QAP instances (from $n = 24$ to 50) by Christofides and Benavent [3] (whose optimal solutions are known). They will be denoted by NUG and CHR, respectively. The experiments reported in [14] showed that these instances are particularly challenging for QMSTP algorithms. A time limit of 1000 seconds was assigned for these problems.
- The last group consists of two sets of large instances: RAND and SOAK. They were generated exactly in the same manner as in [28] and [22], respectively. All the instances in the RAND set represent complete graphs with integer edge costs uniformly distributed in $[1, 100]$. The costs between edges are also integers and uniformly distributed in $[1, 20]$. In the SOAK instances, nodes are distributed uniformly at random on a 500×500 grid. The edge costs are the integer Euclidean distance between these points. The costs between edges are uniformly distributed between $[1, 20]$. For each value of $n \in \{150, 200, 250\}$, there are 3 instances, leading to a total of 9 instances for each set. All methods were stopped using a time limit that varied according to problem size (400, 1200, and 2000 seconds for problems from $n = 150$ to 250, respectively).

Non-parametric tests [6] have been used to compare the results of the different optimization algorithms under consideration. The only condition to be fulfilled for the use of non-parametric tests is that the algorithms to be compared should have been tested under the same conditions (that is, the same set of problem instances, the same stopping conditions, the same number of runs, etc). In the first place, *average ranking* for each algorithm is firstly computed according to *Friedman's* test. This measure is obtained by computing, for each problem instance, the ranking r_a of the observed results for algorithm a assigning to the best of them the ranking 1, and to the worst the ranking $|A|$ (A is the set of algorithms). Then, an average measure is obtained from the rankings of this algorithm for all test problems. For example, if a certain algorithm achieves rankings 1, 3, 1, 4, and 2, on five problem instances, the average ranking is $\frac{1+3+1+4+2}{5} = \frac{11}{5}$. Note that the lower an average ranking is, the better its associated algorithm

¹<http://homes.dsi.unimi.it/~cordone/research/qmst.html>

is. We have considered two alternative methods based on non-parametric tests to analyze the experimental results:

- The first method is the application of the *Iman and Davenport* test and the *Holm* method as a post hoc procedure. The first test may be used to see whether there are significant statistical differences among the compared algorithms. If differences are detected, then Holm’s test is employed to compare the best algorithm (*control algorithm*) against the remaining ones.
- The second method is the utilization of the *Wilcoxon matched-pairs signed-ranks* test. With this test, the results of two algorithms may be directly compared. In statistical terms, this test answers the question: Do the two samples represent two different populations? In the context of algorithms’ comparison the Wilcoxon test determines if the results of two methods are significantly different.

5.2 Study of the Memory-less Based Method: IG

In this section, we investigate the effect of the different parameters and strategies applied in IG and their interactions. In particular, we consider:

- The acceptance criteria: RB and RW.
- The memory-less based improvement methods: FL, BL, and AL.
- The number of removed solution components $n_d=0.2n, 0.4n, 0.6n,$ and $0.8n$.

With the purpose of fine-tuning this method, we employed two types of training problem instances: 1) the 14 CHR instances and 2) 14 *large instances* from the SOAK and RAND sets with $n = 150$ to $n = 200$. All the IG variants in this experiment apply C1 in the construction phase (we will study the application of C2 in the next experiment). The IG methods were stopped using a time limit of 360 seconds and one run was performed for each problem instance.

In each experiment, we compute for each instance the overall best solution value, *BestValue*, obtained by the execution of all methods under consideration. Then, for each method, we compute the relative deviation between the best solution value found by the method and the *BestValue*. In Table 1, we report the average of this relative deviation (*Dev*) across all the instances considered in each particular experiment and the number of instances (*#Best*) for which the value of the best solution obtained by a given method matches *BestValue*. We also show the average *rankings* (computed by the Friedman test) obtained by these IG variants (Section 5.1).

In order to analyze the results, we have applied Iman-Davenport’s test (the level of significance considered was 0.05). We have observed the existence of significant differences among the rankings (the statistical values, 79.41 and 160.16, are greater than the critical ones, 1.56 and 1.55, respectively). Then, we have compared the best ranked algorithm for each training set (control algorithm), [RW, AL, $0.4n$] and [RW, AL, $0.6n$], with the other IG versions, by means of Holm’s test (a post hoc statistical analysis) with $p = 0.05$. The last column in Table 1 indicates whether Holm’s test finds statistical differences between the control algorithm and the corresponding algorithm.

We can draw the following conclusions from Table 1:

- The acceptance criterion has the largest impact on the IG performance. Unexpectedly, the best outcomes (rankings) are obtained when the RW acceptance criterion is used (Holm’s test confirms that its superiority is statistically significant). As a matter of fact, it is unusual to find IG models in the literature employing this strategy (most of them show a bias towards maintaining high quality solutions). Thus, we may remark that the combination of the diversification provided by RW and the intensification of the improvement procedure benefits the IG performance.

IG					LARGE Instances				
(AC, LS, n_d)	<i>Av. Ran.</i>	<i>Dev</i>	<i>#Best</i>	Holm	(AC, LS, n_d)	<i>Av. Ran.</i>	<i>Dev</i>	<i>#Best</i>	Holm
RB FL 0.2n	21.5	6.1971	0	yes	RB BL 0.8n	21.9	0.0530	0	yes
RB AL 0.4n	21.3929	6.2129	0	yes	RB FL 0.8n	21.6667	0.0586	0	yes
RB BL 0.2n	21.2857	6.1791	0	yes	RB AL 0.8n	21.6667	0.0526	0	yes
RB FL 0.4n	20.7143	6.1676	0	yes	RB BL 0.6n	21.1667	0.0442	0	yes
RB BL 0.4n	20.2143	6.1697	0	yes	RB AL 0.6n	20.6	0.0413	0	yes
RB AL 0.2n	20.0714	6.0514	0	yes	RB FL 0.6n	19.8	0.0468	0	yes
RB AL 0.6n	18.6071	5.6991	0	yes	RB BL 0.4n	17.3333	0.0247	0	yes
RB FL 0.6n	17.8571	5.5717	0	yes	RB BL 0.2n	16.5333	0.0224	0	yes
RB BL 0.6n	17.7143	5.6132	0	yes	RB FL 0.4n	15.9333	0.0222	0	yes
RB AL 0.8n	14.5714	4.7705	0	yes	RB AL 0.4n	15.7	0.0214	0	yes
RB BL 0.8n	14.1429	4.7515	0	yes	RB AL 0.2n	15.0333	0.0198	0	yes
RB FL 0.8n	13.9286	4.6530	0	yes	RB FL 0.2n	14.6667	0.0205	0	yes
RW FL 0.8n	8.7857	0.1185	1	no	RW BL 0.2n	10.5333	0.0088	0	yes
RW FL 0.6n	8.6429	0.0996	0	no	RW BL 0.8n	10.1333	0.0081	0	yes
RW BL 0.6n	8.2857	0.0996	0	no	RW AL 0.2n	9.4	0.0078	0	yes
RW BL 0.8n	7.9286	0.1136	1	no	RW FL 0.2n	9	0.0073	0	yes
RW AL 0.8n	7.3214	0.0959	1	no	RW FL 0.8n	7.5333	0.0065	0	no
RW BL 0.2n	6.2857	0.0775	2	no	RW AL 0.8n	7.3333	0.0062	0	no
RW AL 0.6n	6.1429	0.0764	2	no	RW BL 0.4n	7.2	0.0055	0	no
RW FL 0.2n	5.8571	0.0626	4	no	RW AL 0.4n	4.7333	0.0038	1	no
RW AL 0.2n	5.5357	0.0633	3	no	RW FL 0.4n	4.2333	0.0031	0	no
RW BL 0.4n	5.0714	0.0762	2	no	RW BL 0.6n	4	0.0027	1	no
RW FL 0.4n	4.8929	0.0565	3	no	RW FL 0.6n	2.1333	0.0010	4	no
RW AL 0.4n	3.25	0.0412	4	no	RW AL 0.6n	1.7667	0.0005	9	no

Table 1: Results of the IG instances

- The choice of the value for n_d has an important influence, as well, on the IG behavior. For both instance sets, the three best ranked algorithms employed the same setting for this parameter ($0.4n$ for the CHR set and $0.6n$ for the large instances).
- Although the results of Holm’s test indicate that there are no significant differences in the performance of the improvement procedures (and then, there exists little sensitivity to changes in this IG component), we choose the ABC local search for all remaining experiments involving IG since the best ranked configurations for the two instance sets are based on this method.

We now undertake to analyze the effects of the greedy constructive algorithms (C1 and C2 described in Section 4.1) on the IG performance. We have implemented IGs with C1 and C2, and the two best values of the n_d parameter identified in the previous experiment ($n_d = 0.4n$ and $0.6n$). They apply the RW acceptance criterion and the ABC local search. The Iman and Davenport’s test indicates the existence of significant differences among the rankings (the statistical values, 58.17 and 164.97 are larger than the critical ones, 2.84 and 2.82, respectively).

IG					LARGE Instances				
(Greedy, n_d)	<i>Av. Ran.</i>	<i>Dev</i>	<i>#Best</i>	Holm	(Greedy, n_d)	<i>Av. Ran.</i>	<i>Dev</i>	<i>#Best</i>	Holm
C2 0.6n	3.6071	0.7424	0	yes	C2 0.4n	3.9333	0.0102	0	yes
C2 0.4n	3.3929	0.6595	0	yes	C2 0.6n	3	0.0084	0	yes
C1 0.6n	1.6786	0.0440	5	no	C1 0.4n	2	0.0034	1	yes
C1 0.4n	1.3214	0.0098	10		C1 0.6n	1.0667	4.66E-05	14	

Table 2: Results of IG versions with different greedy constructive algorithms

With regard to this result, we compare the best ranked IG for each instance set, $[C1, 0.4n]$ and $[C1, 0.6n]$, with the other IG variants, by means of Holm’s test. Table 2 reports its results. Our main observation about the data in this table is that Holm’s test revealed the clear advantage of C1 on C2. It is interesting to remark that, quite surprisingly, the simplest constructive algorithm, when iterated inside IG, becomes the most effective one.

5.3 Study of the Memory Based Methods: SO and TLS

In this section, firstly, we present a series of preliminary experiments that were conducted to set the values of the key search parameters of TLS (Section 4.2), *TabuTenure* and *MaxIter*. In par-

ticular, we have built different SO instances (RW, C1, and $n_{tabu} = 0.05$) with $n_d = \{0.4n, 0.6n\}$, $TabuTenure = \{2, 10, 50\}$, and $MaxIter = \{100, 200, 500\}$. Table 3 summarizes the results of these algorithms and the conclusions of Holm’s test (the Iman-Davenport test finds significant performance differences between the considered algorithms because its statistical values, 14.03 and 27.56, are greater than the critical ones, 1.669 and 1.665, respectively).

SO		CHR Instances				SO		LARGE Instances			
(n_d, TT, MI)	<i>Av. Ran.</i>	<i>Dev</i>	<i>#Best</i>	Holm	(n_d, TT, MI)	<i>Av. Ran.</i>	<i>Dev</i>	<i>#Best</i>	Holm		
0.6n 50 500	16.75	0.1947	0	yes	0.6n 50 500	15.2	0.0072	0	yes		
0.4n 50 200	13.3929	0.1096	0	yes	0.6n 2 500	14.9333	0.0064	0	yes		
0.6n 10 500	13	0.1204	1	yes	0.6n 10 500	14.6667	0.0067	0	yes		
0.4n 50 500	12.6786	0.1259	1	yes	0.6n 2 200	13.3333	0.0062	0	yes		
0.6n 50 100	12.6071	0.1156	2	yes	0.6n 10 200	13.0333	0.0059	0	yes		
0.6n 50 200	12.1786	0.1298	3	yes	0.6n 2 100	12.7333	0.0060	0	yes		
0.6n 10 200	12.1429	0.1065	1	yes	0.6n 50 100	12.5	0.0055	0	yes		
0.4n 10 500	11.3571	0.1016	2	yes	0.6n 50 200	12.1667	0.0055	0	yes		
0.6n 10 100	10.8571	0.0961	2	yes	0.6n 10 100	11.8333	0.0053	0	yes		
0.4n 10 200	9.8214	0.0869	3	yes	0.4n 10 500	9.0333	0.0034	1	yes		
0.4n 50 100	9.7857	0.0790	2	yes	0.4n 50 500	8.3	0.0034	1	yes		
0.6n 2 500	6.1786	0.0324	5	no	0.4n 2 500	7.6	0.0029	0	no		
0.4n 10 100	5.7143	0.0350	6	no	0.4n 2 200	5.2667	0.0020	0	no		
0.6n 2 200	5.6429	0.0425	3	no	0.4n 50 200	5.1667	0.0018	3	no		
0.6n 2 100	5.1429	0.0364	3	no	0.4n 10 200	4.8	0.0019	2	no		
0.4n 2 200	4.8929	0.0327	4	no	0.4n 2 100	4.0667	0.0014	4	no		
0.4n 2 100	4.8214	0.0342	4	no	0.4n 10 100	3.3333	0.0012	3	no		
0.4n 2 500	4.0357	0.0164	5		0.4n 50 100	3.0333	0.0011	6			

Table 3: Results of SO with TLS when applying different n_d , $TabuTenure$, and $MaxIter$ values

The results in Table 3 show that the best outcomes for the CHR instances are obtained with the lowest $TabuTenure$ value, and the ones for the large instances with the lowest $MaxIter$ values. Interestingly, the $TabuTenure$ parameter had less impact on the performance of TLS when dealing with large instances, and the same happens with the $MaxIter$ parameter in the case of CHR. Nevertheless, given the relatively robust performance achieved by using $n_d = 0.4n$, $TabuTenure=10$, and $MaxIter = 100$ (second best ranked algorithm for the large instance set and Holm’s test did not detect significant differences between this configuration and best one for CHR), we choose this setting for all remaining experiments involving this local search operator.

Next, we investigate the effects of varying the n_{tabu} parameter associated with SO (number of removed elements that cannot be reinserted into the solution). In particular, we have generated 5 SO configurations with $n_{tabu} = \{0.05n_d, 0.1n_d, 0.25n_d, 0.5n_d, n_d\}$. In order to study the results, we have applied Iman-Davenport’s test (the level of significance considered was 0.05). We have observed the existence of significant differences among the rankings (the statistical values, 7.85 and 277.66, are greater than the critical ones, 2.54 and 2.53, respectively). Then, we analyze the performance of these SO instances by means of Holm’s test (Table 4).

SO		CHR Instances				SO		LARGE Instances			
(n_{tabu})	<i>Av. Ran.</i>	<i>Dev</i>	<i>#Best</i>	Holm	(n_{tabu})	<i>Av. Ran.</i>	<i>Dev</i>	<i>#Best</i>	Holm		
n_d	4.2857	0.10776	yes	2	n_d	5	0.02139	0	yes		
0.5 n_d	3.4286	0.06932	yes	4	0.5 n_d	4	0.01429	0	yes		
0.1 n_d	2.9286	0.05291	no	3	0.25 n_d	3	0.00697	0	yes		
0.25 n_d	2.7143	0.03983	no	3	0.1 n_d	1.6	0.00129	6	no		
0.05 n_d	1.6429	0.00514		12	0.05 n_d	1.4	0.00069	9			

Table 4: Results of SO with different n_{tabu} values

The results in Table 4 strongly reveal a clear superiority of the SO version with $n_{tabu} = 0.05n_d$ for both instance sets. Clearly, too much diversity (high n_{tabu} values) is not suitable to allow SO to reach fitter search areas. To sum up, the effects derived from the combination of the exploration power of TLS and a moderate diversification by SO (low n_{tabu} values) are sufficient to attain a high level of robustness for both instance sets.

5.4 Comparison Between IG, SO, and ITS

The first objective of this section is to compare the results for IG, SO, and Iterated Tabu Search ITS [19]. ITS is another memory-based approach proposed to deal with the QMSTP that has proved to be one of the most appealing contemporary metaheuristic approaches for this problem and other combinatorial optimization problems with quadratic objective functions, including the unconstrained binary quadratic optimization problems [15], the maximum diversity problem [16], and the Max-2-SAT problem [17, 18]. We should point out that IG, SO and ITS were run under the same computational conditions (machine, programming language, compiler, and time limits; see Section 5.1) in order to enable a fair comparison between them. We have used the source code of ITS provided by the author². Henceforth, all studied algorithms were run 10 times on each problem instance.

In Table 5, we have summarized the results of the IG and SO versions that achieved the best outcomes in Sections 5.2 and 5.3, respectively, and the ones of ITS (best performance measure values are outlined in boldface). For each algorithm, in column *Avg-Dev*, we include the average of the relative deviations from *BestValue* (Section 5.2) of the solution values found by the algorithm in the 10 runs on a set of problem instances and, in the case of the column *%best*, we outline the percentage of runs in which the algorithm reached *BestValue* for the instances in the corresponding group.

Inst. Set	IG		SO		ITS		HSII	
	<i>Avg-Dev</i>	<i>%best</i>	<i>Avg-Dev</i>	<i>%best</i>	<i>Avg-Dev</i>	<i>%best</i>	<i>Avg-Dev</i>	<i>%best</i>
NUG	0.0745	0	0.0429	7.33	0.0104	15.33	0.0176	8.67
CHR	0.2434	10	0.0945	23.57	0.9918	10	0.0665	20
RAND	0.0016	8.89	0.0048	0	0.0046	0	0.0021	1.11
SOAK	0.001	6.67	0.0072	0	0.0016	3.33	0.0012	2.22
CP	0.0005	71.94	0.004	25.56	0.0004	77.50	0.0006	72.22
<i>Avg.</i>	0.0642	19.5	0.03068	11.292	0.20176	21.232	0.0176	20.844

Table 5: Comparison between IG, SO, ITS, and HSII

The results of IG, SO, and ITS (Table 5) allow us to make the following observations.

- The high *Avg-Dev* values for IG, SO, and, specially, for ITS on the CHR set suggest that it includes the hardest QMSTP instances. Note that, in this case, the results of SO are much better than those of its competitors. The high levels of diversity promoted by the memory-based techniques in SO became decisive to guarantee an effective search when dealing with these complex problem instances.
- For the large instance sets, RAND and SOAK, the *Avg-Dev* and *%best* measures for IG show better quality than the ones for SO and ITS. We should note that, in contrast with the previous case, SO obtained the lowest quality results. For problems with many vertices, the action of the memory-based techniques in SO may produce detrimental disruptions in the solutions, causing misbehavior in the SO search. However, the diversification originated from the random deletion of elements during the destruction phase in IG may be enough to favor the progress towards promising search zones.
- ITS outperforms IG and SO on the CP and NUG instance sets.

Hence, it seems that, somewhat unsurprisingly, there is no single best algorithm; SO is superior for the most complex instance set, IG for the large instances, and ITS becomes the winner for the case of the easiest instances. Thus, regarding these results and with the aim of producing a robust operation, we have built a hybrid algorithm, dubbed HSII, that combines SO, IG, and ITS. Specifically, HSII is a *relay collaborative* hybrid metaheuristic [24] that executes SO, IG, and ITS in a pipeline fashion. First, SO is performed during the $P_{SO}\%$ of the

²ITS is publicly available at <http://www.soften.ktu.lt/~gintaras/qmstp.html>

imposed time limit and the best found solution becomes the initial solution for IG, which is run during $P_{IG}\%$ of this time. Finally, the output of IG is supplied as input to ITS. We have set P_{SO} and P_{IG} to 25% and 50%, respectively. The idea of this hybridization scheme is: (1) to use SO as *diversification* agent to reach good initial solutions for the most difficult problem instances, then (2) to allow IG enough time to offer a suitable behavior on large instances, and finally (3) to employ ITS as an effective *improvement* procedure for refining the best solution found in the previous stages.

Table 5 has the *Avg-Dev* and *%best* values for HSII. In general, for most problem sets, this algorithm might obtain *Avg-Dev* values very similar to the ones returned by the fittest algorithms (or even better, as was the case for CHR). Thus, we may conclude that the combination of the proposed IG and SO approaches along with ITS (by following a simple hybridization scheme) resulted really advisable to achieve an acceptable level of robustness across a wide range of different QMSTP instances.

5.5 HSII vs. State-of-the-art Metaheuristics for the QMSTP

In this section, we undertake a comparative analysis among HSII and the current best algorithms for the QMSTP, ITS [19], ABC [23], and local search algorithm with tabu thresholding (LS-TT) [14]. We should point out that HSII, ITS, and ABC were run under the same computational conditions (Section 5.1). We have implemented ABC in C. Since we could not obtain the source code of LS-TT, we have used the results reported in [14] for the NUG and CHR instances to compare with our algorithm (they were obtained with a single run). The parameter values used for each considered algorithm are the ones recommended in the original works. Their results are outlined in Tables 9-13 in Appendix A. With the aim of determining the position of SO and IG with regards to the state-of-the-art, we have included them in this comparative study as well.

In order to detect the differences among HSII and the other algorithms, we have applied Wilcoxon’s test. Tables 6-8 have the *%best* and *Avg-Dev* measures and summarize the results of this procedure for $p = 0.05$, where the values of $R+$ (associated to HSII) and $R-$ of the test are specified. The last column indicates whether Wilcoxon’s test found statistical differences between these algorithms. If $\min\{R^+, R^-\}$ is less than or equal to the critical value, this test detects significant differences between the algorithms, which means that an algorithm outperforms its opponent. Particularly, if this occurs and $R^- = \min\{R^+, R^-\}$, then HSII is statistically better than the other algorithm. All large instances (RAND and SOAK) and QAP based instances (NUG and CHR) have been grouped in order to apply the statistical test to a significant number of instances.

Algorithm	<i>Avg-Dev</i>	<i>%best</i>	$R+$	$R-$	Diff?
IG	0.0005	71.94	300.0	366.0	no
SO	0.0040	25.56	628.5	1.5	yes
ITS	0.0004	77.50	174.5	455.5	yes
ABC	0.0061	20.28	661.0	5.0	yes
HSII	0.0006	72.22			

Table 6: Comparison on the CP set (Wilcoxon’s test; critical value = 208)

Algorithm	NUG set		CHR set		$R+$	$R-$	Diff?
	<i>Avg-Dev</i>	<i>%best</i>	<i>Avg-Dev</i>	<i>%best</i>			
IG	0.0745	0	0.2459	10.00	434.0	1.0	yes
SO	0.0429	7.33	0.0965	22.86	294.5	140.5	no
ITS	0.0104	15.33	0.9965	10.00	196.0	210.0	no
ABC	0.2597	0.00	1.9261	0.00	435.0	0.0	yes
LS-TT	0.0671	0.00	0.2918	7.14	411.0	24.0	yes
HSII	0.0176	8.67	0.0686	20.00			

Table 7: Comparison on the NUG and CHR sets (Wilcoxon’s test; critical value = 126)

Algorithm	RAND set		SOAK set		R+	R-	Diff?
	<i>Avg-Dev</i>	<i>%best</i>	<i>Avg-Dev</i>	<i>%best</i>			
IG	0.0016	8.89	0.0010	6.67	23.0	148.0	yes
SO	0.0048	0	0.0072	0	171.0	0.0	yes
ITS	0.0046	0	0.0016	3.33	162.0	9.0	yes
ABC	0.0111	0.00	0.0077	0.00	171.0	0.0	yes
HSII	0.0021	1.11	0.0012	2.22			

Table 8: Comparison on the RAND and SOAK sets (Wilcoxon’s test; critical value = 40)

The results presented in Tables 6-8 reveal the following.

- For large instances (Table 8), HSII has the upper hand in the statistical comparison over its competitors (ABC and ITS).
- For complex instances (Table 7), our hybrid metaheuristic clearly obtained statistically significant improvements relative to ABC and LS-TT. In addition, analyzing the computational time reported in [14] (where a Pentium IV PC at 3 GHz was used to carry out experiments) for LS-TT on these instances (Tables 10 and 11), we may conclude that, in general, LS-TT required more time than our hybrid algorithm (with a time limit of 1000 seconds). Therefore, our proposal outperforms this algorithm in all aspects, when considering the results on the NUG and CHR instance sets.

On the other hand, Wilcoxon’s test did not detect significant differences between HSII and ITS on these instances. According to the *Avg-Dev* and *%best* measures, our algorithm clearly beats ITS for the CHR instances. For the case of the NUG instances, they show similar values for the *Avg-Dev* measure, which explains that Wilcoxon’s test did not find statistical differences between them when the two sets of instances were considered.

- HSII was found to be superior to ABC for the CP instances (Table 6), however, only in this case, ITS could statistically outperform our proposal (even so, note that HSII attained a similar *%best* value). It is worth mentioning that, in [19], iterated tabu search proved superior to other advanced methods precisely on this instance set.

In summary, this experimental analysis confirms that our hybrid HSII approach is a very attractive alternative to the existing approaches for the QMSTP.

Finally, we should note that SO achieved better *Avg-Dev* and *%best* values than ABC and LS-TT in the case of the NUG and CHR sets (Table 7), and IG was able to outperform all other algorithms on the RAND and SOAK sets (Table 8). These facts evidence the great potential of the proposed SO and IG metaheuristics to effectively handle large and highly complex problem instances, which posed real challenges for previous metaheuristic approaches in the literature.

6 Conclusions

Our research study demonstrates the effectiveness of a strategy for solving the QMSTP that alternates between constructive and destructive phases, as originally proposed in strategic oscillation (SO) and more recently in the iterated greedy (IG) method. We limit consideration of SO in this study to a one-sided oscillation that does not cross the feasibility boundary in order to make it more comparable to the IG approach, and differentiate it from IG primarily by introducing tabu search (TS) memory as in previous SO algorithms.

Our tests disclose that the memory-based SO method is able to solve complex QMSTP instances better than other previous algorithms. On the other hand, our implementation of the IG algorithm succeeds in performing most effectively for large problems that are not highly complex, while the iterated tabu search (ITS) algorithm performs best in application to easier instances. Based on these findings we additionally developed a hybrid method HSII

that combines these three algorithms, which proved very effective across the board with the exception of one class of problem instances, where iterated tabu search remained the winner.

Overall, the ability of the alternating constructive/destructive strategies to give superior outcomes for larger problems, with memory proving valuable for complex problems and a disregard for memory proving valuable for easier problems, invites further consideration of other forms of strategic oscillation, particularly by crossing feasibility boundaries (whose value is underscored in [9]) and by going beyond the reliance on randomization in carrying out destructive moves by introducing more advanced strategies for selecting these moves (as indicated in [7]).

Acknowledgments

We are thankful to Dr. Temel Öncan for providing the NUG and CHR instances used in [14]. This work was supported by the Research Projects TIN2011-24124, TIN2009-07516, TIN2012-35632, and P08-TIC-4173.

A Results of the Algorithms

Tables 9-13 outline the results of the algorithms concerning the experiment in Section 5.5. The columns with heading ‘Ave’ (respectively, ‘Min’) present the difference between the average of the objective function value of the solutions reached by an algorithm after 10 runs (respectively, the objective function value of the best solution out of these 10 solutions) and the value in the second column. Additionally, Tables 10 and 11 have the computational times required by LS-TT to achieve their results (both of them were directly extracted from [14]).

Instance	Best Known V.	HSII		ITS		ABC	
		Ave	Min	Ave	Min	Ave	Min
n40_m257.1	5945	0	0	0	0	13.8	0
n40_m257.2	56237	0	0	0	0	0	0
n40_m257.3	6925	0	0	0	0	0	0
n40_m257.4	57874	0	0	0	0	0	0
n40_m522.1	5567	3.2	0	0.6	0	24.8	18
n40_m522.2	51851	0	0	24.6	0	392.2	220
n40_m522.3	6456	0	0	0	0	18.9	18
n40_m522.4	53592	9.9	0	13	0	104.5	43
n40_m780.1	5368	0.5	0	0.9	0	68.1	13
n40_m780.2	49817	18.2	0	6.2	0	591.4	129
n40_m780.3	6208	0	0	1	0	8	0
n40_m780.4	51229	77.1	0	0	0	647.6	0
n45_m326.1	7521	0	0	0	0	1.4	0
n45_m326.2	70603	0	0	0	0	42.9	0
n45_m326.3	8720	0	0	0	0	4.7	0
n45_m326.4	72676	0	0	0	0	82	82
n45_m663.1	7161	16.8	0	15.7	0	51.9	44
n45_m663.2	66889	23.8	0	0	0	287.6	0
n45_m663.3	8225	0.6	0	0	0	36.3	25
n45_m663.4	68737	46	0	0	0	440.2	0
n45_m990.1	6944	7.3	1	4.7	0	76.9	47
n45_m990.2	64840	45.6	0	50.9	0	1048.3	583
n45_m990.3	7827	0	0	0	0	4	0
n45_m990.4	66508	148	0	141.9	0	1042.2	331
n50_m404.1	9393	0	0	0	0	34.1	24
n50_m404.2	88942	0	0	0	0	370.4	349
n50_m404.3	10717	0	0	0	0	0	0
n50_m404.4	91009	0	0	0	0	165	165
n50_m820.1	8958	10.5	0	1.3	0	76.8	33
n50_m820.2	84020	100.4	0	4.4	0	1256.6	601
n50_m820.3	10100	0	0	0	0	46.7	37
n50_m820.4	86231	147.6	0	18.7	0	1169.9	599
n50_m1225.1	8713	25.1	0	21	0	128.7	53
n50_m1225.2	81858	205.5	27	192.5	27	1128.1	450
n50_m1225.3	9836	8.4	0	1.9	0	47.8	20
n50_m1225.4	83838	31.6	0	102.1	0	883.6	43

Table 9: Results for CP instance set

Instance	Opt. Value	HSII		ITS		ABC		LS-TT	
		Ave	Min	Ave	Min	Ave	Min	Ave/Min	Time
nug12	578	0	0	0	0	165.8	78	27	639
nug14	1014	20.8	12	9.2	0	273.8	126	70	724
nug15	1150	16.8	2	10.2	0	334.6	254	115	1348
nug16a	1610	39.4	24	36.6	28	432	334	132	2311
nug16b	1240	33.2	6	28.4	8	369.8	240	110	2936
nug17	1732	58.6	42	49.2	36	475.8	334	142	3422
nug18	1930	74.8	54	61	34	488.6	294	126	3482
nug20	2570	127	92	100	74	645.4	330	290	5151
nug21	2438	137.2	102	106.8	64	870.6	604	260	5184
nug22	3596	190.2	154	161	116	1397.2	984	272	5482
nug24	3488	237.6	200	187	160	1133	852	386	5914
nug25	3744	252.6	196	224.4	210	1116.4	778	339	5983
nug27	5234	342.6	300	303.4	222	1481	1050	732	6025
nug28	5166	353.2	318	321.4	240	1300.2	1072	653	6087
nug30	6124	482	404	429.6	382	1775.8	1564	799	6227

Table 10: Results for NUG instance set

Instance	Opt. Value	HSII		ITS		ABC		LS-TT	
		Ave	Min	Ave	Min	Ave	Min	Ave/Min	Time
chr12a	9552	0	0	14625.6	7142	20331.4	4738	1618	783
chr12b	9742	92	0	16221.2	6614	22457	11810	1011	790
chr12c	11156	3	0	12243.8	6278	14143.2	4654	1556	783
chr15a	9896	446.6	56	16624.8	6822	27719.4	14328	1742	1239
chr15b	7990	1628	394	21137.8	9218	33899.8	20350	2155	1136
chr15c	9504	1035.6	0	19956.2	9798	25663.2	16062	3265	1254
chr18a	11098	3439.2	2736	22572.4	11398	31298.6	13856	1659	3325
chr18b	1534	3.4	0	0	0	1123	626	142	3354
chr20a	2192	225.8	84	162.4	40	4142.4	2550	253	4968
chr20b	2298	223.8	164	184.4	142	3435	1406	432	4652
chr20c	14142	7976	6064	31867.6	22416	50674.6	35700	15982	4763
chr22a	6156	312.6	178	269	234	3759.8	2532	2604	5089
chr22b	6194	336.6	202	249.4	120	4067.6	2714	2208	4741
chr25a	3796	929.6	514	787	504	7728.2	4744	5862	5223

Table 11: Results for CHR instance set

Instance	Best Known V.	HSII		ITS		ABC	
		Ave	Min	Ave	Min	Ave	Min
RAND-150-1	192606	304.1	0	638.5	340	2223.1	1688
RAND-150-2	192607	315.8	0	762.9	427	2187	1611
RAND-150-3	192577	215.6	0	726.1	388	2112.8	1305
RAND-200-1	350517	506.6	0	1270.2	699	3874.3	2646
RAND-200-2	350389	513.4	0	1434.7	923	3756.3	3395
RAND-200-3	351057	228.4	0	883.8	409	2856.4	2112
RAND-250-1	556929	505.6	0	2306.5	1522	5585.8	4935
RAND-250-2	557474	376.1	0	2004.2	1346	4898	3230
RAND-250-3	556813	650.4	0	2676.8	2491	5423.3	4684

Table 12: Results for RAND instance set

Instance	Best Known V.	HSII		ITS		ABC	
		Ave	Min	Ave	Min	Ave	Min
SOAK-150-1	206721	368.4	204	283.9	0	1557.2	931
SOAK-150-2	206761	519.5	341	392.6	0	2024.1	1445
SOAK-150-3	206781	173.2	0	178.6	21	1336.4	752
SOAK-200-1	370137	393.5	128	396.3	0	3196.8	2282
SOAK-200-2	369982	201.6	0	369	46	2311	1659
SOAK-200-3	370045	300.8	0	345.7	1	2860	2072
SOAK-250-1	581819	464.7	0	1250.4	463	4260.3	2980
SOAK-250-2	581691	322.3	0	1181.9	454	4132.7	2718
SOAK-250-3	581854	736.8	0	1671.8	854	4491.1	3863

Table 13: Results for SOAK instance set

References

- [1] Assad A, Xu W. The quadratic minimum spanning tree problem. *Naval Research Logistics* 1992; 39: 399–417.
- [2] Cordone R, Passeri G. Heuristic and exact approaches to the quadratic minimum spanning tree problem. In: *Seventh Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW08)*, Gargnano, Italy, Universit degli Studi di Milano, 2008, pp. 52–55.
- [3] Christofides N, Benavent E. An exact algorithm for the quadratic assignment problem. *Operations Research* 1989; 37(5):760–8.
- [4] Fanjul-Peyroa L, Ruiz R. Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research* 2010; 207: 55–69.
- [5] Gao J, Lu M. Fuzzy quadratic minimum spanning tree problem. *Applied Mathematics and Computation* 2005; 164: 773–788.
- [6] García S, Molina D, Lozano M, Herrera F. A study on the use of nonparametric tests for analyzing the evolutionary algorithms’ behaviour: a case study on the CEC’2005 special session on real parameter optimization. *Journal of Heuristics* 2009; 15: 617–644.
- [7] Glover F, Laguna M. *Tabu Search*. Dordrecht: Kluwer, 1997.
- [8] Glover F. Heuristics for Integer Programming Using Surrogate Constraints. *Decision Sciences* 1977; 8: 156–166.
- [9] Glover, F., Hao, J.K. The Case for Strategic Oscillation. *Annals of Operations Research* 2011; 183 (1): 163–173.
- [10] Jacobs LW, Brusco MJ. A local-search heuristic for large set-covering problems. *Naval Research Logistics* 1995; 42: 1129–1140.
- [11] Kruskal JB. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society* 1956; 7: 48–50.
- [12] Lozano M, Molina D, García-Martínez C. Iterated greedy for the maximum diversity problem. *European Journal of Operational Research* 2011; 214: 31–38.
- [13] Nugent CE, Vollman TE, Ruml J. An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research* 1968; 16: 150–73.
- [14] Öncan T, Punnen AP. The quadratic minimum spanning tree problem: a lower bounding procedure and an efficient search algorithm. *Computers and Operations Research* 2010; 37: 1762–1773.
- [15] Palubeckis G. Iterated tabu search for the unconstrained binary quadratic optimization problem. *Informatica* 2006; 17: 279–296.
- [16] Palubeckis G. Iterated tabu search for the maximum diversity problem. *Applied Mathematics and Computation* 2007; 189: 371–383.
- [17] Palubeckis G. Solving the weighted Max-2-SAT problem with iterated tabu search. *Information Technology and Control* 2008; 37: 275–284.
- [18] Palubeckis G. A new bounding procedure and an improved exact algorithm for the Max-2-SAT problem. *Applied Mathematics and Computation* 2009; 215: 1106–1117.
- [19] Palubeckis G, Rubliauskas D, Targamadz A. Metaheuristic approaches for the quadratic minimum spanning tree problem. *Information Technology and Control* 2010; 39(4): 257–268.

- [20] Ruiz R, Stützle T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research* 2007; 177: 2033–2049.
- [21] Ruiz R, Stützle T. An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research* 2008; 187: 1143–1159.
- [22] Soak SM, Corne DW, Ahn BH. The edge-window-decoder representation for tree-based problems. *IEEE Transactions on Evolutionary Computation* 2006; 10: 124–144.
- [23] Sundar S, Singh A. A swarm intelligence approach to the quadratic minimum spanning tree problem. *Information Sciences* 2010; 180: 3182–3191.
- [24] Talbi E-G. A taxonomy of hybrid metaheuristics. *J Heuristics* 2002; 8(5): 541–65.
- [25] Xu W. On the quadratic minimum spanning tree problem. In: Gen M, Xu W, editors. *Proceedings of 1995 Japan–China International Workshops on Information Systems*, Ashikaga, Japan, 1995, pp. 141–148.
- [26] Ying KC. Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic. *International Journal of Advanced Manufacturing Technology* 2008; 38(3–4): 348–354.
- [27] Ying KC, Cheng HM. Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. *Expert Systems with Applications* 2010; 37(4): 2848–2852.
- [28] Zhou G, Gen M. An effective genetic algorithm approach to the quadratic minimum spanning tree problem. *Computers and Operations Research* 1998; 25: 229–237.

Bibliography

- [ABR03] Aiex R., Binato S., y Resende M. (2003) Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing* 29(4): 393–430.
- [Adl93] Adler D. (1993) Genetic algorithm and simulated annealing: a marriage proposal. In *Proc. of the IEEE international conference on neural network*, pp. 1104–1109.
- [AGA99] Allahverdi A., Gupta J., y Aldowaisan T. (1999) A review of scheduling research involving setup considerations. *Omega* 27(2): 219–239.
- [AK89] Aarts E. y Korst J. (1989) *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons.
- [AK99] Azizoglu M. y Kirca O. (1999) On the minimization of total weighted flow time with identical and uniform parallel machines. *European Journal of Operational Research* 113(1): 91 – 100.
- [AK02] Aarts E. y Korst J. (2002) Selected topics in simulated annealing. In *Essays and Surveys in Metaheuristics*, pp. 1–37. Kluwer Academic Publishers Group.
- [AL97] Aarts E. y Lenstra J. K. (Eds.) (1997) *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA.
- [Ant09] Antonio C. (2009) A study on synergy of multiple crossover operators in a hierarchical genetic algorithm applied to structural optimisation. *Structural and Multidisciplinary Optimization* 38(2): 117–135.
- [ARPT05] Aiex R., Resende M., Pardalos P., y Toraldo G. (2005) GRASP with path relinking for three-index assignment. *Inform Journal on Computing* 17(2): 224–247.
- [AX92] Assad A. y Xu W. (1992) The quadratic minimum spanning tree problem. *Naval Research Logistics* 39(3): 399–417.
- [AY05] Aydin M. y Yigit V. (2005) *Parallel Simulated Annealing*, pp. 267–288. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley.
- [BARS08] Blum C., Aguilera M. J. B., Roli A., y Sampels M. (Eds.) (2008) *Hybrid Metaheuristics: An Emerging Approach to Optimization*. Springer.
- [BB96] Brassard G. y Bratley P. (1996) *Fundamentals of algorithmics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

- [BBB12] Bouamama S., Blum C., y Boukerram A. (2012) A population-based iterated greedy algorithm for the minimum weight vertex cover problem. *Applied Soft Computing* 12(6): 1632 – 1639.
- [Bea98] Beasley J. (1998) Heuristic algorithms for the unconstrained binary quadratic programming problem. Technical report, The Management School, Imperial College.
- [BEP⁺07] Blazewicz J., Ecker K., Pesch E., Schmidt G., y Weglarz J. (2007) *Handbook on Scheduling: Models and Methods for Advanced Planning (International Handbooks on Information Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- [BFM97] Bäck T., Fogel D., y Michalewicz Z. (1997) *Handbook of Evolutionary Computation*. Institute of Physics Publishers.
- [BH00] Brucker P. y Hurink J. (2000) Solving a chemical batch scheduling problem by local search. *Annals of Operations Research* 96(1): 17–38.
- [BHS89] Brown D., Huntley C., y Spillane A. (1989) A parallel genetic heuristic for the quadratic assignment problem. In *Proceedings of the third international conference on Genetic algorithms*, pp. 406–415. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [Blu10] Blum C. (2010) Hybrid Metaheuristics – Guest Editorial. *Computers & Operations Research* 37(3): 430–431.
- [BM73] Baker K. y Merten A. (1973) Scheduling with parallel machines and linear delay costs. *Naval Research Logistics Quarterly* 20: 793–804.
- [BP94] Belouadah H. y Potts C. (1994) Scheduling identical parallel machines to minimize total weighted completion time. *Discrete Applied Mathematics* 48(3): 201–218.
- [BPRR11] Blum C., Puchinger J., Raidl G., y Roli A. (2011) Hybrid metaheuristics in combinatorial optimization: A survey. *App. Soft Comput.* 11: 4135–4151.
- [BR03] Blum C. y Roli A. (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.* 35(3): 268–308.
- [BRA05] Blum C., Roli A., y Alba E. (2005) *An Introduction to Metaheuristic Techniques*, pp. 1–42. John Wiley & Sons, Inc.
- [BS02] Beyer H. y Schwefel H. (2002) Evolution strategies—a comprehensive introduction. *Nat. Comput.* 1(1): 3–52.
- [BSMD08] Bandyopadhyay S., Saha S., Maulik U., y Deb K. (2008) A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Trans. Evol. Comput.* 12(3): 269–283.
- [Bux89] Buxey G. (1989) Production scheduling: Practice and theory. *European Journal of Operational Research* 39: 17–31.
- [CF04] Cotta C. y Fernandez A. (2004) A hybrid GRASP - Evolutionary algorithm approach to Golomb ruler search. In Yao X., Burke E., Lozano J., Smith J., MereloGuervos J., Bullinaria J., Rowe J., Tino P., Kaban A., y Schwefel H. (Eds.) *Parallel Problem Solving from Nature - PPSN VIII*, volumen 3242 of LNCS, pp. 481–490.

- [CFW98] Chen H., Flann N., y Watson D. (1998) Parallel genetic simulated annealing: a massively parallel simd algorithm. *IEEE Trans. Parallel Distrib. Syst.* 9(2): 126–136.
- [CHS02] Cordon O., Herrera F., y Stützle T. (2002) A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing* 9: 141–175.
- [CK12] Ciornei I. y Kyriakides E. (2012) Hybrid ant colony-genetic algorithm GA-API for global continuous optimization. *IEEE Trans. Syst., Man, Cybern. B* 42(1): 234–245.
- [CL96] Culberson J. C. y Luo F. (1996) Exploring the k-colorable landscape with iterated greedy. In *Dimacs Series in Discrete Mathematics and Theoretical Computer Science*, pp. 245–284. American Mathematical Society.
- [CLM05] Campos V., Laguna M., y Martí R. (2005) Context-independent scatter and tabu search for permutation problems. *INFORMS J. Comput.* 17(1): 111–122.
- [CLPM07] Chen D., Lee C., Park C., y Mendes P. (2007) Parallelizing simulated annealing algorithms based on high-performance computer. *J. Global Optim.* 39(2): 261–289.
- [CLV06] Coello C., Lamont G., y Veldhuizen D. (2006) *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer-Verlag New York, Inc.
- [CMT04] Cahon S., Melab N., y Talbi E. (2004) ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. *J. Heuristics* 10(3): 357–380.
- [COC98] Cho H., Oh S., y Choi D. (1998) A new evolutionary programming approach based on simulated annealing with local cooling schedule. In *Proc. of the Congress on Evolutionary Computation*, pp. 598–602.
- [Coe02] Coello C. (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Comput. Meth. Appl. Mech. Eng.* 191(11-12): 1245 – 1287.
- [Cot98] Cotta C. (1998) A study of hybridisation techniques and their application to the design of evolutionary algorithms. *AI Communications* 11(3-4): 223–224.
- [CS90] Cheng T. y Sin C. (1990) A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research* 47(3): 271 – 292.
- [CTA99] Croce F. D., Tadei R., y Asioli P. (1999) Scheduling a round robin tennis tournament under courts and players availability constraints. *Annals of Operations Research* 92: 349–361.
- [CTA05] Cotta C., Talbi E.-G., y Alba E. (2005) *Parallel Hybrid Metaheuristics*, pp. 347–370. John Wiley & Sons, Inc.
- [CWYH09] Cheng H., Wang X., Yang S., y Huang M. (2009) A multipopulation parallel genetic simulated annealing-based QoS routing and wavelength assignment integration algorithm for multicast in optical networks. *App. Soft Comput.* 9(2): 677–684.
- [DC91] Dodin B. y Chan K. H. (1991) Application of production scheduling methods to external and internal audit scheduling. *European Journal of Operational Research* 52(3): 267 – 279.

- [DDAB09] Dasgupta S., Das S., Abraham A., y Biswas A. (2009) Adaptive computational chemotaxis in bacterial foraging optimization: An analysis. *IEEE Trans. Evol. Comput.* 13(4): 919–941.
- [Deb01] Deb K. (2001) *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley.
- [DMC96] Dorigo M., Maniezzo V., y Colorni A. (1996) The Ant System: Optimization by a colony of cooperating agents. *IEEE Trans. Syst., Man, Cybern. B* 26(1): 29–41.
- [DS04] Dorigo M. y Stützle T. (2004) *Ant Colony Optimization*. MIT Press.
- [DT92] De la Maza M. y Tidor B. (1992) Increased flexibility in genetic algorithms: The use of variable boltzmann selective pressure to control propagation. In *Proc. of the ORSA CSTS Conference - Computer Science and Operations Research: New Developments in their Interfaces*, pp. 425–440.
- [EAK05] El-Abd M. y Kamel M. (2005) A taxonomy of cooperative search algorithms. In MJ B., C B., A R., y M S. (Eds.) *Hybrid Metaheuristics*, volumen 3636 of *LNCS*, pp. 32–41. Springer.
- [EP74] Elmaghraby S. y Park S. (1974) Scheduling jobs on a number of identical machines. *AIIE Transactions* 6(1): 1–13.
- [ES91] Eshelman L. y Schaffer J. (1991) Preventing premature convergence in genetic algorithms by preventing incest. In Belew R. y Booker L. (Eds.) *Int. Conf. on Genetic Algorithms*, pp. 115–122. Morgan Kaufmann.
- [ES03] Eiben A. y Smith J. (2003) *Introduction to Evolutionary Computing*. Springer-Verlag.
- [FL08] Framinan J. y Leisten R. (2008) A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. *OR Spectrum* 30: 787–804.
- [FM93] Forrest S. y Mitchell M. (1993) Relative building block fitness and the building block hypothesis. In Whitley L. (Ed.) *Foundations of Genetic Algorithms 2*, pp. 109–126. Morgan Kaufmann.
- [Fog95] Fogel D. (1995) *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press.
- [FOW66] Fogel L., Owens A., y Walsh M. (1966) *Artificial Intelligence Through Simulated Evolution*. John Wiley.
- [Fox93] Fox B. (1993) Integrating and accelerating tabu search, simulated annealing, and genetic algorithms. *Ann. Oper. Res.* 41: 47–67.
- [FPR10] Fanjul-Peyro L. y Ruiz R. (2010) Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research* 207(1): 55–69.
- [FPR11] Fanjul-Peyro L. y Ruiz R. (2011) Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Computers & Operations Research* 38(1): 301–309.
- [FR89] Feo T. y Resende M. (1989) A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters* 8(2): 67–71.

- [FR95] Feo T. y Resende M. (1995) Greedy randomized adaptive search procedures. *Journal of Global Optimization* 6(2): 109–133.
- [GA07a] Grosan C. y Abraham A. (2007) Hybrid evolutionary algorithms: methodologies, architectures, and reviews. In Grosan C., Abraham A., y Ishibuchi H. (Eds.) *Hybrid Evolutionary Algorithms*, pp. 1–17. Springer.
- [GA07b] Grosan C. y Abraham A. (2007) Hybrid evolutionary algorithms: Methodologies, architectures, and reviews. In *Hybrid Evolutionary Algorithms*, pp. 1–17. Springer.
- [GDLM08] Gortazar F., Duarte A., Laguna M., y Martí R. (2008) Context-independent scatter search for binary problems. Technical report, Colorado LEEDS School of Business, University of Colorado at Boulder.
- [GDLM10] Gortázar F., Duarte A., Laguna M., y Martí R. (2010) Black box scatter search for general classes of binary optimization problems. *Computers & Operations Research* 37(11): 1977–1986.
- [GH11] Glover F. y Hao J.-K. (2011) The case for strategic oscillation. *Annals of Operations Research* 183(1): 163–173.
- [GJ90] Garey M. R. y Johnson D. S. (1990) *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.
- [GK03] Glover F. y Kochenberger G. (Eds.) (2003) *Handbook of Metaheuristics*. Kluwer Academic Publishers.
- [GKD89] Goldberg D., Korb B., y Deb K. (1989) Messy genetic algorithms: motivation, analysis, and first results. *Complex Syst.* 3: 493–530.
- [GL97] Glover F. y Laguna M. (1997) *Tabu Search*. Kluwer Academic Publishers.
- [GLMD11] Gallego M., Laguna M., Martí R., y Duarte A. (2011) Tabu search with strategic oscillation for the maximally diverse grouping problem. *Journal of Operational Research Society* In Press.
- [Glo77] Glover F. (1977) Heuristics for integer programming using surrogate constraints. *Decision Sciences* 8(1): 156–166.
- [Glo96] Glover F. (1996) Tabu search and adaptive memory programming - advances, applications and challenges. In *Interfaces in Computer Science and Operations Research*, pp. 1–75. Kluwer.
- [GML08] García-Martínez C. y Lozano M. (2008) Local search based on genetic algorithms. In Siarry P. y Michalewicz Z. (Eds.) *Advances in Metaheuristics for Hard Optimization*, Natural Computing, pp. 199–221. Springer.
- [GML09a] García-Martínez C. y Lozano M. (2009) Evaluating a local genetic algorithm as context-independent local search operator for metaheuristics. *Soft Comput.* In press.
- [GML09b] García-Martínez C. y Lozano M. (2009) Simulated annealing based on local genetic search. In *Proc. of the IEEE Int. Conf. Evolutionary Computation*, pp. 2569–2576.
- [Gol89] Goldberg D. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co.

- [Gol90] Goldberg D. (1990) A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Syst.* 4: 445–460.
- [HDCL10] Hong W., Dong Y., Chen L., y Lai C. (2010) Taiwanese 3G mobile phone demand forecasting by SVR with hybrid evolutionary algorithms. *Exp. Syst. App.* 37(6): 4452 – 4462.
- [HF06] Hedar A. y Fukushima M. (2006) Derivative-free filter simulated annealing method for constrained continuous global optimization. *J. Global Optim.* 35(4): 521–549.
- [HH06] Hwang S. y He R. (2006) A hybrid real-parameter genetic algorithm for function optimization. *Adv. Eng. Inform.* 20(1): 7–21.
- [HJJ03] Henderson D., Jacobson S., y Jacobson A. (2003) The theory and practice of simulated annealing. In *Handbook of Metaheuristics*, pp. 287–319. Kluwer Academic Publishers Group.
- [HLS05] Herrera F., Lozano M., y Sánchez A. (2005) Hybrid crossover operators for real-coded genetic algorithms: An experimental study. *Soft Comput.* 9(4): 280–298.
- [Hol75] Holland J. (1975) *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.
- [HQ06] Han W. y Que P. (2006) Defect reconstruction of submarine oil pipeline from mfl signals using genetic simulated annealing algorithm. *J. Jpn. Petr. Inst* 49: 145–150.
- [HWJ⁺12] Huang Q., White T., Jia G., Musolesi M., Turan N., Tang K., He S., Heath J., y Yao X. (2012) Community detection using cooperative co-evolutionary differential evolution. In Coello C., Cutello V., Deb K., Forrest S., Nicosia G., y Pavone M. (Eds.) *Parallel Problem Solving from Nature - PPSN XII*, volumen 7492 of *LNCS*, pp. 235–244. Springer Berlin Heidelberg.
- [J.B56] J.B. K. (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematical Society*, pp. 48–50.
- [JB95] Jacobs L. y Brusco M. (1995) A local-search heuristic for large set-covering problems. *Naval Research Logistics* 42: 1129–1140.
- [JBdS92] Jarrah A. I. Z., Bard J. F., y de Silva A. H. (1992) A heuristic for machine scheduling at general mail facilities. *European Journal of Operational Research* 63(2): 192–206.
- [JBT09] Jourdan L., Basseur M., y Talbi E.-G. (2009) Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research* 199(3): 620 – 629.
- [Kar72] Karp R. (1972) Reducibility among combinatorial problems. In Miller R. y Thatcher J. (Eds.) *Complexity of Computer Computations*, pp. 85–103. Plenum Press.
- [Kau89] Kauffman S. (1989) Adaptation on rugged fitness landscapes. *Lec. Sci. Complex.* 1: 527–618.
- [KGOK12] Karaboga D., Gorkemli B., Ozturk C., y Karaboga N. (2012) A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review* In Press: 1–37.

- [KGV83] Kirkpatrick S., Gelatt Jr C., y Vecchi M. (1983) Optimization by simulated annealing. *Sci.* 220(4598): 671–680.
- [KTBS10] Kendall G., Tan K., Burke E., y Smith S. (2010) Preface for the special volume on computational intelligence in scheduling. *Annals of Operations Research* 180: 1–2.
- [Lag03] Laguna M. (2003) *Scatter Search*. Kluwer Academic Publishers Boston, Mass.
- [LB10] Lozano M. y Blum C. (2010) A hybrid metaheuristic for the longest common subsequence problem. In Blesa M., Blum C., Raidl G., Roli A., y Sampels M. (Eds.) *Hybrid Metaheuristics*, volumen 6373 of *LNCS*, pp. 1–15.
- [LGH10] L \tilde{A} $\frac{1}{4}$ Z., Glover F., y Hao J. (2010) A hybrid metaheuristic approach to solving the UBQP problem. *Eur. J. Oper. Res.* 207(3): 1254–1262.
- [LGM10] Lozano M. y García-Martínez C. (2010) Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Comput. Oper. Res.* 37: 481–497.
- [LHKM04] Lozano M., Herrera F., Krasnogor N., y Molina D. (2004) Real-coded memetic algorithms with crossover hill-climbing. *Evol. Comput.* 12(3): 273–302.
- [Liu99] Liu J. (1999) The impact of neighbourhood size on the process of simulated annealing: Computational experiments on the flowshop scheduling problem. *Comput. Ind. Eng.* 37: 285–288.
- [LJ00] Li B. y Jiang W. (2000) A novel stochastic optimization algorithm. *IEEE Trans. Syst., Man, Cybern. B* 30: 193–198.
- [LKH93] Lin F., Kao C., y Hsu C. (1993) Applying the genetic approach to simulated annealing in solving some NP-hard problems. *IEEE Trans. Syst., Man, Cybern.* 23(6): 1752–1767.
- [LLYL11] Lin S.-W., Lee Z.-J., Ying K.-C., y Lu C.-C. (2011) Minimization of maximum lateness on parallel machines with sequence-dependent setup times and job release dates. *Computers & Operations Research* 38(5): 809–815.
- [LM99] Laguna M. y Marti R. (1999) GRASP and path relinking for 2-layer straight line crossing minimization. *Inform. Journal on Computing* 11(1): 44–52.
- [LMGM11] Lozano M., Molina D., y García-Martínez C. (2011) Iterated greedy for the maximum diversity problem. *European Journal of Operational Research* 214(1): 31 – 38.
- [LMPn10] LaTorre A., Muelas S., y Peña J. (2010) A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. *Soft Comput.* pp. 1–13.
- [LMS03] Lourenço H., Martin O., y Stützle T. (2003) Iterated local search. In Glover F. y Kochenberger G. (Eds.) *Handbook of Metaheuristics*, pp. 321–353. Kluwer Academic Publishers.
- [LPF11] Lin Y., Pfund M., y Fowler J. (2011) Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Computers & Operations Research* 38(6): 901–916.

- [LW07] Li X. y Wei X. (2007) An improved genetic algorithm-simulated annealing hybrid algorithm for the optimization of multiple reservoirs. *Water Res. Manag.* 22: 1031–1049.
- [LY09] Li K. y Yang S.-L. (2009) Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Applied Mathematical Modelling* 33(4): 2145–2158.
- [LZT11] Li Z., Zhang Y., y Tan H.-Z. (2011) IA-AIS: An improved adaptive artificial immune system applied to complex optimization problems. *Applied Soft Computing* 11(8): 4692 – 4700.
- [LZXM10] Liu G., Zhou D., Xu H., y Mei C. (2010) Model optimization of svm for a fermentation soft sensor. *Exp. Syst. App.* 37(4): 2708 – 2713.
- [McN59] McNaughton R. (1959) Scheduling with deadlines and loss functions. *Management Science* 6(1): 1–12.
- [MF04] Michalewicz Z. y Fogel D. B. (2004) *How to solve it: modern heuristics*. Springer-Verlag New York, Inc., New York, NY, USA.
- [MG95] Mahfoud S. y Goldberg D. (1995) Parallel recombinative simulated annealing: A genetic algorithm. *Parallel Comput.* 21(1): 1–28.
- [MH97] Mladenovic N. y Hansen P. (1997) Variable neighborhood search. *Comput. Oper. Res.* 24: 1097–1100.
- [MMD10] Marinakis Y., Marinaki M., y Dounias G. (2010) A hybrid particle swarm optimization algorithm for the vehicle routing problem. *Engineering Applications of Artificial Intelligence* 23(4): 463 – 472.
- [Mok01] Mokotoff E. (2001) Parallel machine scheduling problems: A survey. *Asia-Pacific Journal of Operational Research* 18(2): 193 – 242.
- [MS10] Mallipeddi R. y Suganthan P. (2010) Ensemble of constraint handling techniques. *IEEE Trans. Evol. Comput.* 14(4): 561–579.
- [MSV10] Maniezzo V., Stützle T., y VoB S. (Eds.) (2010) *Matheuristics - Hybridizing Metaheuristics and Mathematical Programming*, volumen 10 of *Annals of Information Systems*. Springer.
- [NI08] Noman N. y Iba H. (2008) Accelerating differential evolution using an adaptive local search. *IEEE Trans. Evol. Comput.* 12(1): 107–125.
- [NW88] Nemhauser G. L. y Wolsey L. A. (1988) *Integer and combinatorial optimization*. Wiley-Interscience, New York.
- [PG10] Palubeckis G Rubliauskas D T. A. (2010) Metaheuristic approaches for the quadratic minimum spanning tree problem. *Information Technology and Control* 39(4): 257–268.
- [PGCP00] Pelikan M., Goldberg D., y Cantú-Paz E. (2000) Linkage problem, distribution estimation, and bayesian networks. *Evol. Comput.* 8(3): 311–340.

- [PO08] Pedomallu C. y Ozdamar L. (2008) Investigating a hybrid simulated annealing and local search algorithm for constrained optimization. *Eur. J. Oper. Res.* 185(3): 1230–1245.
- [PR00] Pendharkar P. y Rodger J. (2000) Nonlinear programming and genetic search application for production scheduling in coal mines. *Annals of Operations Research* 95(1): 251–267.
- [PR02] Pitsoulis L. y Resende M. (2002) Greedy randomized adaptive search procedures. In P.M.Pardalos y M.G.C.Resende (Eds.) *Handbook of Applied Optimization*, pp. 168–181. Oxford University Press.
- [PR05] Puchinger J. y Raidl G. (2005) Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In Mira J. y Álvarez J. (Eds.) *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, volumen 3562 of *LNCS*, pp. 113–124. Springer Berlin.
- [PS82] Papadimitriou C. H. y Steiglitz K. (1982) *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., New York.
- [PSL05] Price K., Storn R., y Lampinen J. (2005) *Differential Evolution: A Practical Approach To Global Optimization*. Springer.
- [PT99] Preux P. y Talbi E. (1999) Towards hybrid evolutionary algorithms. *Int. Trans. Oper. Res.* 6(6): 557–570.
- [PTCY10] Peng F., Tang K., Chen G., y Yao X. (2010) Population-based algorithm portfolios for numerical optimization. *IEEE Trans. Evol. Comput.* 14(5): 782–800.
- [Rai06a] Raidl G. (2006) A unified view on hybrid metaheuristics. In Almeida F., Aguilera M. B., Blum C., Vega J. M., Pérez M. P., Roli A., y Sampels M. (Eds.) *Hybrid Metaheuristics*, volumen *LNCS* 4030, pp. 1–12. Springer.
- [Rai06b] Raidl G. R. (2006) A unified view on hybrid metaheuristics. In *Hybrid Metaheuristics*, pp. 1–12.
- [RDGML10] Rodriguez-Diaz F., Garcia-Martinez C., y Lozano M. (2010) A GA-based multiple simulated annealing. In *IEEE Congress on Evolutionary Computation*, pp. 195–201.
- [Rec73] Rechenberg I. (1973) *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog.
- [Rec94] Rechenberg I. (1994) *Evolutionsstrategie'94*. Frommann-Holzboog.
- [Ree93] Reeves C. (1993) *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, Inc.
- [RG87] Rosenbloom E. y Goertzen N. (1987) Cyclic nurse scheduling. *European Journal of Operational Research* 31: 19–23.
- [RG08] R C. y G P. (2008) Heuristic and exact approaches to the quadratic minimum spanning tree problem. In *Seventh Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW08)*, pp. 52–55.

- [RH02] Ribeiro C. y Hansen P. (2002) *Essays and Surveys in Metaheuristics*. Kluwer Academic Publishers.
- [RH05] Rosen S. L. y Harmonosky C. M. (2005) An improved simulated annealing simulation optimization method for discrete parameter stochastic systems. *Computers & Operations Research* 32(2): 343 – 358.
- [RMVCD11] R. Martí V. Campos M. R. y Duarte A. (2011) Multiobjective grasp with path relinking. Technical report, AT&T Labs Research.
- [Roc98] Rochat Y. (1998) A genetic approach for solving a scheduling problem in a robotized analytical system. *Journal of Heuristics* 4: 245–261.
- [RPB10] Raidl G., Puchinger J., y Blum C. (2010) Metaheuristic hybrids. In Gendreau M. y Potvin J.-Y. (Eds.) *Handbook of Metaheuristics*, volumen 146 of *International Series in Operations Research & Management Science*, pp. 469–496. Springer US.
- [RR03] Resende M. y Ribeiro C. (2003) Greedy randomized adaptive search procedures. In Glover F. y Kochenberger G. (Eds.) *Handbook of Metaheuristics*, pp. 219–249. Kluwer Academic Publishers.
- [RS07] Ruiz R. y Stützle T. (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research* 177(3): 2033–2049.
- [RS08] Ruiz R. y Stützle T. (2008) An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research* 187(3): 1143–1159.
- [RUW02] Ribeiro C., Uchoa E., y Werneck R. (2002) A hybrid GRASP with perturbations for the steiner problem in graphs. *Inform Journal on Computing* 14(3): 228–246.
- [SAB88] Sarin S., Ahn S., y Bishop A. (1988) An improved branching scheme for the branch and bound procedure of scheduling n jobs on m parallel machines to minimize total weighted flowtime. *International Journal of Production Research* 26(7): 1183–1191.
- [SCD⁺12] Santamaría J., Cordon O., Damas S., Martí R., y Palma R. (2012) GRASP and path relinking hybridizations for the point matching-based image registration problem. *Journal of Heuristics* 18: 169–192.
- [SHS03] Smith K., Hoos H., y Stützle T. (2003) Iterated robust tabu search for MAX-SAT. In Carbonell J. y Siekmann J. (Eds.) *Proc. of the Canadian Society for Computational Studies of Intelligence Conf.*, volumen 2671 of *LNCS*, pp. 129–144. Springer.
- [Sim08] Simon D. (2008) Biogeography-based optimization. *IEEE Trans. Evol. Comput.* 12(6): 702 –713.
- [SK91] Strenski P. y Kirkpatrick S. (1991) Analysis of finite length annealing schedules. *Algorithmica* 6: 346–366.
- [SM08] Siarry P. y Michalewicz Z. (Eds.) (2008) *Advances in Metaheuristics for Hard Optimization*. Natural Computing. Springer.
- [SP97] Storn R. y Price K. (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* 11(4): 341–359.

- [Spe00] Spears W. (2000) *Evolutionary Algorithms: The Role of Mutation and Recombination*. Springer.
- [SS10] Sundar S. y Singh A. (2010) A swarm intelligence approach to the quadratic minimum spanning tree problem. *Information Sciences* 180(17): 3182 – 3191.
- [SSF02] Salamon P., Sibani P., y Frost R. (2002) *Facts, Conjectures and Improvements for Simulated Annealing*. Monographs on Mathematical Modeling and Computation. SIAM.
- [Tal02] Talbi E. (2002) A taxonomy of hybrid metaheuristics. *J. Heuristics* 8(5): 541–564.
- [TB05] Thompson D. y Bilbro G. (2005) Sample-sort simulated annealing. *IEEE Trans. Syst., Man, Cybern. B* 35(3): 625–632.
- [Thi02] Thierens D. (2002) Adaptive mutation rate control schemes in genetic algorithms. In *Proc. of the Congress on Evolutionary Computation*, pp. 980–985.
- [Thi04] Thierens D. (2004) Population-based iterated local search: restricting neighborhood search by crossover. In Deb K., Poli R., Banzhaf W., Beyer H.-G., Burk E., Darwen P., Dasgupta D., Floreano D., Foster J., Harman M., Holland O., Lanzi P., Spector L., Tettamanzi A., Thierens D., y Tyrrel A. (Eds.) *Proc. of the Genetic and Evolutionary Computation Conf.*, volumen 3103 of *LNCS*, pp. 234–245. Springer.
- [TW10] Tang L. y Wang X. (2010) An improved particle swarm optimization algorithm for the hybrid flowshop scheduling to minimize total weighted completion time in process industry. *IEEE Trans. Control Syst. Technol.* 18(6): 1303–1314.
- [URS10] Urlings T., Ruiz R., y Stützle T. (2010) Shifting representation search for hybrid flexible flowline problems. *European Journal of Operational Research* 207(2): 1086 – 1095.
- [VA87] Van Laarhoven P. y Aarts E. (1987) *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers Norwell.
- [VH02] Vredeveld T. y Hurkens C. (2002) Experimental comparison of approximation algorithms for scheduling unrelated parallel machines. *Inform. Journal on Computing* 14(2): 175–189.
- [VOR99] Voß S., Osman I., y Roucairol C. (1999) *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers Norwell.
- [VRH09] Vrugt J., Robinson B., y Hyman J. (2009) Self-adaptive multimethod search for global optimization in real-parameter spaces. *IEEE Trans. Evol. Comput.* 13(2): 243–259.
- [VT07] Ventresca M. y Tizhoosh H. (2007) Simulated annealing with opposite neighbors. In *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on*, pp. 186 –192.
- [W95] W X. (1995) A hybrid metaheuristic for the longest common subsequence problem. In M G. y W X. (Eds.) *Proceedings of 1995 Japan–China International Workshops on Information Systems*, pp. 141–148.

- [WLR01] Weng M., Lu J., y Ren H. (2001) Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics* 70(3): 215–226.
- [Wot07] Wotzlaw A. (2007) *Scheduling Unrelated Parallel Machines: Algorithms, Complexity, and Performance*. VDM Verlag, Germany.
- [WP99] Watson R. y Pollack J. (1999) Hierarchically consistent test problems for genetic algorithms. In *Proc. of the Congress on Evolutionary Computation*, volumen 2, page 1413.
- [WWR05] Wang Z., Wong Y., y Rahman M. (2005) Development of a parallel optimization method based on genetic simulated annealing algorithm. *Parallel Comput.* 31(8-9): 839–857.
- [XSVB06] Xavier-de-Souza S., Suykens J., Vandewalle J., y Bollé D. (2006) Cooperative behavior in coupled simulated annealing processes with variance control. In *Symposium on Nonlinear Theory and its Applications*, pp. 114–119.
- [Yao91] Yao X. (1991) Simulated annealing with extended neighbourhood. *Int. J. Comput. Math.* 40: 169–189.
- [YC10] Ying K.-C. y Cheng H.-M. (2010) Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. *Expert Systems with Applications* 37(4): 2848–2852.
- [YLJ12] Yu J., Lee S.-H., y Jeon M. (2012) An adaptive ACO-based fuzzy clustering algorithm for noisy image segmentation. *International Journal of Innovative Computing Information and Control* 8(6): 3907–3918.
- [YP95] Yip P. y Pao Y. (1995) Combinatorial optimization with use of guided evolutionary simulated annealing. *IEEE Trans. Neural Netw.* 6(2): 290–295.
- [YTY07] Yang Z., Tian Z., y Yuan Z. (2007) GSA-based maximum likelihood estimation for threshold vector error correction model. *Comput. Stat. Data Anal.* 52(1): 109 – 120.
- [ZG98] Zhou G. y Gen M. (1998) An effective genetic algorithm approach to the quadratic minimum spanning tree problem. *Computers & Operations Research* 25(3): 229–237.
- [ZJLK10] Zaidi M., Jarboui B., Loukil T., y Kacem I. (2010) Hybrid meta-heuristics for uniform parallel machine to minimize total weighted completion time. In *Proc. of 8th International Conference of Modeling and Simulation (MOSIM'10)*.
- [ZWJZ08] Zhang Q., Wang J., Jin C., y Zeng Q. (2008) Localization algorithm for wireless sensor network based on genetic simulated annealing algorithm. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, pp. 1–5.