Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Cinvestav Tamaulipas

Algoritmos avanzados para el Problema de la Suma del Ancho de Banda Cíclico en Grafos

Tesis que presenta:

María Valentina Narváez Terán

Para obtener el grado de:

**Doctor en Ciencias
en Ciencias en Ingeniería y Tecnologías Computacionales**

Director de la Tesis:
Dr. Eduardo Arturo Rodríguez Tello

Cd. Victoria, Tamaulipas, México. December, 2021

RESEARCH CENTER FOR ADVANCED STUDY
FROM THE NATIONAL POLYTECHNIC INSTITUTE

Cinvestav Tamaulipas

Advanced algorithms for the Cyclic Bandwidth Sum Problem in Graphs

Thesis by:

María Valentina Narváez Terán

as the fulfillment of the
requirement for the degree of:

**Doctor of Science
in Engineering and Computing
Technologies**

Thesis Director:
Dr. Eduardo Arturo Rodríguez Tello

Cd. Victoria, Tamaulipas, México. December, 2021

The thesis of María Valentina Narváez Terán is approved by:

Dr. Gregorio Toscano Pulido

Dr. José Torres Jiménez

Dr. Ricardo Landa Becerra

Dr. Broderick Crawford Labrín

Dr. Eduardo Arturo Rodríguez Tello, Committe Chair

Cd. Victoria, Tamaulipas, México., December 14 2021

Contents

List of Figures

List of Tables

List of Algorithms

Resumen

Algoritmos avanzados para el Problema de la Suma del Ancho de Banda Cíclico en Grafos

por

María Valentina Narváez Terán
Cinvestav Tamaulipas
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2021
Dr. Eduardo Arturo Rodríguez Tello, Director

El problema de la suma del ancho de banda cíclico (CBSP) es un problema de embebido de grafos. Esta clase de problemas consiste en la incrustación de los vértices y aristas de un grafo huésped dentro de los vértices y caminos de un grafo anfitrión. En el problema de la suma del ancho de banda cíclico, el objetivo es minimizar la suma de distancias cíclicas de los vértices adyacentes del grafo huésped embebidos en el anfitrión, el cuál tiene una topología de ciclo. Se trata de un problema de optimización combinatoria que pertenece a la clase NP-hard.

En este trabajo, se estudiaron los aspectos que influyen en la dificultad del CBSP, se desarrollaron mejores métodos de solución y se propuso un análisis del problema desde una perspectiva integral. La dificultad del CBSP es influenciada por la baja capacidad de discriminación de la función objetivo, causando neutralidad y multimodalidad en el paisaje de aptitud, lo cual puede afectar negativamente el desempeño de los algoritmos de búsqueda. Con el fin de lidiar con la neutralidad y multimodalidad, se propuso una función de evaluación alternativa capaz de incrementar la capacidad de discriminación entre soluciones, manteniendo la compatibilidad con el objetivo del problema. Al ser incorporada en diversos algoritmos de búsqueda, esta función permitió mejorar significativamente su desempeño. Los efectos de la función de evaluación alternativa fueron estudiados mediante el análisis del paisaje de aptitud, comprobando que permite reducir la neutralidad y multimodalidad, mientras mantiene la estructura global del paisaje de aptitud. Los enfoques de solución propuestos consideran el compromiso entre calidad de solución y tiempo de ejecución ofrecido por las técnicas metaheurísticas, la capacidad adaptativa de las hiperheurísticas y la obtención de óptimos mediante algoritmos exactos.

Se llevó a cabo un análisis extensivo de diferentes configuraciones de operadores genéticos en el marco de un algoritmo memético y su interacción con la función de evaluación alternativa, obteniendo un algoritmo memético capaz de mejorar significativamente los resultados reportados en la literatura. Se implementó un bandido multiarmado dinámico como método hiperheurístico para la selección automática de los operadores y de la función de evaluación en el algoritmo memético. El resultado fue una mejora significativa en la calidad de solución obtenida. En este trabajo se propusieron los primeros enfoques de tipo exacto para el problema, consistiendo en modelos programación con restricciones y un algoritmo de ramificación y poda. Comparativamente, el uso de programación con restricciones obtuvo un mejor desempeño en términos de calidad de solución y tiempo de ejecución. Finalmente, este trabajo reporta un análisis conjunto ligando el paisaje de aptitud, las características de las instancias del problema y el desempeño observado en los algoritmos propuestos. Lo anterior reveló nuevo conocimiento sobre la interacción de dicho factores y su efecto en la dificultad.

Abstract

Advanced algorithms for the Cyclic Bandwidth Sum Problem in Graphs

by

María Valentina Narváez Terán
Cinvestav Tamaulipas
Research Center for Advanced Study from the National Polytechnic Institute, 2021
Dr. Eduardo Arturo Rodríguez Tello, Advisor

The cyclic bandwidth sum problem (CBSP) is a graph embedding problem. These problems consist in the embedding of the vertices and edges of a guest graph in the vertices and paths of a host graph, respectively. In the CBSP the objective is to minimize the sum of the cyclic distances for adjacent guest vertices embedded in a cycle host graph. This is an $\mathcal{NP}$-hard problem.

This work extended to the study of aspects influencing the CBSP difficulty, the development of improved solution approaches and the analysis of the problem from a global perspective. The CBSP difficulty is influenced by the low discrimination ability of the objective function, which causes neutrality and multimodality in the fitness landscape of the problem and can negatively affect the performance of search algorithms. In order to address the neutrality and multimodality issues, an alternative evaluation function was proposed. This function was proven able to increase the discrimination ability while being compatible with the problem's objective. It was also demonstrated to help improving the results of search algorithms. The effects of the alternative evaluation function on the fitness landscape were studied, showing that it reduces the neutrality and multimodality, while mostly preserving the global structure of the fitness landscape.

The proposed solution approaches considered the compromise between solution quality and execution time offered by metaheuristics, the adaptive ability of hyperheuristics and the production of optimal solutions offered by exact algorithms. An extensive analysis of different operator configurations within a memetic algorithm and their interaction with the alternative evaluation function was performed, resulting in a memetic algorithm able to significantly improve previously reported results. A dynamic multi-armed bandit hyperheuristic approach was implemented in order to automatize the operator selection of the memetic algorithm. The result was a significant improvement of the solution quality.

The first exact approaches for the CBSP were proposed, consisting in the use of constraint programming and a branch and bound algorithm. Comparatively, the former approach had the best performance, considering solution quality and execution time. Finally, this work reports an analysis that links fitness landscape features, instance specific features and the performance of the proposed solution approaches, revealing new knowledge on the interaction of these features and problem difficulty.

1 Introduction

1.1 Antecedents

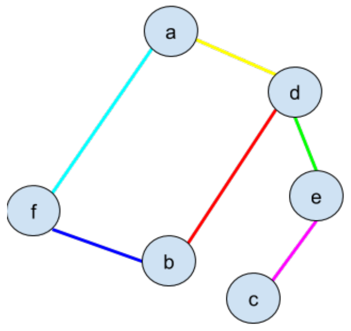
A graph G is a mathematical object that consists of a set of vertices V and a set of edges E. The order of G is given by its number of vertices $n = |V|$, while its number of edges $e = |E|$ is the size of the graph. Two vertices u and v are adjacent if, and only if, there is an edge $(u, v) \in E$. For the scope of this document, the referred graphs are simple, undirected, and finite. A graph is undirected if the set of edges E is a set of non-ordered pairs of elements of V. If for each edge $(u, v) \in E$ it is true that $u \neq v$ and there is only one edge between each pair of adjacent vertices, then the graph is a simple graph. A graph is finite if the set of vertices is finite. For further information in terminology, see [42, 114].

Graphs are widely employed for representing a broad variety of problems in numerous subareas of computer science, from the design of circuits, network infrastructures and data bases, to social network analysis and automatized route generation. In the combinatorial optimization area there are multiple problems involving graphs, such as the Traveling Salesman Problem, Graph Coloring, Max Vertex Cover, etc.

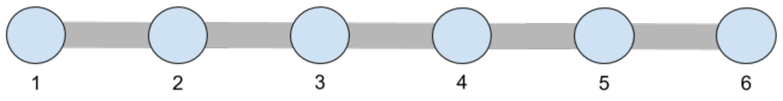
Graph Embedding Problems (GEP) are a family of combinatorial problems, which objective is to find the best way of embedding a *guest* graph $G = (V, E)$ into a *host* graph $H = (V_H, E_H)$, in such a way that the vertices of the guest are assigned to host vertices and its edges are placed into paths of the host graph. Figure 1.1 illustrates an example of graph embedding. The guest graph G is shown in Figure 1.1(a), with colored edges to visually identify their the correspondent host paths in the embedding. The host graph H is the path graph P_6 of size $n = 6$ in Figure 1.1(b), its edges are gray and thick to better picture the embedding of guest edges. Figure 1.1(c) shows how guest graph's vertices are embedded into host graph's vertices, and how the edges of the guest graph are embedded into host graph's paths.

Several tasks across different areas of computer science can be modeled as GEP instances. Some relevant examples include: automatized graph drawing [84, 120], error correcting codes [49], modeling of channel disposition in electric circuits [75], parallel and distributed computer architecture simulation [81], code theory [12], multidiffusion network design, compressed sensing in sensor networks [69], etc.

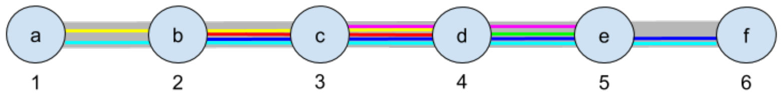
In the general case, which corresponds to guest graphs of any order size and topology, finding optimal solutions for the GEP is a complex and non-viable task. This is because the number of potential solutions for this type of problems grows at factorial rate, and there is no algorithm able to solve them in polynomial time. However, in practical applications, approximated solutions for optimization problems are often accepted as good enough. It has been demonstrated that metaheuristic algorithms are able to get solutions very close to the optimum in acceptable execution time [30, 74, 79, 95]. The performance of these types of algorithms has been particularly successful when the design of their essential components includes relevant information concerning the problem. For example, by implementing neighborhood functions suitable for an efficient exploration of the search space [80, 112], or alternative evaluation schemes to better guide the search [107, 108].



(a) Guest graph G.



(b) Host graph $H = P_6$.



(c) An embedding of G into P_6.

Figure 1.1: Embedding of a guest graph G into a host path graph $H = P_6$. Vertices and edges of G are embedded into vertices and paths of H. Guest edges were colored for identifying them in the embedding.

Specific GEP can be distinguished from each other by their optimization objective and the topology of the host graph. GEP can be classified into three main categories [20] according to the objective of the problem.

- Bandwidth: the objective is to optimize the distance among guest nodes when embedded in host nodes.
- Bandwidth sum: the sum of distances among guest nodes when embedded in host nodes.
- Cudwidth: the number of disrupted guest edges affected by a cut on any host edge.

The bandwidth problem, bandwidth sum problem and cutwidth problem exemplify each of the three GEP categories, where in all cases, the host graph is a path graph. The cyclic bandwidth [67], the cyclic bandwidth sum [59] and the cyclic cutwidth [71] are similar problems, where the host graph is a cycle instead.

The cyclic bandwidth sum problem (CBSP) is a GEP where the host graph is a cycle and the objective is to minimize the sum of cyclic distances. The CBSP is is the main object of study of this thesis.

1.2 Motivation

The CBSP is a challenging subject with relevancy in the areas of graph theory and combinatorial optimization. As other GEP [28], it is also an $\mathcal{NP}$-Hard problem [139].

Formulas for calculating the theoretical value of the optimum have been reported [58], for some specific graph topologies such as paths, cycles, wheels, complete bipartite graphs, cycle powers, etc. Upper bounds for some other topologies, specifically Cartesian products of paths, cycles and complete graphs have also been established [18].

In 2012, Satsangi et al. [117] proposed a General Variable Neighborhood Search (GVNS) algorithm. GVNS was able to achieve optimal solutions for graphs with known optimal values. Later in 2016 Hamon et al. [48] developed a heuristic, named MACH, for solving the CBSP by following the guest graph structure. MACH was able to reach theoretical optimums for specific graph topologies in less time than GVNS. Two new algorithms for the CBSP were proposed in 2014, a Memetic Algorithm (MA) and a Basic Variable Neighborhood Search [85], that improved the results reached by Hamon et al. [48], specially in the case of guest graphs with random topologies and graphs derived from sparse matrices.

While these proposals exist, the development of algorithms able to efficiently solve the CBSP presents an opportunity for proposing solution approaches that improve previous results. Another relevant aspect, not previously approached in the literature of this problem, is the study of the difficulty of the problem by analysing its fitness landscape.

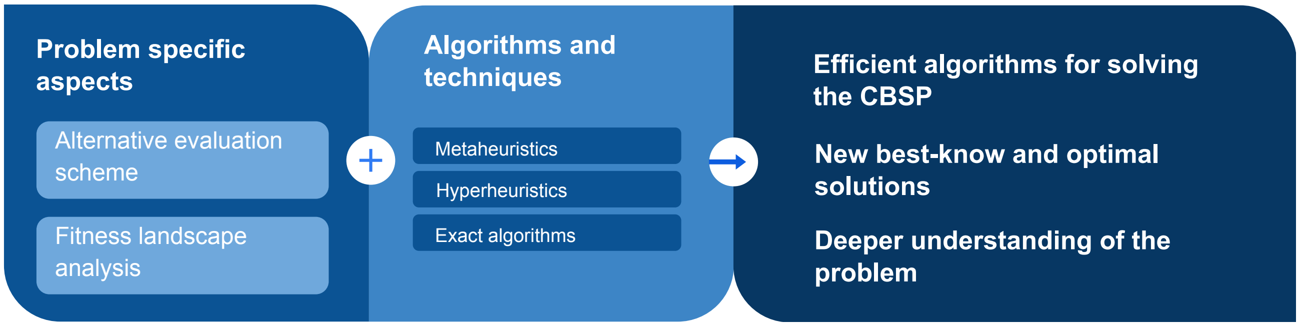

Problem specific aspects
Alternative evaluation scheme
Fitness landscape analysis
Algorithms and techniques
Metaheuristics
Hyperheuristics
Exact algorithms
Efficient algorithms for solving the CBSP
New best-know and optimal solutions
Deeper understanding of the problem

Figure 1.2: Outline of the proposal main aspects.

1.3 Solution approach

Some factors associated to the difficulty of the CBSP are the neutrality and multimodality of its fitness landscape, and they are directly related to the objective function. In other combinatorial problems this type of challenges have been addressed by proposing alternative evaluations schemes, that effectively transform the relationships among potential solutions in such a way that they can be exploited by search methods [104, 107, 108].

In regards to the evaluation function of the CBSP, this thesis proposes the design of a new evaluation scheme for the problem, which considers not only the sum of cyclic distances but also the distribution of their values, as a mean to reduce neutrality by prioritizing solutions that are easier to be further improved. Such approach is to be studied in depth, addressing its impact on the performance of particular solution methods and its effects on the structure of the fitness landscape. The proposal extends to the design and implementation of efficient algorithms for the CBSP. It was considered worth exploring the use of metaheuristics and exact algorithms, as well as tools such as the automatized selection of operators via hyperheuristic methods.

Figure 1.2 summarizes the proposal as a combination of the aforementioned problem specific aspects as well as the algorithms and techniques in order to develop solution methods for the CBSP, able to improve the current results from the literature and in the process, to gain further knowledge about the problem and its difficulty.

1.4 Hypothesis

The research hypothesis can be stated as follows.

- The analysis of the fitness landscape of the CBSP, specially related to the evaluation function, will allow us to 1) understand the aspects that influence the problem difficulty and 2) design efficient algorithms able to manage the challenges associated to the optimization problem, such as neutrality and multimodality.

1.5 Aim of the thesis

The main objective of this research is:

- To develop advanced (exact and approximate) algorithms for solving the CBSP that are competitive with those from the literature in terms of solution quality, execution time, and maximum size of the problem instances processed.

To this end, the following specific objectives must be fulfilled.

- Propose an alternative evaluation scheme for the CBSP that reduces the neutrality, while being consistent with the objective of the problem and provides a better guidance for search based methods helping them to improve their performance.
- Study the interactions of different types of genetic operators and evaluation schemes in order to propose an improved memetic algorithm for the CBSP.
- Propose a hyperheuristic algorithm able to self-adapt its configuration of multiple operators and the evaluation schemes during execution time, achieving improved results in a consistent way with respect to variations on the topology of the instances.

- Study the difficulty of the CBSP by analyzing its fitness landscape, the effects of the proposed alternative evaluation scheme on the fitness landscape, the behavior of search algorithms when exploring it and the relation with instance features and difficulty.

- Propose the first exact algorithm for the CBSP able to solve instances of $n \leq 40$ vertices.

1.6 Outline of the document

The following is a summary of the content of the remaining chapters of the document.

- Chapter 2 formally defines the problem and it summarizes the related work on the CBSP . It also presents background concepts about the fitness landscape analysis, the type of algorithms and techniques involved with this work.

- Chapter 3 introduces three alternative evaluation schemes for the CBSP. The ability of each of the new evaluation schemes to provide more discrimination among solution and guide local search-based algorithms are assessed experimentally and compared with the conventional evaluation function.

- Chapter 4 presents a comparative of Memetic Algorithms for the CBSP employing different genetic operators and comparing the conventional evaluation function with the best performing of the alternative evaluation schemes from Chapter 3. The chapter includes a performance evaluation of the different MA versions compared with that of previously reported algorithms, and it discusses the strengths of certain combinations of operators for producing an efficient MA for the CBSP.

- The use of the Dynamic Multi-Armed Bandit as a hyper-heuristic approach to adapt the use of the different MA operators is explored in Chapter 5. Experimental results of this technique are compared with the results of the best-performing MA and the state-of-the-art algorithms.

- Chapter 6 presents the fitness landscape analysis of the CBSP, studying how its features are affected by the proposed alternative evaluation function. The chapter examines the relationship among instance features, the fitness landscape characteristics and the performance of algorithms. The search dynamics of the proposed memetic algorithm and the dynamic multi-armed bandit based implementation are analyzed using a recently devised technique for the visualization and measurement of search trajectories.

- Chapter 7 introduces the proposal for modeling and solving the CBSP as a constraint programming problem. An initial model was built and iteratively refined by the incorporation of new constraints based on the known exact, upper and lower bounds of the optimal value according to instance topology. The chapter presents an assessment of the success in solving relatively small CBSP instances ($n \leq 40$) when implementing different strategies for exploring the search space.

- Chapter 8 is a summary of the main findings and achieved contributions produced across the different aspects contemplated in this research. This includes the proposed metaheuristic, hyperheuristic and exact algorithms, the alternative evaluation scheme, its impact on the search process and on the fitness landscape, as well as the relationship among instance features, fitness landscape and problem difficulty. Finally, a general outline for further research topics derived from this research is presented.

2

State of the art

2.1 Introduction

This chapters provides an overview of the current literature on the CBSP in order for the reader to understand the starting point of this research. The chapter also introduces the main algorithms and techniques involved in the proposal.

The formal definition of the CBSP and the proof of its complexity are presented in Section 2.2.1 and Section 2.2.3, respectively. Section 2.3 summarizes the related work, including the theoretical bounds of the optimal value and the algorithms to solve it that have been reported in the literature. Meanwhile, Section 2.4 introduces concepts related to the algorithms and techniques involved with the proposed solution approach to the CBSP.

2.2 The Cyclic Bandwidth Sum Problem

The CBSP was originally formulated by Yuan in 1995 [59]. Later, Yuan proved that the CBSP belongs to the $\mathcal{NP}$-Hard class. The problem arises in some relevant application areas like VLSI designs [10], code design [49], simulation of network topologies for parallel computer systems [81], scheduling in broadcasting based networks [70], and sensor networks [69].

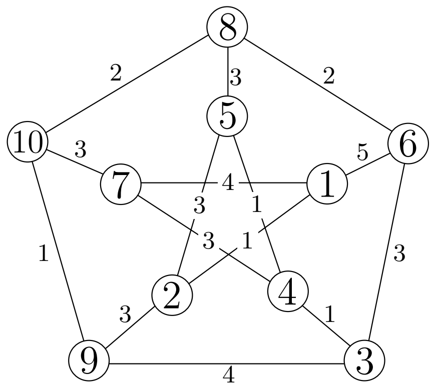
2.2.1 Formal definition of the CBSP

The CBSP is formally defined as follows. Let $G = (V, E)$ be a simple finite undirected graph (the guest) of order n, and C_n a cycle graph (the host) with vertex set $|V_H| = n$ and edge set E_H. Given an injection $\varphi : V \to V_H$, representing an embedding of G into C_n, the Cyclic Bandwidth Sum (CBS) for G with respect to φ is defined as:

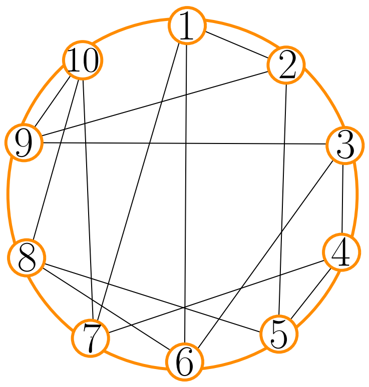
$$\text{CBS}(G, \varphi) = \sum_{(u,v) \in E} |\varphi(u) - \varphi(v)|_n \, , \tag{2.1}$$

where $|x|_n = \min\{\, |x|,\ n - |x| \,\}$ (with $1 \leq |x| \leq n - 1$) is called the *cyclic distance*, and the vertex in V_H associated to vertex $u \in V$ is denoted by the label $\varphi(u)$. The CBSP consists in finding the optimal embedding φ^*, such that CBS(G, φ^*) is minimum, i.e., $\varphi^* = \arg\min_{\varphi \in \Phi}\{\text{CBS}(G, \varphi)\}$ with Φ denoting the set of all possible embeddings. The embedding that satisfies this condition is the optimal embedding. An embedding can be seen as a labeling of the guest graph G, using as labels the vertices of the host graph H, numbered from 1 to n. Because of this reason, the terms *embedding* and *labeling* are often used interchangeably.

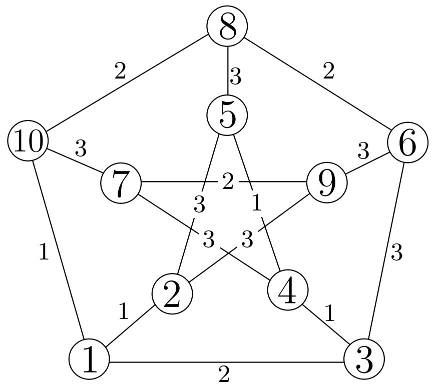
For example, consider the graph $G = (V, E)$ of order $n = 10$ in the Figure 2.1(a), with the embedding φ given by the numbers shown inside each vertex. The cyclic distance for each edge $(u, v) \in E$ is evaluated using the expression $\min\{|\varphi(u) - \varphi(v)|, n - |\varphi(u) - \varphi(v)|\}$, and represented by the number associated to that edge. In this particular example, the CBS of G is CBS$(G, \varphi) = 39$.



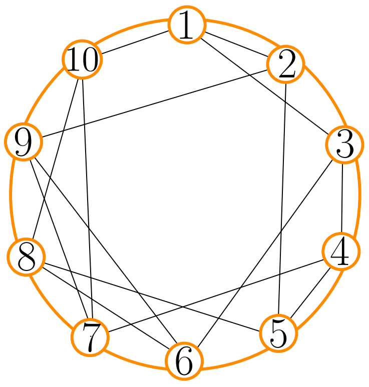
(a) An embedding φ for graph G.



(b) G embedding in the cycle C_n respect to φ.



(c) An embedding φ' for the graph G.



(d) G embedding in the cycle C_n respect to φ'.

Figure 2.1: Example of a CBSP instance.

The embedding φ' presented in the Figure 2.1(c) produces the value CBS$(G, \varphi') = 33$, which is better. The resulting embeddings of graph G into C_n with respect to φ and φ' are presented in the Figures 2.1(b) and 2.1(d), respectively.

2.2.2 Search space size

The search space of the problem is composed by all the possible embeddings. Since the host graph is a cycle, the number of possible embeddings equals to the number of possible ways to assign n items to n slots arranged circularly, i.e., cyclic permutations. Therefore, the size of the search space for the problem is:

$$|\Phi| = \frac{(n-1)!}{2}\,. \tag{2.2}$$

2.2.3 Problem complexity

The complexity theory studies problems in relationship to how demanding the process of solving them can be, quantifying this in terms of the amount of resources, whether in space or time, that are required. Problems are classified into classes, according to said amount of resources, with the classes that require more resources being considered harder.

A decision problem belongs to the class $\mathcal{P}$ if a deterministic Turing machine can be solve it in polynomial time. The class $\mathcal{NP}$ includes decision problems that can solved in polynomial time by a non-deterministic machine and can also be verified by a deterministic Turing machine in polynomial time [4]. The class $\mathcal{P}$ is contained within the class $\mathcal{NP}$, since it is obvious that any problem that can be solved in polynomial time by a deterministic machine can also be solved in polynomial time by a non-deterministic one. An open question is if $\mathcal{P} = \mathcal{NP}$, i.e., if $\mathcal{NP}$ problems could be solved in polynomial time by a deterministic machine [35, 121]. However, there is a strong suspicion that $\mathcal{P} \neq \mathcal{NP}$, being this a common assumption.

A problem belongs to a particular class if it can be established a polynomial time reduction between said problem and a known member of the class [2]. The class-complete problems are problems to which all the other members of their class can be reduced to in this way. The $\mathcal{NP}$-Complete problems are therefore decision problems to which all other decision problems in $\mathcal{NP}$ can be reduced to. The problem of boolean satisfactibility belongs to the class $\mathcal{NP}$ and it was the first problem proven to be $\mathcal{NP}$-Complete [4].

The optimization problems are characterized by a set of instances, a set of solutions, a cost function and an objective. Depending on the nature of the solution [4], they can be stated as:

- Construction: to build an optimal solution
- Evaluation: to find the value of the optimum
- Decision: to determine if there is an optimum of cost $\leq k$.

An optimization problem belongs to the $\mathcal{NP}$-Hard class if their associated decision problem can be reduced to problems in $\mathcal{NP}$ in polynomial time. Optimization problems in the $\mathcal{NP}$-Hard class are at least as difficult as $\mathcal{NP}$ problems, being stated that the decision problem can not be harder than the constructive one. Yuan [59] proved that the CBSP belongs to the $\mathcal{NP}$-Hard class by establishing a polynomial time reduction relationship between the CBSP and the also $\mathcal{NP}$-Hard Bandwidth Sum Problem (BSP) [20, 28].

The BSP is another GEP mathematically related to the CBSP (see Section 2.3.1.1). A similarity between the problems is that both of them consist on minimizing a sum of distances, but in the case of the BSP, the host graph is a path instead of a cycle, meaning that the distances are *linear* instead of *cyclic*. Formally, the BSP is defined as follows. Let $G = (V, E)$ be a simple finite undirected graph (the guest) of order n, and P_n a path graph (the host) with vertex set $|V_H| = n$ and edge set E_H. Given an injection $\psi : V \to V_H$, representing an embedding of G into P_n, the Bandwidth Sum (BS) for G with respect to ψ is defined as:

$$\mathsf{BS}(G, \psi) = \sum_{(u,v)\in E} |\psi(u) - \psi(v)| \,, \tag{2.3}$$

where $|\psi(u) - \psi(v)|$ is the (linear) distance for edge $(u, v) \in E$, and the vertices in V_H associated to vertices $u, v \in V$ are denoted by the labels $\psi(u)$ and $\psi(v)$. The BSP consists in finding the optimal embedding ψ^*, such that $\mathsf{BS}(G, \psi^*)$ is minimum, i.e., $\psi^* = \arg\min_{\psi\in\Psi}\{\mathsf{BS}(G, \psi)\}$, with Φ denoting the set of all possible embeddings.

Yuan's work [59] proved that the CBS can be solved for graph G, by constructing, in polynomial time, another graph G', and solving the BSP for that graph. Graph $G' = (V_{G'}, E_{G'})$ is created with order $n' = |V_{G'}| = 2n^3 + 1$, such that graph G is a subgraph of G' and $G' - E = S_{2n^3}$, i.e., if the edges of G are removed from G', the result is the star graph S_{2n^3}. The central node of the star subgraph is a node $v_0 \in E_{G'}$ such that $v_0 \notin E_G$. Notice graph G' can be built from G in polynomial time.

The core of the demonstration is then showing that there is two spanning subgraphs of G' that share no edges: one of which has the same edge set than the star S_{2n^3}, and another which has the same edge set than G. Yan then proves that the second graph can not have overstep edges, this is edges for which $|\psi(u) - \psi(v)| \geq n/2$, implying cyclic distances are equal to the linear ones. Thus, the CBS for graph G' would be:

$$CBS(G') = BS(G') = BS(G) + n^3(n^3 + 1) \,, \tag{2.4}$$

where $n^3(n^3 + 1)$ is the BS for the star S_{2n^3}.

2.2.4 Differences from the Bandwidth Sum Problem

While the problems can be reduced to each other, in practice, the difference in host graph topology is relevant in the creation of solving algorithms for GEP. Therefore, n algorithm designed specifically

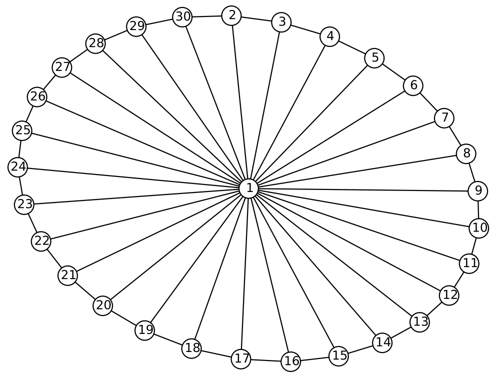


Figure 2.2: A wheel graph W_{30} of order $n = 30$.

for one of these problems may not have a good performance for the other. Consider as example the wheel graph W_{30} of order $n = 30$ in Figure 2.2. For this graph, 1,000 locally optimal embeddings were generated by a first-improvement hill-climber algorithm for the CBSP, starting from random solutions and running by 10,000 iterations. The cost of each embedding was evaluated for both the CBSP and BSP as depicted in Equations 2.1 and 2.3, respectively.

Pearson's correlation coefficient between both cost samples was little ($r = 0.28$), showing that the correlation between the problems is weak. Also, for this problem instance, an embedding with the optimal cost for the CBSP (CBS$(G, \varphi^*) = 255$) has a cost of 491 for the BSP, which is far away from the optimal cost for this instance (BS$(G, \varphi^*) = 283$). Thus, there is a need to develop algorithms especially dedicated to the resolution of the CBSP.

2.3 Existing solution approaches

2.3.1 Calculating the value of the optimum

Early approaches to the CBSP were focused on the mathematical nature of the problem, particularly on the calculation of the value of the optimum respect to the topology of the guest graph. In this regard, there are topologies for which this value can be calculated exactly, and other ones for which there are lower and upper bounds.

2.3.1.1 *Topology independent bounds*

Upper bound for any graph. In 2001, Jianxiu [58] calculated an upper bound for any guest graph G, independently of its topology, with respect to its number of vertices n and edges e:

$$\mathsf{CBS}(G) \leq \frac{e\left\lfloor\frac{n}{2}\right\rfloor\left\lceil\frac{n}{2}\right\rceil}{n-1}\,. \tag{2.5}$$

Upper and lower bounds respect to the BSP. In 2007, Chen and Yan [18] established the following topology independent relationship between the optimal values of the BSP and the CBSP:

$$\frac{\mathsf{BS}(G)}{2} \leq \mathsf{CBS}(G) \leq \mathsf{BS}(G)\,, \tag{2.6}$$

where $\mathsf{BS}(G)$ is the Bandwidth Sum of G detailed in Equation **??**.

2.3.1.2 *Optimal value for specific topologies*

Chen and Yan [18] established the formulas for calculating the value of the optimum for the following specific graph topologies.

Path: A graph P_n with $n = |V|$ vertices $V = \{v_1, v_2, v_3, \ldots, v_{n-1}, v_n\}$ connected by $e = n-1$ edges such as $E = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)\}$.

$$\mathsf{CBS}(P_n) = n - 1\,. \tag{2.7}$$

Cycle: A graph C_n with n vertices $V = \{v_1, v_2, v_3, \ldots, v_{n-1}, v_n\}$ connected by $e = n$ edges such as $E = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n), (v_n, v_1)\}$.

$$\mathsf{CBS}(C_n) = n\,. \tag{2.8}$$

Wheel: A graph W_n consisting of a cycle C_{n-1} and a vertex $v_n \in V$, such as $(v_i, v_n) \forall v_i \in C_{n-1}, v_n \notin C_{n-1}$. A wheel has n vertices and $e = 2n-2$ edges.

$$\mathsf{CBS}(W_n) = n + \left\lfloor \frac{1}{4} n^2 \right\rfloor . \tag{2.9}$$

Complete: A graph K_n with n vertices, with an edge connecting each pair of vertices.

$$\mathsf{CBS}(K_n) = \begin{cases} \frac{n^3}{8} & n \text{ is even} \\ \frac{n(n-1)(n+1)}{8} & n \text{ is odd} \end{cases} . \tag{2.10}$$

Complete Bipartite: A graph $K_{x,y}$ with two disjoint subsets of vertices X and Y with sizes $|X| = x$ and $|Y| = y$. There is an edge between a pair of vertices u and v if, and only if, u and v belong to different subsets.

$$\mathsf{CBS}(K_{x,y}) = \begin{cases} \frac{xy^2+x^2y}{4} & x, y \text{ are even} \\ \frac{xy^2+x^2y+x}{4} & x \text{ is even, } y \text{ is odd} \\ \frac{xy^2+x^2y+x+y}{4} & x, y \text{ are odd} \\ \frac{xy^2+x^2y+y}{4} & x \text{ is odd, } y \text{ is even} \end{cases} . \tag{2.11}$$

Power of cycle: Let C_n be the cycle graph or order n. The k-th power of C_n, denoted C_n^k is a graph of order n, where a pair of vertices u and v are adjacent if its distance (the length of the shortest path between them) is not greater than k.

$$\mathsf{CBS}(C_n^k) = \frac{1}{2} nk(k+1), \quad 1 \le k \le \left\lfloor \frac{n-1}{2} \right\rfloor . \tag{2.12}$$

2.3.1.3 Upper bounds for the Cartesian products of graphs

Let $V(G_n)$ and $E(G_n)$ be the set of vertices and edges of a graph G_n of order n. The Cartesian product of two graphs G_n and H_m, denoted $G_n \times H_m$ is the graph with the set of vertices $V(G_n) \times V(H_m)$ where $[u_1, v_1]$ is adjacent to $[u_2, v_2]$ if $(u_1, u_2) \in E(G_n)$ and $v_1 = v_2$ or if $(v_1, v_2) \in E(H_m)$ and $u_1 = u_2$. Jianxiu [58] reported the following upper bounds for graphs resulting from the Cartesian product of two graphs with path P_n, cycle C_n and complete K_n topologies.

Cartesian product of paths $P_m \times P_n$:

$$\mathsf{CBS}(P_m \times P_n) \leq m(n-1) + n^2(m-1), \quad m \geq n\,. \tag{2.13}$$

Cartesian product of cycles $C_m \times C_n$:

$$\mathsf{CBS}(C_m \times C_n) \leq m(n^2 + 2n - 2), \quad m \geq n \geq 3\,. \tag{2.14}$$

Cartesian product of complete graphs $K_m \times K_n$:

$$\mathsf{CBS}(K_m \times K_n) \leq \frac{1}{6}mn\Big(n^2 + 3n\Big\lfloor\frac{m}{2}\Big\rfloor\Big\lceil\frac{m}{2}\Big\rceil - 1\Big), \quad m \geq n\,. \tag{2.15}$$

Cartesian product of a path and a complete graph $P_m \times K_n$:

$$\mathsf{CBS}(P_m \times K_n) \leq \frac{1}{2}m^2 n\Big\lfloor\frac{n}{2}\Big\rfloor\Big\lceil\frac{n}{2}\Big\rceil + n(m-1)\,. \tag{2.16}$$

Cartesian product of a path and a cycle $P_m \times C_n$:

$$\mathsf{CBS}(P_m \times C_n) \leq n(m^2 + m - 1)\,. \tag{2.17}$$

Cartesian product of a cycle and a complete graph $C_m \times K_n$:

$$\text{CBS}(C_m \times K_n) \leq n\Big(\frac{1}{2}m^2\Big\lfloor\frac{n}{2}\Big\rfloor\Big\lceil\frac{n}{2}\Big\rceil + 2m - 2\Big)\,. \tag{2.18}$$

2.3.2 Algorithms for the CBSP

There are relatively few algorithms proposed to solve the CBSP in the general case. These proposals include a combination of general and reduced variable neighborhood search (RVNS) reported by Satsangi et al. [116, 117], a constructive heuristic based on following the guest graph structure by Hamon et al. [46, 48], as well as MA and BVNS by the author of this document [85]. The most relevant aspects of these proposals are summarized in this section.

2.3.2.1 General Variable Neighborhood Search

This approach, by Satsangi et al. [117], employs RVNS as a method to improve the initial solution, previous to the application of the GVNS routine. The initial solution is the identity permutation. RVNS improves the initial solution along an established number of iterations, using a set of six neighborhood operators. Each iteration starts by getting a new solution by means of the first neighborhood. The new solution replaces the current one if its CBS value is lower. The same steps take place with the remaining neighborhoods.

GVNS starts by getting a new solution via perturbation of the best solution obtained in the RVNS phase. The perturbation operators are the same as the operators from the RVNS phase. Next, local search is applied, using two different neighborhoods. The resulting solution replaces the current one, if it is better. Otherwise, the same process takes place using the next perturbation operator. An iteration concludes once all perturbation and neighborhood operators have been used. The algorithm executes a fixed number of iterations.

Satsangi et al. [117] reported the first benchmark, which included graphs with unknown optimal values up to $n \leq 199$ vertices and $e \leq 1,342$ edges, and graphs of order $n \leq 200$ and size $e \leq 1,000$ with known optimal values. For the former group, GVNS provided the first best-know solutions, and for the later one, it was able to produce optimal solutions. Satsangi et al. also proved that for many graphs exist solutions with a lower cost than the suggested by the theoretical upper bounds. However, the performance of GVNS has been proved poor and time consuming when compared to more recent approaches.

2.3.2.2 MACH

MACH is a constructive heuristic proposed by Hamon et al. [48]. It has two phases, the first one is the partitioning of the graph in disjoint paths; the second phase is the labeling and recombination of the paths to construct a full solution.

The disjoint paths are constructed by a depth-first search (DFS) algorithm guided by a vertex similarity criterion. The DFS starts in the non-visited vertex with the lowest degree and continues to the non-visited adjacent vertex with higher similarity. The similarity among adjacent vertices is assessed by the Jaccard index [57]. The construction of a path concludes when there is no more non-visited adjacent vertices, and the process continues until all vertices are included in a path. MACH constructs a solution by labeling and merging the disjoint paths. First, the paths are sorted according to their length in a decreasing order. The longest path is inserted in an empty list, becoming a partial solution. Next, MACH extends the partial solution by finding the best position in the list to insert the second longest path. The best insertion position is the one that produces the lowest cost increment. The same process is repeated until all paths have been inserted, so the solution is completed.

In the experimental results reported by Hamon et al., MACH had a better performance than GVNS, reaching better results for graphs with known optimal values up to $n \leq 448$ vertices and $e \leq 50,176$ edges; and graphs with unknown optimal values up to $n \leq 1,474$ vertices and $e \leq 1,923$ edges [48]. MACH is a purely constructive algorithm with low execution time for most topologies with known optimal values. But some other graphs have topologies not suitable for this algorithm, such as random graphs and graphs derived from sparse matrices. In those cases, the execution time of MACH can increase considerably: up to $13,803$ seconds for a graph of order $n = 5,300$ [85].

2.3.2.3 Memetic Algorithm

In 2016, the author of this document presented a mememtic algorithim (MA) for the CBSP [85]. This MA employed binary tournament selection and the order-based crossover operator for producing a feasible permutation encoded offspring. A selective mutation operator was implemented. It exchanges each gen with a random one and keeps only the mutations that produce fitness improvements. To balance this, an inversion operator was also used on some mutated individuals. The survival strategy implemented was (μ, λ).

Experiments with MA extended the benchmark up to 412 instances, with graphs with known optimal value up to $n \leq 1,000$ vertices and $e \leq 250,000$ edges, and graphs with unknown optimal value up to $n \leq 5,300$ vertices and $e \leq 8,271$ nodes. The topologies included were random graphs, Harwell-Boeing sparse matrices, Cartesian products, paths, cycles, wheels, powers of graphs and complete bipartite graphs. For all the experiments the execution time was limited to 300 seconds. MA reached equal cost solutions when compared to MACH for 165 instances and better cost for 231 instances, with only 16 instances on which MACH had better results than MA.

2.3.2.4 Basic Variable Neighborhood Search

The author of this document also presented a basic variable neighborhood search (BVNS) for the CBSP. BVNS implemented an iterative process of variable size perturbation and first-improvement local search with two neighborhood alternatives. When the local search fails to find a better solution, the neighborhood changes and the perturbation size for the next iteration is increased. The perturbation size is reset to one when a new best-found solution is reached.

In the experiments with BVNS, three strategies for the initial solutions were tested: random solutions [34], a greedy algorithm, and MACH. The greedy initialization version of BVNS (BVNS+Greedy) offered the best balance between solution quality and execution time. BVNS+Greedy was tested under the same methodology than MA [85], producing results equal

Table 2.1: Algorithms for the CBSP.

Algorithm			Known optimal value			Unknown optimal value		
			G	n	e	G	n	e
GVNS	Satsangi *et al.* [117]	Poor solution quality, time demanding	C_{200} and C_{100}^{10}	200	1,000	*will_199*	199	1,342
MACH	Hamon *et al.* [48]	Topology dependent and time demanding in some cases	$K_{224,224}$	448	50,176	*bcspwr06*	1,474	1,923
MA	Narváez-Terán [85]	Premature convergence	$K_{500,500}$	1,000	250,000	*bcspwr10*	5,300	8,271
BVNS	Narváez-Terán [85]	Local optima stagnation	$K_{500,500}$	1,000	250,000	*bcspwr10*	5,300	8,271

to MACH for 157 instances, better ones for 238 of them and worse ones for 17.

Table 2.1 summarizes the reported algorithms for the CBSP, their performance issues, as well as the maximal order and size of the graphs in the benchmarks they employed.

2.4 Optimization algorithms and techniques

There is a wide variety of techniques designed to tackle optimization problems. This section presents a general overview of the optimization techniques divided in broad groups, specifically the ones involved with this work.

2.4.1 Exact algorithms and approximation algorithms

Exact algorithms are methods to produce optimal solutions for problems. Some exact algorithms, such as branch and bound (BB) and constraint programming (CP), focus on reducing the size of search space by defining rules to discard certain subregions where the optimum is assumed not to be located [128]. Even with this type of strategies in place, for most optimization problems, the size of the search space causes that the application of exact algorithms would be limited to relatively small size instances [128]. While the scope for exact algorithms is limited, they can be very efficient within it.

The approximated methods are algorithms that produce suboptimal *good* solutions. They are often classified in approximation algorithms and heuristics, where the later also comprehends metaheuristics and hyperheuristics. The approximation algorithms are polynomial time methods that produce bounded nearly optimal solutions [5]. The performance ratio of an approximation algorithm is a constant factor expression that defines how much worse than the optimum are the solutions produced by it [22], in other words, how *proximal* to the optimum those solutions are in the worst or the average case scenarios.

For other approximated methods, such as heuristics and metaheuristics, it can be complicated to calculate performance ratios, specially in the average case scenario, not only because of the difficulty to predict the behavior of sophisticated stochastic operators, but also because of the often wide performance variations among problem instances [5]. There can be significant differences between the worst case scenario and the average one, since these estimations depend on assumptions about the distribution of instances, which may not be realistic [128]. Moreover, an algorithm having a better theoretical performance ratio than other does not necessarily mean that in practice its solutions would also be better for all the cases. Because of this, the claims of good performance for approximation methods are typically supported by their actual average results among a diverse enough set of instances and a big enough number executions, employing performance metrics such as solution quality, execution time and statistical significance [5, 19]. These types of analysis are essential for the comparison and evaluation of approximation algorithms.

2.4.2 Heuristics and metaheuristics

Heuristics are problem specific exploration methods based on educated guesses for exploiting some aspect of the problem. In contrast, metaheuristics are not defined by any specific problem, they are instead algorithmic frameworks applicable to a wide variety of problems. In the literature [16, 19], metaheuristics are defined by:

- Absence of any hypothesis on mathematical properties of the objective function, besides it being computable for all solutions in the search space.
- Performance dependence on parameters, which values are often set empirically. These parameters control the balance between exploration of new areas of the search space and the exploitation of known good ones.
- An initial solution is required. It can be obtained by a trivial method, such as a random solution initialization, or by a more elaborated heuristic.
- Execution ends by meeting predefined stop criteria, such as: number of iterations, execution time, or number of evaluations of the objective function.
- Generally, they are easy to implement and often they can be turned into parallel algorithms achieving a significant speedup.

2.4.3 Single-solution and population-based metaheuristics

Single-solution metaheuristics focus on the improvement of one solution by employing neighborhoods and local search [128]. The neighborhood of a solution is the set of solutions than can be obtained by applying once an operation that slightly modifies the current solution to become a different one [68]. For example, for a binary encoded solution, the one-flip neighborhood is the set of solutions obtained by flipping one bit. A distance can be calculated between pairs of solutions, in terms of how many times the neighborhood operator has to be applied to one solution to obtain the other [118]. Following the previous example, two binary solutions that differ in exactly three bits are at three steps of distance from each other. In local search, the neighborhood operator is used to look around the immediate surroundings of a solution in the search space, and based on that, the algorithm takes a step towards a neighboring solution better than the current one [40, 54]. There are different strategies to choose among neighboring solutions, for example, to choose a random one, to take a step towards the first better neighboring solution to be found (first-improvement); or to compute the whole neighborhood and choose the best solution in it (best-improvement). Single-solution metaheuristics can be seen as customized, more intelligent extensions of local search, and are also referred as trajectory-based methods [16]. Some examples of single-solution metaheuristics are Iterated local search (ILS), variable neighborhood search (VNS), simulated annealing (SA), and tabu search (TS) [16].

Population-based metaheuristics employ a group of solutions to search simultaneously on multiple regions of the search space [128]. In this sense, population-based metaheuristics are multi-trajectory search algorithms. Population-based metaheuristics use the quality of their solutions as feedback in the creation of new solutions via some sort of recombination, and they often incorporate nature inspired behaviors [15]. Some examples of population-based metaheuristics inspired by nature are evolutionary algorithms, like genetic algorithms (GA) [137] and differential evolution (DE) [99], and swarm intelligence, such as ant colony optimization [29] and particle swarm optimization (PSO) [140]. In population-based metaheuristics, the search is a group effort, guided heavily by solutions in good regions of the search space. For example, in a GA the genes of fitter individuals are more likely to proliferate, and in PSO, particles in good regions are likely to pull the swarm towards their direction.

There are hybrid metaheuristics as well [14], some of them employ typical single-solution approaches within population-based metaheuristics. For example, the memetic algorithms (MA) combining the use of evolutionary operators and local search [82], and some of them hybridize metaheuristics and exact algorithms [61].

2.4.4 Fitness landscape

The concept of the fitness landscape was first devised by Wright [138]. It became an approach to analyze the behavior of evolutionary algorithms [101] in terms of populations moving through a landscape, with valleys and hills representing the variations on fitness values. The fitness landscape is also a very useful tool to better understand problems and their challenging factors.

The fitness landscape of a problem is defined as a triplet [62] consisting of:

- A set of solutions under a certain encoding.
- A fitness function to evaluate the fitness values of solutions.
- A neighborhood structure that can provide a distance measure between pairs of solutions. Ideally, this should capture the number of times that an operator is applied to make a transition from one solution to the other one.

Assuming minimization, the valleys are the regions of better fitness, with the global optimum located in the lowest point of the landscape. A solution is a local optimum if it is the fittest solution among its neighborhood. Of course, to make a full computation of the fitness landscape of a problem would be equivalent to knowing all the possible solutions and their distances to each other, and therefore having solving the problem, which is impractical. To keep the fitness landscape analysis practicable, it is typical to base it on the assumption that the fitness landscape looks generally the same everywhere, i.e., it is isotropic [97], and therefore, studying selected samples of it is enough to get a generalized idea of its structure.

Fitness landscapes that contain a high number of local optima are generally thought to be more difficult [1, 50], but this is not the only criterion to consider. Other important related aspects are ruggedness and neutrality [93, 100]. Ruggedness is associated with little correlation between neighboring solutions, while neutrality presents as large plateaus of solutions with the same fitness. Both are indicatives of difficulty because they mean that there is few useful information to decide a good search direction.

The fitness landscape analysis has proven to be a useful tool for a better understanding of problems, search dynamics and algorithm performance [78, 125]. Numerous techniques have been devised in order to decompose landscapes into measurable features that describe the intrinsic difficulties of a problem [76, 77, 96, 100**?**]. Some of these techniques include measures based on correlation to determine how much the fitness of a solution may depend from solutions around it [126, 135], the relationship between the distance to the optimum and fitness [60], analyzing the the distribution of fitness values [113], the estimating the size of structures like plateaus and basins [51], examining operator's ability to improve solutions [123, 131, 132, 134], visualization techniques related to search dynamics [89, 90], etc.

2.4.5 Hyperheuristics and adaptive operator selection

Often, the success of metaheuristics depends on the combination of operators and their ability to cope with the intrinsic difficulties of a problem. Choosing the operators that better suit a given problem (or a set of instances of a problem) can be a defiant and time consuming task [37]. The adaptive operator selection (AOS) is the problem of choosing suitable operators during execution time [33, 37]. An approach to the AOS are hyperheuristic frameworks based on reward systems [44]. Hyperheuristics are methods to decide which one from a set of low level heuristics (operators or entire algorithms) to use at a given time during execution [17]. The decision making process within hyperheuristics is an empirical one. It is based on rewards directly influenced by the performance of low level heuristics in past iterations. When dealing with optimization problems, which difficulty may vary among instances, hyperheuristics frameworks can help to build more robust algorithms able to self adapt their operators to unknown scenarios. Some hyperheuristic frameworks are the multi-armed bandit (MAB) [3] and the island model [11] for evolutionary algorithms.

2.4.6 Conclusions

The chapter introduced key fitness landscape concepts, such as ruggedness and neutrality, related with challenges associated to the traditional evaluation scheme of the CBSP; as well as an overview of optimization techniques and tools to be implemented in combination with problem specific information in the scope of the proposal described in Section 1.3.

After having considered the current related work for the CBSP presented in this chapter, it is notorious that there are opportunity areas in both the analysis of the problem and the development of better, more efficient algorithms to solve it. The remaining chapters present the aspects of the proposed approach to the CBSP related to the design of an alternative evaluation scheme with improved capability to guide the search, the design of algorithms for the CBSP, the selection of their operators, the analysis of the fitness landscape of the problem, and the implementation of tools such as adaptive paradigms and parallelism to further improve the performance of the proposed algorithms for solving the CBSP.

3 An alternative evaluation function for the CBSP

3.1 Introduction

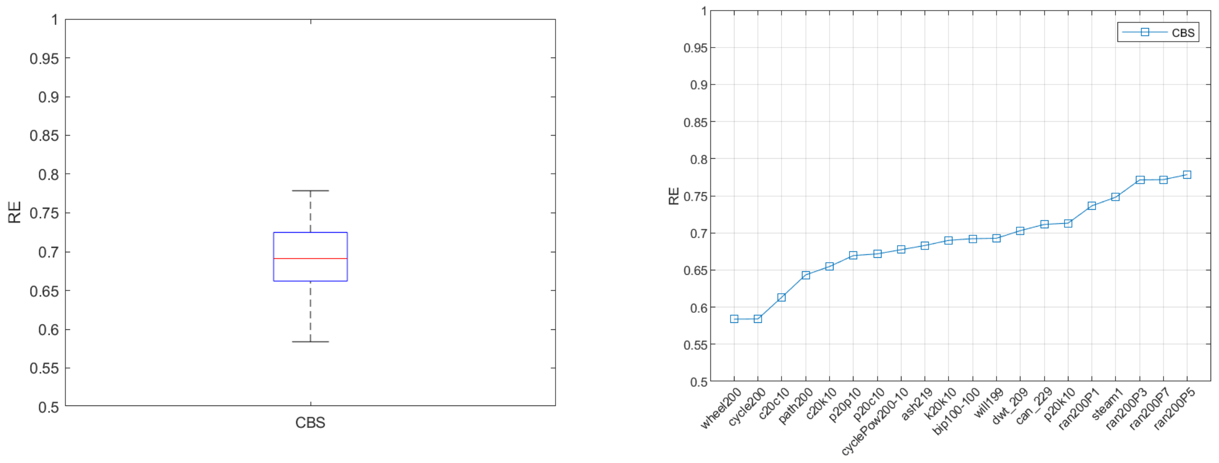
The fitness function is highly associated with the main challenges of solving an optimization problem, whether it is by exact or heuristic based methods. The function has not only the role of assessing the cost of a given potential solution; its nature is relevant for establishing cost bounds, to predict the value of the optimal cost and it impacts directly on the structure of the fitness landscape. Search based approaches rely on the fitness function to make crucial decisions regarding further search directions that determine whether or not the algorithm will be able to reach certain regions of the search space, will be able to escape from local optima, and overall, how successful it will be for solving the problem. It has been demonstrated that some fitness functions impact negatively in those aspects, and that solving methods can have their performance significantly improved by employing more informative fitness functions that still keep consistency regarding the objective of the problem [38, 39].

When looking for a fitness function that is different from the objective function of the problem, it arises the question of what makes a fitness function more or less helpful guiding the search and under which circumstances. To begin with, a fitness function should make it easy to determine if a solution is better than another one. Focusing on this particular aspect, in the ideal case, each solution should get assigned a different cost. Extrapolating to the fitness landscape, which also involves a notion of neighbourhood under a particular solution encoding, the fitness of solutions should become better as they become closer to the global optimum, and it would also imply the absence of multimodality. However, in practice, that is rarely the case for problems that are not already easily solved, and it is indeed not the case for the CBSP.

The CBS function is a sum of e cyclic distances, which can have values in the range 1 to $\lfloor n/2 \rfloor$. There can be more than one combination of cyclic distances adding up to the same total value. The number of fitness equivalence classes that the CBS function can create is relatively small when compared to the number of possible embeddings, specifically, $(e\lfloor n/2 \rfloor \lceil n/2 \rceil)/(n-1) - (n-1)$ different cost values to assign to solutions in a search space of size $(n-1)!/2$.

In order to establish how the poor discrimination capabilities of the CBS function are, this was assessed for a set of 20 representative instances [109]. The instances belong to the different topologies described in Appendix A.1. This set is also employed in the rest of experiments along this chapter. It includes: six Cartesian products of graphs with known upper bounds, five standard graphs with known optimal solution values, five graphs from the Harwell-Boeing Sparse Matrix Collection, and four Erdös-Rényi random graphs. The graphs in the set have $199 \leq n240$ vertices and $199 \leq e \leq 651128$ edges. In the case of Harwell-Boeing instances, graphs of order as close as possible to 200 were chosen from the collection. The rest of the graphs have exactly 200 vertices.

A sample of $c = 100,000$ random embeddings for each graph in the set was taken. The embeddings were assigned descending ranks in function of their cost, with same cost solutions sharing the same rank. The relative entropy $RE(D)$ metric in Equation 3.1 was measured for the distribution of ranks [109]. The maximum value of the relative entropy metric is $RE(D) = 1$, representing the



(a) Average relative entropy among instances.

(b) Relative entropy for each instance.

Figure 3.1: Potential of discrimination for the CBS function, measured by the relative entropy of 100,000 embeddings for 20 representative instances.

ideal case where for each rank j, there is $D_j = 1$ solution. The maximum relative entropy would correspond to every embedding in the sample belonging to a different fitness equivalence class.

$$\text{RE}(D) = \frac{\sum_{j=1}^{c} \frac{D_j}{c} \log\left(\frac{D_j}{c}\right)}{\log\left(\frac{1}{c}\right)}, \tag{3.1}$$

As expected, the potential of discrimination of the CBS function is not the ideal, with the relative entropy measure ranging from 0.58 to 0.77.

In the scenario of a neighbourhood based search, it is possible that the neighborhood of a solution contains one or more improving candidates of the same fitness. Under the CBS function, which of those candidates is chosen is a random decision instead of one based on which one is more likely to lead towards improvements, so the search would be blindly moving across a fitness plateau. While fitness plateaus are composed of neighboring solutions within the same fitness equivalence class, those solutions are not equivalent search-wise, in the sense that each one of them could lead towards different search directions, some of them better, some others worse. Also, in a plateau the search could spend valuable iterations cycling among its solutions.

So far, all previous solution approaches have employed the problem's objective function as their fitness function [47, 117]. It is likely that the poor discrimination of the CBS function has played a role on hindering their performance by propitiating the type of neutrality inducing blind search decisions discussed before and thus guiding those solution methods to local optima where they become stagnant.

The goal of designing a new alternative fitness function for the problem is to introduce distinctions among solutions belonging to the same fitness equivalence class under the CBS function. Those distinctions should be based on a notion of which features make a solution more likely to lead to further improvements than other solutions in their same fitness equivalence class. For example, consider a scenario where a neutral fitness neighbour of a visited local optima happens to be in the attraction basin of another local optima of better quality. How could an alternative fitness function asses that? Our assumption is that the distribution of the cyclic distances contains useful information towards this task. Based on that, three candidates for a new alternative fitness function for the CBSP were designed.

This chapter describes the research work on designing a new more informative evaluation function that also keeps consistency with the main goal of the CBSP and that it is able to provide improved guidance for search methods through neutral regions of the search space improving their performance.

A comparative study considering the CBS function and the three new evaluation schemes is carried out by following the methodology devised by Garza-Fabre et al. [39]. It includes: a) an investigation of their discrimination potential, b) an analysis concerning the consistency of the three new evaluation functions with regard to the primary objective of the CBSP, and c) an assessment of the practical usefulness of the four evaluation approaches when used within two distinct search algorithms.

The remaining sections of the chapter are organized as follows. Section 3.2 explains the methodology used for function comparisons. Section 3.3 introduces three new evaluation schemes for the CBSP and it presents the analysis of their complexity. Section 3.4 discusses the studied evaluation schemes, addressing their capacity for discrimination and the CBS-compatibility. In Section 3.5 the ability of the proposed functions to guide search algorithms and their influence in the convergence process is compared to that of the CBS function by employing Steepest Descent and Iterated Local Search. A comparison of performance between two state-of-the-art methods and the ILS implementation, equipped with the best alternative evaluation scheme is presented in Section 3.5.4. Finally, Section 3.6 summarises this chapter.

3.2 A framework for comparing fitness functions

The idea of exploring the use of different fitness functions for a given problem has been employed numerous times outside of this work. It has been successfully applied in problems such as the prediction of protein structure [39], vehicle routing [73], nurse rostering [122], the job-shop problem [83]. Alternative fitness functions have also been proposed for other GEP, such as the bandwidth problem [105] and the minimum linear arrangement [103] (another name sometimes employed for the bandwidth sum problem).

Across those previous works, a generalized framework for studying the effectiveness of alternative fitness functions has taken shape [38]. It focuses on assessing the quality of the new proposed functions under three main criteria: a) that the new function is able to increase the discrimination by inducing more fitness equivalence classes with lower cardinality, b) that it remains consistent with the original objective of the problem and c) that it has demonstrated effectiveness in guiding search algorithms.

In this work, the first criteria was evaluated employing the notion of potential for discrimination, measured as the relative entropy [23] for a set of fitness-ranked embeddings [109]. In order to address the second criteria, the notion of CBS-compatibility [109] was devised, in a similar way to the HP-compatibility formulated for analyzing alternative fitness functions for the problem of protein structure prediction under the HP model [38].

Definition 1 *An alternative evaluation function $f : \Phi \to \mathbb{R}$ is said to be CBS-compatible if and only if $f(\varphi) < f(\varphi') \Rightarrow$ CBS$(\varphi) \leq$ CBS(φ') for every pair of solutions $\varphi, \varphi' \in \Phi$. Otherwise, if at least one pair of embeddings φ, φ' exists such that CBS$(\varphi) <$ CBS(φ') but $f(\varphi) > f(\varphi')$, then function f is not CBS-compatible.*

The notion of compatibility with respect to the original objective function of the problem is crucial for ensuring that the goal of the optimization process does not change. It is desirable that an alternative fitness function introduces changes in fitness relationships among solutions belonging to the same fitness class under the original function. However, it is also desirable that those relationships are not disrupted among the rest of the solutions.

Finally, for the third criteria, a thoughtful comparison on the performance of two basic search algorithms, steepest descent (SD) and iterated local search (ILS) was performed, employing each of the new evaluation functions and the original one.

3.3 Alternative evaluation scheme candidates

It was observed that one feature that distinguish CBS embeddings of the same cost is their distribution of cyclic distances. Under the problem's objective function, each cyclic distance contributes to the CBS value independently. For example, a solution where more than half the cost is due to one particularly big cyclic distance, while the rest of cyclic distances are comparatively small, can be evaluated the same than a solution where the all the cyclic distances are around the same magnitude. The core idea for the design of the fitness function alternatives is that the exemplified cases should not be evaluated as equivalent. Such distinction is intended to be relevant and helpful regarding the search space exploration, and not as a change in the problem definition.

In order to design evaluation schemes that weight-in the cost contribution of each cyclic distance, a rewritten version of the CBS function in terms of how many distances of each possible value are present given a particular solution is introduced. This is expressed in Equation 3.2, where d_k accounts for the number of cyclic distances of cost k. Under this definition, a cyclic distance has an implicit weight equal to its value k.

$$\text{CBS}(G, \varphi) = \sum_{k=1}^{\lfloor n/2 \rfloor} k \cdot d_k \ , \tag{3.2}$$

The three alternative fitness functions have a similar structure than Equation 3.2, each of them with its particular way of assigning cyclic distance weights in function of k. Figure 3.2 shows how the weights for the alternative functions behave with respect to the cost k of the cyclic distances, along with the implicit weights of the CBS function. The weight assignations were designed under distinct approaches to define what makes a solution potentially more desirable regarding its cyclic distance values distribution. For function f_1 and f_2, it was assumed that solutions where most of the cyclic distances have small values are more likely to lead to further improvement.

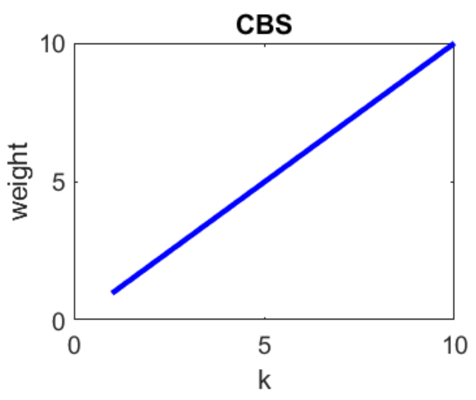
The intended effect for this feature is to direct the search towards solutions where the majority of cyclic distances are small.

Function f_3 was design under the opposite assumption, with the idea that the occurrences of large cyclic distances may be easier to eliminate after applying operations such as swaps, insertions or mutations, thus resulting in solutions of lower cost. Therefore, function f_3 assigns weights that decrease exponentially as the magnitude k of cyclic distances increases, with the intended effect of giving preference to solutions with majority of large cyclic distances. Since the sum of the weights times their occurrences will always be in the range of 0 to 1, function f_3 design was complemented by adding that result to the CBS original value. Thanks to that, function f_3 allows to preserve the CBS evaluation as the integer part of the result, ensuring that the new fitness evaluation will only cause fitness relationship changes among solutions that are equal under the original CBS function.

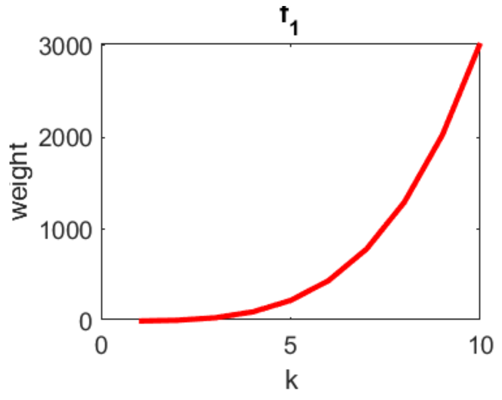
$$f_1(G, \varphi) = \sum_{k=1}^{\lfloor n/2 \rfloor} \left(\sum_{i=1}^{k} i^3 \right) \cdot d_k \ , \tag{3.3}$$

$$f_2(G,\varphi) = \sum_{k=1}^{\lfloor n/2 \rfloor} n^{(k+1)} \cdot d_k \ , \tag{3.4}$$

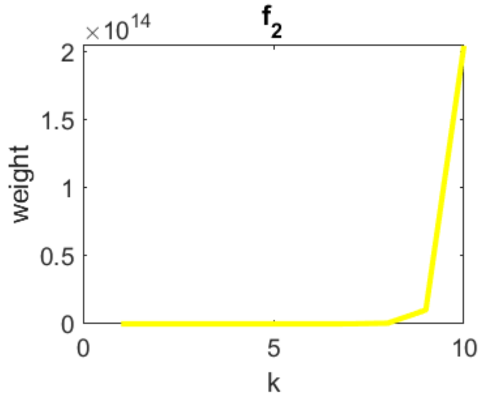
$$f_3(G,\varphi) = \text{CBS}(G,\varphi) + \sum_{k=1}^{\lfloor n/2 \rfloor} \left(\frac{1}{n2^k}\right) \cdot d_k \ . \tag{3.5}$$



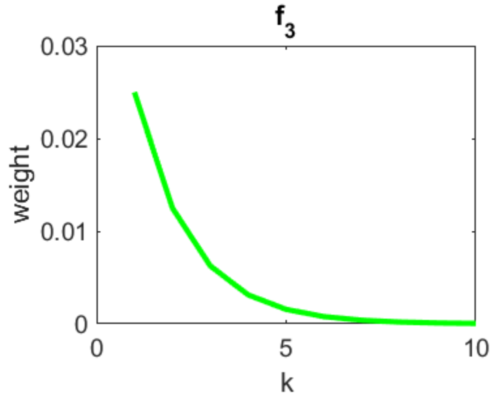
(a) CBS function



(b) f_1



(c) f_2



(d) f_3

Figure 3.2: Weights assignations in function of cyclic distance values ($1 \leq k \leq \lceil n/2 \rceil$) for the CBS function and the three alternative evaluation functions, for $n = 20$.

3.3.1 Alternative fitness functions complexity

The CBS function, computed as the sum of the cyclic distances for all edges of G has a complexity of $\mathcal{O}(e)$. Since $e \leq (n(n-1))/2$, $\mathcal{O}(e) \approx \mathcal{O}((n^2 - n)/2)$.

For an efficient evaluation of the alternative functions, the weights assignations can be precalculated, since they are invariable with respect to the possible values k. As previously mentioned, $k \in 1, ..., n/2$, so the complexity for weight computation is $\mathcal{O}(n)$. The sum of weights for particular embeddings is computed in e steps, so the complexity for the alternative functions is $\mathcal{O}(e + n) \approx \mathcal{O}((n^2 - n)/2 + n)$.

All the alternative functions, as well as the CBS function, can be implemented to efficiently update the cost of potential solutions after operations that only affect part of them, such as swaps, insertions or mutations. To do this, it is enough to recalculate values for at most $|\mathcal{A}(u_1)|+, ..., +|\mathcal{A}(u_x)|$ edges involving nodes affected by the operation, where x is the number of nodes affected.

3.4 Comparing alternative evaluation schemes

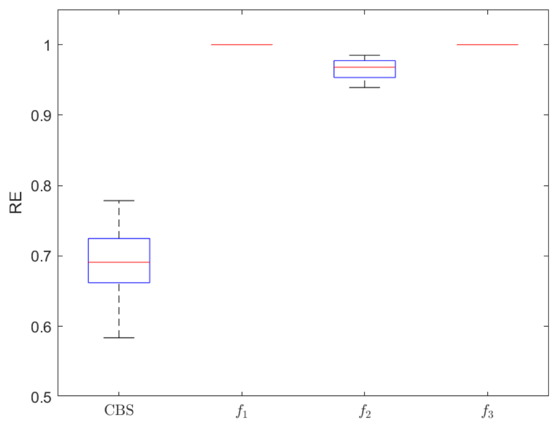
In order to determine the success of the alternative evaluation schemes the evaluation methodology previously introduced in Section 3.2 was applied.

3.4.1 Potential of discrimination

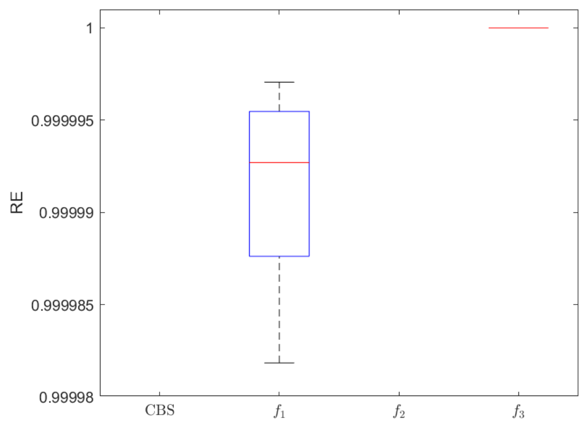
The design goal of the alternative functions was to increase the potential of discrimination with respect to the CBS function. In order to evaluate the achievement of this goal, the potential of discrimination was measured as the relative entropy (RE) of the distribution of ranks for 100,000 embeddings for each instance [109].

Figure 3.3 presents the results for the potential of discrimination assessment. All the three new evaluation schemes have better relative entropy values than the CBS function, as well as less dispersion across varying topologies. This last aspect is better illustrated by Figure 3.4, showing the relative entropy values for each instance, where function f_3 behaves perfectly, with RE values equal to one in all cases, and f_1 is a close competitor with RE values superior to 0.9997.

The relative entropy values for functions f_1 and f_2 behave roughly similarly to the CBS function



(a) Overall statistics for the 20 selected instances.

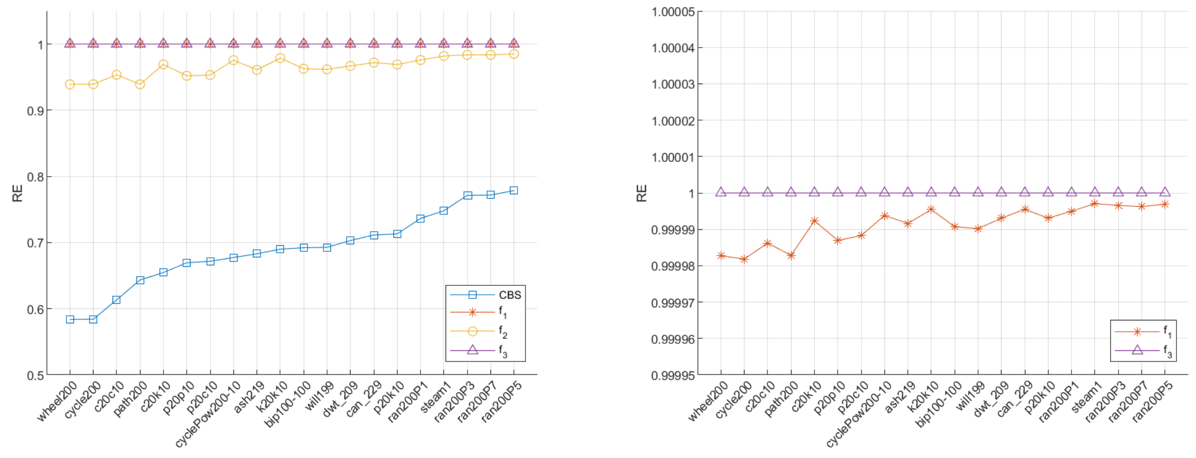


(b) Zoom-in of Figure 3.3(a), with focus on results for f_1 and f_3.

Figure 3.3: The three proposed alternative fitness functions exhibit increased relative entropy (RE) values when compared with the CBS function, thus offering a better potential for discrimination.

regarding certain graphs. For example, the higher RE values correspond to the Erdos-Renyi random graphs and HB graphs, and the worst to Cartesian products, cycle and wheel topologies. That suggest that the occurrence of more high cardinality fitness classes is caused by the existence of size n cycle subgraphs inside of the host graph in topologies such as the ones with lowest RE values: wheel, cycle and the Cartesian product of cycles. Meanwhile, in this experiment function f_3 was the only one to produce perfect rank distributions (with RE=1), and to remain unaffected by graph topology.

Even though this experiment is not specifically measuring fitness landscape neutrality, it is possible to infer from it that the CBS function induces more fitness neutrality in comparison with the other three. Notice, for example, that only an average of 68% of the sample had unique fitness values. If the remaining solutions will cluster into neutral fitness plateaus would also depend on the solution encoding and the neighborhood definition, but the fitness function alone can offer useful insights. Contrasting with the CBS function, the alternative functions assure that the likeness of neutrality is reduced in varying degrees by each one of them. Remarkably, under function f_3 the sampled set of 100,000 solutions per instance contains no fitness equivalent solutions, so it is possible to conclude



(a) RE values for each instance.

(b) Zoom-in of Figure 3.4(a), focused on RE values for f_1 and f_3.

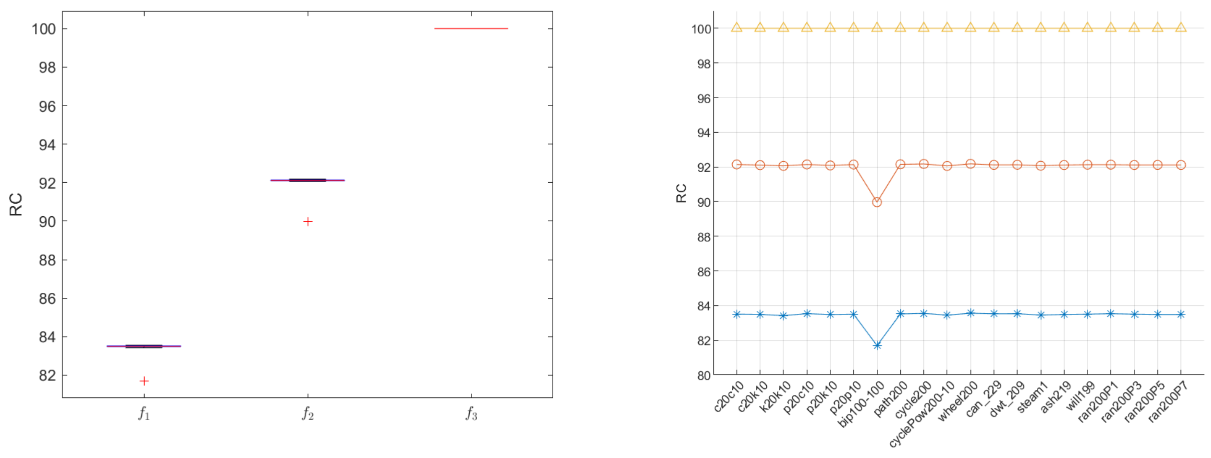
Figure 3.4: Relative entropy (RE) for each analyzed evaluation function. Each point represents the average of 50 independent executions over one tested instance.

that, at least for this sample, there are not any plateaus.

3.4.2 CBS-compatibility

Consider a set of solutions and the pair-wise fitness relationships among them, which define if one is worse, better or equal than the other. The CBS-compatibility is expressed as a percentage, measuring the proportion of pair-wise unequal fitness relationships, induced by the CBS function, that are preserved by the each of the alternative functions. If the result of this metric is lower than 100% it indicates that there are some solutions that are considered better than their respective pairs by the CBS function, but worse by the alternative function, and vice-versa. Low percentages of CBS-compatibility indicate that an alternative function is unlikely to keep consistency with the original problem goal.

The CBS-compatibility was measured for the pair-wise unequal fitness relationships of a set of 100,000 random solutions sampled for each of the 20 instances in the set [109]. This experiment was repeated 50 times and its average results appear in Figure 3.5.



(a) Overall RC value for the 20 selected graphs instances. (b) Average results by instance

Figure 3.5: Relative CBS-compatibility (RC) values for the alternative functions computed for a sample of 100,000 solutions.

The worst compatibility evaluation corresponds to function f_1 with 84% on average, followed by f_2 with 92%. As expected, function f_3 exhibits 100% compatibility, since it was designed to add a smaller than 1 value to the CBS evaluation. That particular feature also makes the CBS-compatibility of f_3 invariable across topologies.

From this experiment it was inferred that the assignation of exponential growing weights based on cyclic distance values used in f_2 and f_3 resulted in smaller deviations from the relative quality of embeddings. It also demonstrates that design of f_3, incorporating the CBS original value as its integer part, makes its perfectly compatible with the problem objective.

3.5 Search performance

In the experiments for evaluating the potential of discrimination and CBS-compatibility of the alternative fitness functions, f_3 got perfect scores in both cases. Regarding the two first criteria, offering more discrimination and being consistent with the problem definition, function f_3 is the leading candidate. The final criteria concerns practical usability as the ability to guide search algorithms. It was expected that, if so far f_3 is better discriminating potential solutions and CBS-compatible, it should observed an improvement in search performance when compared to the CBS function and the other two alternative functions.

The search performance tests for the functions were carried on by implementing two basic search algorithms: a steepest descent (SD) and an iterated local search (ILS) [109]. The steepest descent is the equivalent of a hill-climber algorithm applied to a minimization problem. It consists in exploring the neighborhood of the current solution, replacing that solution by an improving neighboring solution and repeating the process until there is no improving neighbors. At that point, the search has found a local optimum and it stops. The iterated local search instead performs a perturbation of the local optimum and then a new local search iteration. The perturbation has the purpose of producing a new solution relatively close to the previously found local optimum, but hopefully in the attraction basin of another one.

The SD and ILS algorithms are simple enough to not have their behaviour influenced by a large number of parameters and their implementations are quite straightforward. Therefore, their simplicity allowed us to focus the analysis on the impact of changing the evaluation scheme.

For the implementation of both algorithms, best-improvement and the 2-swap neighborhood were employed, for embeddings encoded as permutations [109]. The steepest descent was executed 50 independent times per instance for each function. For a fair comparison, every run starts from fixed initial random solutions. For the ILS experiments, the tested values for perturbation strength were $PS = \{5, 10, 15\}$, and a maximum running time of $MT = \{300, 600, 900\}$ seconds as stop criterion. The assessments of O-RMSE and statistical significance followed the methodology that was previously introduced in Appendix **??**.

3.5.1 Steepest descent performance

Table 3.1 presents the number of vertices $|V|$ and edges $|E|$ of the each graph and their best-known/optimal* cost B followed by the results of particular algorithms. Those results include the

Average Avg solution cost, its standard deviation Dev and average iterations I, as well as the O-RMSE values. The statistical comparison is shown in Table 3.2, according to the methodology described in Appendix A.3.

The worst performance for the SD agorithm was attained when using the alternative function f_1, as denoted by the O-RMSE metric in Table 3.1 and distribution of RMSE values in Figure 3.6. Function f_1 exhibited a near perfect behaviour in the experiments for assessing the potential of discrimination, where it was able to assign unique fitness values to most of the solutions in the sample. However, it was demonstrated that f_1 is not CBS-compatible, as its RC values were the lowest among the three considered alternatives. As expected, the augmented potential of discrimination is only truly useful if the new evaluation scheme does not affect the original objective. In contrast, the increment in potential of discrimination of function f_2 with respect to the CBS function is not as high as the one of f_1, but f_2 has better compatibility. As result, f_2 was able to lead the SD algorithm towards better solutions than both f_1 and the CBS function. It is therefore not surprising that the SD using function f_3, which has the highest potential of discrimination as well as the highest compatibility, achieved a significantly improved performance compared with its pairs.

The fitness relationship changes among solutions that function f_3 introduces are limited to the scope of solutions that had the same CBS value under the original function. The design of f_3 states that, when solutions have the same CBS, the one with more occurrences of higher cost cyclic distances is picked, as large cyclic distances may be easier to further improve. That unique particularity of function f_3 allows that, while the SD algorithm is still primarily optimizing the CBS represented by the integer part of the fitness, it is now possible to decide among equal CBS solutions by considering their cyclic distance values encapsulated on the strictly smaller than one floating part of their f_3 fitness.

Table 3.1: Performance by instance of the SD algorithm when using the four studied evaluation functions.

				CBS			f_1			f_2			f_3		
Graph	$\|V\|$	$\|E\|$	B	*Avg*	*Dev*	*I*	*Avg*	*Dev*	*I*	*Avg*	*Dev*	*I*	*Avg*	*Dev*	*I*
c20c10	200	400	2360⋆	4993.24	585.13	1561.46	8692.44	683.55	4766.96	6049.16	632.13	4739.20	4197.64	703.10	3803.36
c20k10	200	1100	5300	10976.28	1304.90	1873.26	13787.64	2949.38	4820.22	9525.32	923.80	3378.38	11067.44	1366.13	2607.32
k20k10	200	2800	62300⋆	62436.00	755.59	2280.46	84772.16	862.00	3302.62	62304.92	2.50	3633.68	62300.00	0.00	3455.10
p20c10	200	390	2256	4809.28	600.77	1572.40	8342.26	738.01	4888.98	5585.40	802.18	4873.40	4062.96	551.73	3518.82
p20k10	200	1090	5200	10537.88	1227.61	1899.76	13101.64	3262.07	4727.20	9056.20	804.87	3393.70	10464.00	1104.77	2596.04
p20p10	200	370	2385	4082.90	619.52	1653.46	7315.62	926.58	4827.54	4657.02	730.89	5159.54	3354.56	489.50	3335.00
bip100-100	200	10000	500000	500000.00	0.00	115.02	500000.00	0.00	914.74	500000.00	0.00	870.84	500000.00	0.00	133.94
path200	200	199	199	1817.30	209.11	831.34	1799.12	260.51	6812.78	1366.92	219.50	5961.18	1066.42	175.07	4975.28
cycle200	200	200	200	1888.56	207.06	850.68	1894.00	262.91	6639.96	1445.96	242.10	6201.88	1080.16	200.13	5226.80
cycleP200-10	200	2000	11000⋆	20534.36	4437.49	2851.96	18292.24	7242.48	7333.08	15249.36	6166.86	6184.46	15624.40	5275.83	5543.84
wheel200	200	398	10200	11885.32	206.32	819.16	11854.64	261.79	6801.38	11455.80	254.86	6136.12	11082.36	173.86	5158.52
can_229	229	774	6301⋆	9281.88	1409.97	3260.88	15772.50	2090.70	6498.38	10778.56	1910.59	6577.76	8577.52	1566.76	4946.46
dwt_209	209	767	7119⋆	8964.78	817.21	2171.70	13774.14	1721.82	6805.96	9382.72	1410.75	6062.88	8978.40	681.48	2815.02
steam1	240	1761	24158⋆	30878.12	3037.17	3422.12	43278.28	5591.47	4852.92	31692.08	4180.80	4890.12	30305.04	3396.44	3753.00
ash219	219	431	6705⋆	8269.36	542.69	1953.90	12586.34	472.12	4731.16	9480.28	652.37	5555.60	7848.82	645.95	4256.12
will199	199	660	14116⋆	15722.04	653.69	2239.12	21010.08	606.12	3786.00	17658.28	817.34	4135.80	15522.24	629.90	3361.78
ran200P1	200	1991	71394⋆	72802.78	705.67	2422.18	80034.24	938.28	3039.18	75927.04	654.93	3567.24	72693.96	619.91	3269.74
ran200P3	200	5970	256987⋆	259502.00	1061.70	2770.02	269017.16	1041.76	2812.12	263671.86	1043.50	3453.40	259250.16	923.42	3476.50
ran200P5	200	9955	452486⋆	455109.68	1167.33	2955.38	465122.66	1223.02	2646.84	459212.96	1329.09	3380.70	454989.04	1178.27	3542.62
ran200P7	200	13827	651128⋆	653407.40	1134.01	2816.72	661903.78	1417.15	2274.00	656706.26	1221.45	3135.66	653187.90	837.67	3448.70
	O-RMSE				122.86%			169.67%			105.44%			76.51%	

Table 3.2: Statistical analysis for comparing the performance of the SD algorithm when using the four analyzed evaluation approaches.

	Instance																					
Function	c20c10	c20k10	k20k10	p20c10	p20k10	p20p10	bip100-100	path200	cycle200	cycleP200-10	wheel200	can_229	dwt_209	steam1	ash219	will199	ran200P1	ran200P3	ran200P5	ran200P7	Overall	
CBS / f_1	+	+	+	+	+	+	⋆	⋆	⋆	⋆	⋆	+	+	+	+	+	+	+	+	+	15 +	0 −
CBS / f_2	+	−	−	+	−	+	⋆	−	−	−	−	+	⋆	⋆	+	+	+	+	+	+	10 +	7 −
CBS / f_3	−	⋆	⋆	−	⋆	−	⋆	−	−	−	−	−	⋆	⋆	−	⋆	⋆	⋆	⋆	⋆	0 +	9 −
f_1 / f_2	−	−	−	−	−	−	⋆	−	−	−	−	−	−	−	−	−	−	−	−	−	0 +	19 −
f_1 / f_3	−	−	−	−	−	−	⋆	−	−	⋆	−	−	−	−	−	−	−	−	−	−	0 +	18 −
f_2 / f_3	−	+	−	−	+	−	⋆	−	−	⋆	−	−	⋆	⋆	−	−	−	−	−	−	2 +	14 −

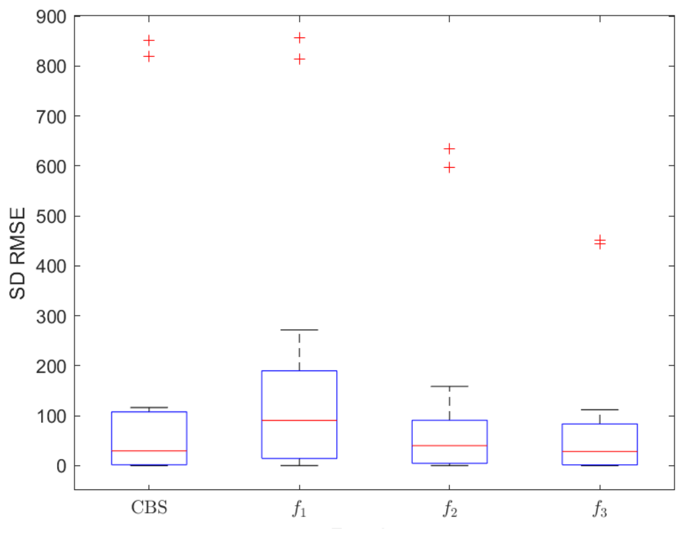


Figure 3.6: RMSE values for the SD algorithm when employing the CBS function and the three alternative functions.

3.5.2 Iterated local search performance

Since the performance of the ILS can be affected by the choice of perturbation strength, for the sake of fairness, it was first tested for three perturbation values per function. In this experiment, the maximum running time was considered as well. The results are summarized by Figure 3.7, showing the O-RMSE values for each of the ILS configurations.

It was observed that the best perturbation strength values for the ILS using each of the functions were $PS = 10$ for the CBS function, $PS = 5$ for f_1, $PS = 10$ for f_2 and $PS = 15$ for f_3. Overall, all ILS configurations benefit from a higher time budget, and in the case of the ILS with f_3, also from a stronger perturbation.

The previously discussed results for the SD are consistent with the observed performance of the ILS for the different functions. The worst O-RMSE values correspond to the ILS using f_1 and its performance is also the most susceptible to perturbation strength increases, possible due to the low

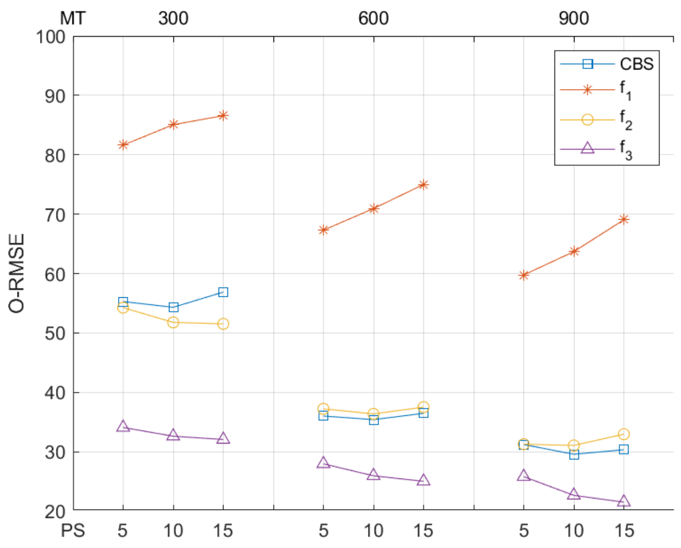
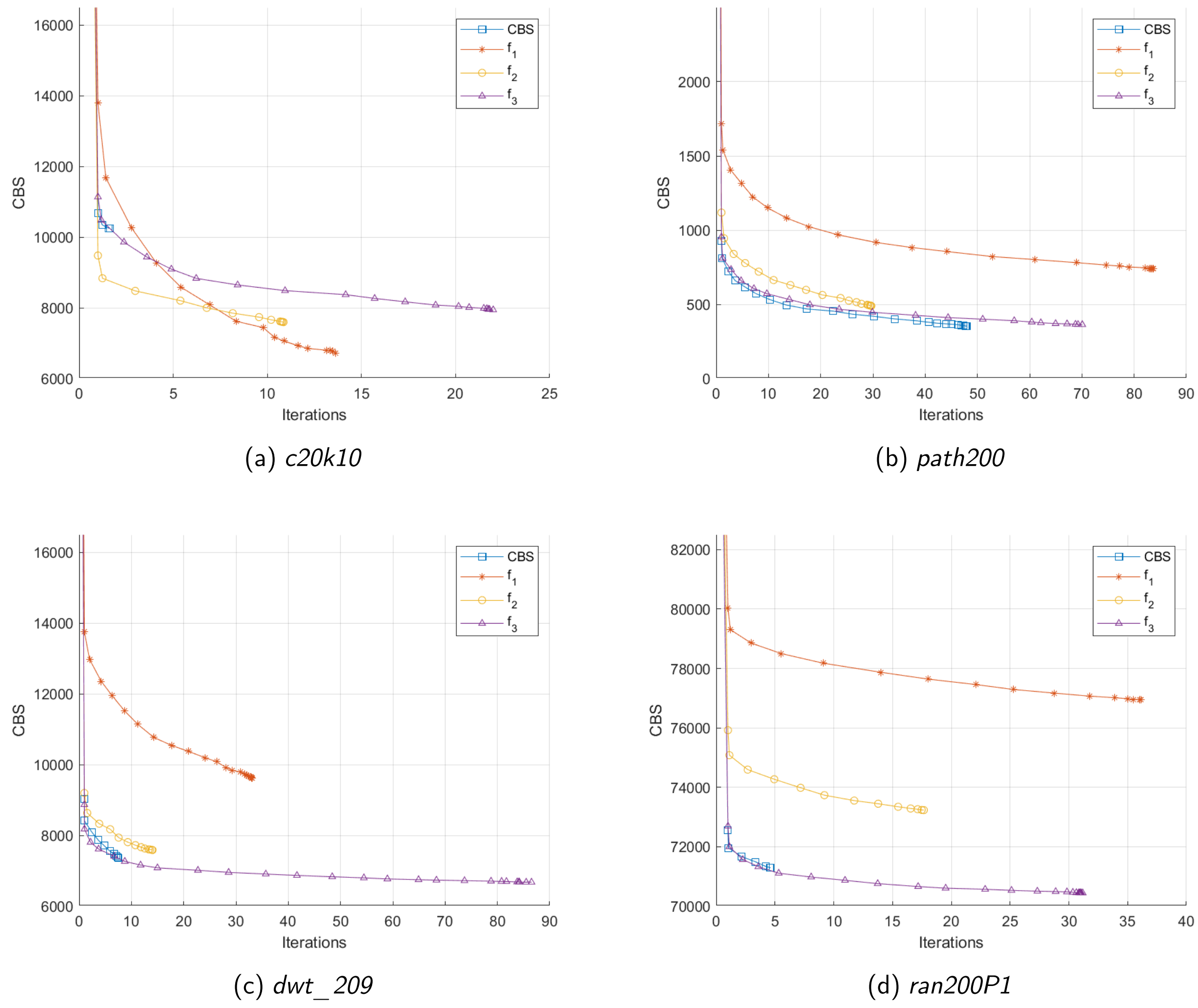


Figure 3.7: O-RMSE values for the ILS employing each of the functions, for perturbation values $PS = \{5, 10, 15\}$ and maximum running time $MS = \{300, 600, 900\}$ seconds.

compatibility of f_1 causing a scenario where relatively close solutions have unrelated CBS values, even if their f_1 values are related. The ILS configurations using function f_2 and the original one had quite similar behaviours across perturbation strength variations, with both attaining close O-RMSE values for $PS = 5$ and the ILS using f_2 being slightly outperforming.

As in the case of the SD, the best performance for the ILS is achieved when using the function f_3. It is notable that this is the only function where the strongest perturbation is preferred. Function f_3 search choices for navigating plateaus are heuristically shaped by its design assumption: that when solutions have the same CBS, the one with more occurrences of large cyclic distances may be easier to improve. If this is the case, the ILS f_3 would be able to reach better local optima by navigating across neutral fitness areas. Most of the search performance evidence supports that claim. For example, consider the convergence plots for the best performing ILS configurations of each function presented in Figure 3.8. Where a stronger perturbation would cause the search lead by the original CBS function to get trapped in local optima, function f_3 allows the ILS to detect promising search directions in neutral areas and keep moving towards them.



(a) *c20k10*

(b) *path200*

(c) *dwt_209*

(d) *ran200P1*

Figure 3.8: Comparison of the convergence profiles of the ILS algorithm equipped with the studied evaluation functions on four representative instances.

There are a couple of outlier instances where some other alternative function bests f_3 regarding the ILS. The outlier instances are spotted in Tables 3.3 and 3.4 showing instance specific results and a victory based statistical significance assessment. In concrete, the ILS using f_1 for instances *c20k10* and *p20k10*, and f_2 for instance *p20c10*. However, it is also worth remarking that the results of the ILS using the original CBS function are often better than those when employing either f_1 or f_2, and always equal or worse in the case of f_3. Therefore, regardless of instance, the ILS demonstrated to be most reliable when it uses the function f_3.

Table 3.3: Performance of the ILS algorithm when using the four studied evaluation functions.

Graph	$\|V\|$	$\|E\|$	B	CBS *Avg*	CBS *Dev*	CBS I	f_1 *Avg*	f_1 *Dev*	f_1 I	f_2 *Avg*	f_2 *Dev*	f_2 I	f_3 *Avg*	f_3 *Dev*	f_3 I
c20c10	200	400	2360*	2521.36	489.60	26.10	3492.92	1228.57	139.02	2395.96	254.28	73.30	2422.44	309.24	246.38
c20k10	200	1100	5300	10251.40	946.65	2.44	6725.40	1253.63	37.08	7596.00	943.50	20.88	7943.28	1502.48	39.02
k20k10	200	2800	62300*	63755.28	2022.45	1.02	83941.12	925.83	37.42	62300.24	0.66	23.86	62300.00	0.00	33.08
p20c10	200	390	2238*	2455.68	399.46	25.42	3251.12	897.94	140.94	2263.92	1.66	71.70	2335.58	268.12	240.90
p20k10	200	1090	5200	9925.76	1040.17	2.96	6169.62	702.93	38.20	7016.00	831.61	21.72	7119.30	1228.09	43.64
p20p10	200	370	1991*	2092.58	168.86	47.42	2684.10	363.98	141.56	2199.94	152.18	61.78	2015.12	34.83	229.06
bip100-100	200	10000	500000	500000.00	0.00	8.96	500000.00	0.00	4.28	500000.00	0.00	4.50	500000.00	0.00	8.06
path200	200	199	199	352.12	69.76	57.40	738.94	65.79	111.18	489.66	74.28	36.88	363.26	51.51	97.92
cycle200	200	200	200	378.68	81.20	58.76	760.92	71.55	107.72	531.28	80.94	37.84	389.08	56.41	93.16
cycleP200-10	200	2000	11000*	16951.28	5441.33	22.70	15153.60	5602.16	61.02	15230.60	6110.02	64.14	16238.20	5297.60	48.62
wheel200	200	398	10200	10439.72	75.58	27.70	10837.24	74.86	56.22	10611.28	89.39	18.68	10421.64	53.25	49.98
can_229	229	774	6243*	6915.46	1169.85	12.80	9402.98	1988.30	61.26	6671.48	885.43	31.30	6264.80	4.23	91.88
dwt_209	209	767	6355*	7369.46	508.75	9.48	9629.50	962.30	50.18	7587.88	525.68	26.92	6677.90	328.73	120.04
steam1	240	1761	24158*	31853.64	3633.94	1.00	45230.10	4189.08	1.00	34145.88	3490.00	1.00	31261.18	2842.05	1.00
ash219	219	431	6229*	6644.94	239.91	11.32	9766.48	490.43	130.04	7357.54	352.16	40.40	6410.88	216.71	137.30
will199	199	660	13699*	13976.68	218.81	18.58	17658.06	625.05	115.94	15132.26	256.08	48.82	13801.82	50.37	144.76
ran200P1	200	1991	69993*	71291.26	386.19	5.84	76954.72	564.40	52.44	73234.92	456.13	26.64	70444.86	230.92	44.64
ran200P3	200	5970	255001*	258634.08	931.37	1.30	265569.88	941.25	21.90	260282.90	811.22	10.24	256094.34	467.71	13.34
ran200P5	200	9955	451266*	455114.62	995.99	1.00	461683.78	1167.28	12.74	456302.98	826.20	6.42	452517.20	592.95	6.24
ran200P7	200	13827	648917*	654312.52	1174.19	1.00	658789.40	1049.85	11.08	654406.30	885.77	4.34	651048.42	697.50	4.30
O-RMSE				29.57%			59.76%			31.05%			21.50%		

Table 3.4: Statistical analysis for comparing the performance of the ILS algorithm when using the four analyzed evaluation approaches.

	Instance																				
Function	c20c10	c20k10	k20k10	p20c10	p20k10	p20p10	bip100-100	path200	cycle200	cycleP200-10	wheel200	can_229	dwt_209	steam1	ash219	will199	ran200P1	ran200P3	ran200P5	ran200P7	Overall
CBS / f_1	+	−	+	+	−	+	⋆	+	+	⋆	+	+	+	+	+	+	+	+	+	+	16 + 2 −
CBS / f_2	⋆	−	−	−	−	+	⋆	+	+	⋆	+	−	⋆	+	+	+	+	+	+	⋆	10 + 5 −
CBS / f_3	⋆	−	−	−	−	−	⋆	⋆	⋆	⋆	⋆	⋆	−	⋆	−	−	−	−	−	−	0 + 12 −
f_1 / f_2	−	+	−	−	+	−	⋆	−	−	⋆	−	−	−	−	−	−	−	−	−	−	2 + 16 −
f_1 / f_3	−	+	−	−	+	−	⋆	−	−	⋆	−	−	−	−	−	−	−	−	−	−	2 + 16 −
f_2 / f_3	⋆	⋆	−	+	⋆	−	⋆	−	−	⋆	−	−	−	−	−	−	−	−	−	−	1 + 14 −

3.5.3 Dealing with search cycles

Search cycles are present when, after the perturbation, the search falls again in the attraction basin of some previously visited local optimum. The occurrence of search cycles can be a symptom of a deficient search and it is prejudicial because it represents a waste of execution time budget.

In order to identify search cycles occurrences in the ILS, a sample of 10,000 local optima visited by the algorithm with each of the functions was created, for four representative instances *c20k10*, *path200*, *dwt_209* and *rand200P1*. The method for identifying search cycles was to measure the distance between pairs of local optima. The distance metric was the interchange distance [21], which is suitable for cyclic permutations.

Figure 3.9 allows to visualize the search cycle occurrences for the fist 250 local optima in the sample for instance *dwt_209*.According with Figure 3.9, search cycles are much more common when the ILS uses the CBS function than when it is guided by any of the alternative functions. The search cycles issue here is associated with the potential of discrimination. Among the alternative functions, f_2 is the only one that causes some search cycles in the ILS and it is also the one with the lowest discrimination, besides the CBS function; and when using either function f_1 or f_3, which have the two highest potentials of discrimination, the ILS does not present search cycles. This is evidence that a low discrimination capability and the blind search decisions function f_2 induces result in a worst performing exploration of the search space in the form of search cycles.

While a high potential of discrimination is helpful for avoiding the search cycles, and their presence has a negative impact, their absence does not necessarily guarantees a successful search. That is the case for the ILS when it uses function f_1, because even if f_1 has high discrimination and it does not induces search cycles, its compatibility is the lowest. Therefore, the same local optima are not revisited, but the search can be still focusing in poor quality regions of the search space because the fitness assigned to them by function f_1 is inconsistent with their CBS cost.

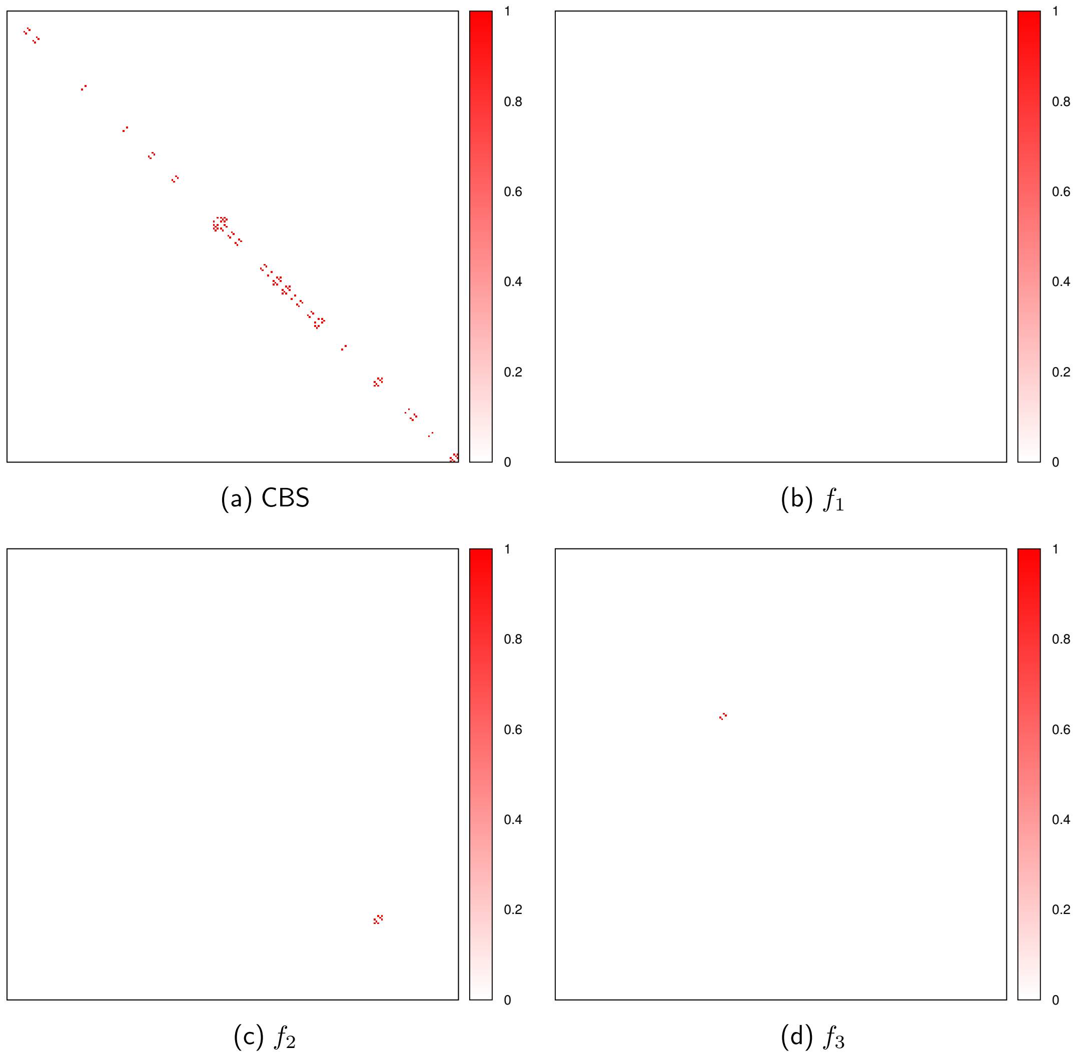


Figure 3.9: Matrices of interchange distances for local optima visited by the ILS using each of the functions. A search cycle occurrence is a revisit of a local optima. Its occurrence is represented by a red cell in the main diagonals, where the interchange distance between two local optima was equal to zero.

The analysis regarding search cycles allowed us to better understand how the potential of discrimination and CBS-compatibility affect search dynamics and why function f_3 was more successful guiding both the SD and ILS as a result of its high scores in both those metrics.

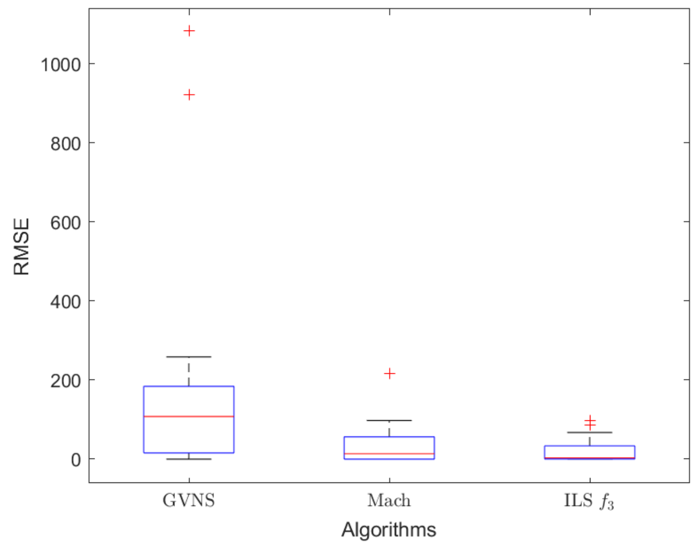


Figure 3.10: RMSE values of the ILS algorithm using function f_3 compared with GVNS and MACH

3.5.4 ILS guided by function f_3 compared with algorithms from literature

Across all the experiments discussed in the previous section, function f_3 matched all the criteria and proved to be a successful alternative to the CBS function. To conclude the analysis on the usefulness of function f_3, the performance of the ILS algorithm using f_3 was compared with that of the algorithms from the CBSP literature, GVNS and MACH. The results detailing O-RMSE, best and average cost and its respective deviation for the instance set are presented in Table 3.5, followed by the pair-wise analysis of statistical significance in Table 3.6.

It is remarkable that the very simple ILS implementation using function f_3 achieved significantly better results than GVNS for all instances, as well as better or equal than MACH for 14 of them. The O-RMSE for the ILS is the lowest among these three algorithms. As shown in Figure 3.10, the RMSE values have a close to zero median and a smaller variation, indicated by their compact box plot and the distance of outliers to the median. Also, the ILS was able to produce optimal solution for 2 out of 5 instances with known optimal value and 14 new best-known solutions for the rest of the considered graphs.

Table 3.5: Performance comparison of the ILS algorithm, equipped with the evaluation function f_3, with respect to two state-of-the-art methods.

Graph	$\|V\|$	$\|E\|$	*Opt*	*UB*	GVNS *Best*	GVNS *Avg*	GVNS *Dev*	MACH *Best*	MACH *Avg*	MACH *Dev*	ILS *Best*	ILS *Avg*	ILS *Dev*
c20c10	200	400		1560	6432	6467.80	24.66	2446	3754.24	1097.34	2360	2422.44	309.24
c20k10	200	1100		50380	15578	15733.00	49.57	5300	5300.00	0.00	5300	7943.28	1502.48
k20k10	200	2800		103300	78610	78911.00	316.92	76614	76973.88	139.44	62300	62300.00	0.00
p20c10	200	390		3790	5657	5663.48	8.30	2256	2256.00	0.00	2238	2335.58	268.12
p20k10	200	1090		100190	15092	15298.88	76.67	5200	5200.00	0.00	5200	7119.30	1228.09
p20p10	200	370		2080	5434	5441.40	5.35	4482	6271.36	703.74	2004	2015.12	34.83
bip100-100	200	10000	500000		500014	500014.00	0.00	500000	500000.00	0.00	500000	500000.00	0.00
path200	200	199	199		2350	2355.20	4.16	199	199.00	0.00	234	363.26	51.51
cycle200	200	200	200		2036	2042.88	7.30	200	200.00	0.00	220	389.08	56.41
cycleP200-10	200	2000	11000		39210	39430.16	77.25	11070	11217.00	72.84	11000	16238.20	5297.60
wheel200	200	398	10200		12476	12476.00	0.00	10200	10200.00	0.00	10280	10421.64	53.25
can_229	229	774		44505	13842	13933.84	37.31	7764	11899.82	2358.96	6255	6264.80	4.23
dwt_209	209	767		40268	13576	13639.50	32.27	8264	10169.66	1124.26	6371	6677.90	328.73
steam1	240	1761		106102	51938	52210.00	158.50	36713	40831.46	2373.94	25913	31261.18	2842.05
ash219	219	431		23705	10364	10397.42	18.94	8335	8955.44	339.76	6245	6410.88	216.71
will199	199	660		33000	17613	17657.68	8.64	19218	21205.34	788.45	13738	13801.82	50.37
ran200P1	200	1991		100050	76487	76533.32	27.12	88900	91052.80	1410.98	70037	70444.86	230.92
ran200P3	200	5970		300000	269521	269526.92	29.30	294397	294397.00	0.00	255192	256094.34	467.71
ran200P5	200	9955		500251	472432	472466.08	22.22	502332	504694.76	1334.77	451417	452517.20	592.95
ran200P7	200	13827		694824	668810	668853.12	6.22	703730	704559.46	947.28	649478	651048.42	697.50
O-RMSE						182.90%			34.96%			21.01%	

Table 3.6: Statistical analysis for comparing the performance of the ILS algorithm using evaluation function f_3 against that of the state-of-the-art methods.

Function	c20c10	c20k10	k20k10	p20c10	p20k10	p20p10	bip100-100	path200	cycle200	cycleP200-10	wheel200	can_229	dwt_209	steam1	ash219	will199	ran200P1	ran200P3	ran200P5	ran200P7	Overall
GVNS / MACH	−	−	−	−	−	+	−	−	−	−	−	−	−	−	−	+	+	+	+	+	6 + 14 −
GVNS / ILS	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	0 + 20 −
MACH / ILS	−	+	−	+	+	−		+	+		+	−	−	−	−	−	−	−	−	−	6 + 12 −

3.6 Conclusions

This chapter exposed the issues associated with using the CBS function and the setbacks that can arise as result of employing it as fitness function in search algorithms. With the purpose of countering those issues, three alternative fitness functions were proposed, each of them designed to introduce a notion of preference among potential solutions that belong to the same fitness class under the CBS function.

The three alternative functions were extensively tested under the three criteria of an evaluation framework for alternative functions that include a) better discrimination ability, b) compatibility with the definition of the problem, and c) demonstrable success guiding search algorithms. The first criteria was addressed by determining the potential of discrimination of the CBS function and the three alternatives using the relative entropy (RE) metric. It was demonstrated that the three proposed functions provide better discrimination than the CBS function, with functions f_3 and f_1 scoring perfect and near-perfect RE values. Regarding the second criteria, the CBS-compatibility definition was introduced and empirically measured, finding relative compatibility percentages of 83.40% for f_1, 92.01% for f_2, and 100% for f_3. The interplay of potential of discrimination and CBS-compatibility was discussed along with the third criteria, analyzing the search performance of the SD and ILS algorithms for each of the functions. It was demonstrated that a high discrimination is only useful if the consistency problem definition is maintained. On this regard for both SD and

ILS, function f_1, which has high discrimination but poor compatibility, was the weakest contender. Compared with f_1, function f_2, with not as high discrimination, got better results than f_1 and the CBS function, due to its better compatibility. The best performance for both the SD and the ILS across different perturbation values was achieved when the algorithms employ function f_3, which has both the best discrimination and compatibility scores.

Consistent results were achieved for the ILS with different perturbation strengths. It was found that function f_3 benefits from stronger perturbations without loosing sight of the good search space regions. It was determined that the search decisions function f_3 takes when it encounters equal CBS solutions (giving preference to solutions with many large cost cyclic distances) resulted in a more effective exploration of the search space, avoiding search cycles and taking advantage of larger perturbations. Our analysis regarding search cycles demonstrated that their occurrence is linked to discrimination capabilities of the fitness functions. It was shown that search cycles are harmful to the search, but also that their absence does not guarantee good search results if the alternative function looses CBS-compatibility.

Finally, the simple ILS implementation equipped with function f_3 was compared against two algorithms from the problem literature. In this experiment the ILS achieved lower O-RMSE rates, significantly improved the results for 70% of the instances, including 2 optimal solutions and 14 new best-found solutions.

The remaining of this works deepens on the study of cases where function f_3 is more effective, while also studying the effect that of other algorithmic components have over diverse solving approaches. The next chapter continues with the investigation of the effect of the alternative evaluation scheme over the performance of Memetic Algorithms employing different configurations of genetic operators.

4 Memetic Algorithms

4.1 Introduction

Memetic algorithms (MA) surged as an approach to optimization through the evolution of biological and behavioral aspects of the individuals in a population, in the form of a genetic algorithm combined with local search. The genetic algorithm is already powerful and one of the most widespread techniques in evolutionary computing. As a population based metaheuristic, it is able to consider several search space regions simultaneously, with the crossover and mutation mechanisms keeping balance between the exploitation of promising regions and the exploration of new ones. The incorporation of local search in the MA adds on extra layers of exploitation and exploration, since it is meant to improve the known solutions generated by the evolutionary process and it can also result in the addition of new genetic material to the population.

MA were initially proposed for solving the TSP [88], and through the decades numerous other optimization problems have been successfully tackled using MA, including GEP such as the antibandwith [102] and the BSP [106]. The previous chapter of this work discussed the promising results obtained by using our proposal of a more informative fitness function in very simple local search methods. In that particular scenario, function f_3 was remarkably successful. We continue by further researching the use of function f_3 into the MA: a population based evolutionary metaheuristic with a local search component. The success of the MA depends on the genetic operators it utilizes and how those operators interact with the problem in question. We investigated this aspect by considering four selection schemes, two recombination mechanisms, three mutation schemes, two survival strategies under two different evaluation schemes, the conventional CBS function and the new function f_3. The results provide insights on how to design a successful MA for the CBSP by analyzing how particular operators and the fitness function interact and affect the MA decisions and overall performance.

The rest of this chapter continues by introducing our MA implementation and the operators that participate in it in Sections 4.2 and 4.3. In Section 4.4 we first address the analysis of how the operators (the interactions among themselves and with the fitness function) affect the performance of the MA and later discuss the top five best performing MA implementations. Section 4.5 contains the comparison of our best performing MA with the algorithms from the problem's literature. Finally, Section 4.6 closes the chapter by summarizing the main findings and how some of they relate to the contents of Chapter 5.

4.2 The memetic algorithm framework

All our memetic algorithms were implemented over the same basic skeleton, with code structures that can be used interchangeably by the different operators. The main routine receives as parameter a set defining the specific operators for selection, crossover, mutation, fitness function and survival strategy. Across this work each specific combination of operators is referred as an *operator configuration*, or just a *configuration* for short. The operators available [110, 111] are four selection schemes [128]:

stochastic, roulette, random and binary tournament; two crossover operators [26, 92]: cyclic and order-based; three mutation operators [128]: insertion, reduced-3-swap and cumulative-swap; two fitness functions: the CBS function and f_3; and two survival strategies [128]: (μ, λ) and $(\mu+\lambda)$. Each of them is described in detail in the following section. In total, there are 96 operator configurations, each one of them creating a different MA implementation, which are referred as numerated MA versions.

Once the operator configuration is defined, all MA versions follow a similar pattern, starting by the initialization of population P and then the population evaluation by the specified fitness function f. In order to keep track of the best-found solution, even if it disappears from the population, a copy of the fittest individual in the initial population is stored in g and kept updated accordingly through the entire process.

A generation begins with the selection operator s picking a couple of candidate parent individuals $P_a, P_b \in P$. After selection, the crossover operator c produces an offspring individual o by recombining parent individuals P_a and P_b, with probability $prob_c$. With probability $1 - prob_c$, individual o is instead a copy of the fittest between the candidate parents. To keep the population diverse, new genetic material is incorporated by employing operators for mutation and inversion. First, the chromosome of the offspring individual o is altered by the mutation operator m (producing individual o'), and later by the inversion fixed operation (producing individual o''), under probabilities $prob_m$ and $prob_i$, respectively. The final offspring individual o'' joins the offspring population O, but the best-found solution g is updated considering also individuals o and o'. Next, the survival determines which individuals remain in the population for the next generation. Finally, the local search is applied to only the fittest individual in the surviving population. The algorithm ends when a stop criterion is met, in this case, a maximum number of fitness function evaluations.

Algorithm 1 Memetic Algorithm

```
 1: input A set of operators $\{s, c, m, f, ss\}$
 2: output The best-found solution $g$
 3: $P \leftarrow$ initializePopulation($P$)
 4: evaluate($f$, $P$)
 5: $O \leftarrow \emptyset$
 6: $g \leftarrow P_{best}$
 7: repeat
 8:     for $j \leftarrow 1$ to $\mu$ do
 9:         $P_a, P_b \leftarrow$ selection($s$, $P$)
10:         $o \leftarrow$ crossover($c$, $P_a$, $P_b$, $prob_c$)
11:         $o' \leftarrow$ mutation($m$, $o$, $prob_m$)
12:         $o'' \leftarrow$ inversion($o'$, $prob_i$)
13:         $O \leftarrow O \cup o''$
14:         $g \leftarrow$ fitter individual among current $g$, $o$, $o'$ and $o''$ under function $f$
15:     end for
16:     $P \leftarrow$ survival($ss$, $P$, $O$)
17:     $O \leftarrow \emptyset$
18:     $P_{best} \leftarrow$ localsearch($P_{best}$, $tries$)
19:     $g \leftarrow$ fitter individual among current $g$ and $P_{best}$
20: until stop criteria is met
21: return $g$
```

4.2.1 Solution encoding and initialization

We choose a permutation based encoding because it simplifies the implementation of the operators and it keeps the complexity of determining cyclic distances simple. This makes it easy to compute the fitness of individuals and to use quick partial recalculations when necessary, for example after a mutation that affects only a couple of vertices.

A permutation of n distinct consecutive numbers can directly be employed as an embedding, where each element represents a guest vertex, and their index in the permutation represents a host vertex, or vice-versa. In fact, our MA implementation uses both cases, defining them as labelings, so that every mapping from a guest vertex to a host vertex takes just one operation. Under these conditions, an individual is formally defined as follows [110]: An individual is represented as $P_i = (\varphi_i, \rho_i, f_i)$ where φ_i and ρ_i are two labelings representing the same embedding: $\varphi_i(u)$ stands for the label associated to vertex u (i.e., the vertex in the host graph associated to vertex u). $\rho_i(u')$ denotes the vertex in G having the label u' (i.e., the vertex hosted in vertex u'); and $f_i = f(\varphi_i, G)$ is the fitness of the individual assessed by the fitness function. Most of the operators work primarily over φ_i, then ρ_i is updated to reflect the changes. The only exception is the *insertion* mutation, which operates vice-versa over ρ_i.

The population was initialized with random permutations because in that way the performance of the MA is not conditioned by any fancy initialization, which allows to focus the analysis solely on the interplay of its operators.

4.3 Operator set

Each operator plays a particular role in the MA. The selection process is intended to introduce fitness based preferences, in such a way that the fittest individuals have more opportunities to reproduce and pass on their genes to the offspring. From a search perspective, the recombination of visited solutions via crossover is a form of exploiting the intermediate regions around known areas of the search space, as it is expected that the offspring of fit parents results even fitter. However, when the genes of the previous fit individuals proliferate along the generations and individuals in the population eventually become genetically similar to each other, the search process converges towards an increasingly narrow search space region. In such conditions, the scope for exploration is reduced and the evolutionary process would loose the ability to generate individuals that differ from the existing population. Using mutation to introduce new genetic material helps to diversify the genetic pool and prevent that the convergence occurs prematurely. The survival strategy also plays a relevant role in this aspect. While the survival of the fittest makes sense at first glance, being too restrictive can have negative impacts on diversity. For example, mutation would result pointless if the individuals carrying new genes were not be allowed to thrive. The local search is also a valuable opportunity to search independently of the evolution process, but overflowing the population with local optima too quickly can result in being unable to escape from them.

An adequate balance between a high fitness population and diversity can be greatly important for avoiding the potential problems that can arise within a MA. Our choosing of the operators studied along this research is intended to consider the strengths of a variety of approaches to do so.

Since we are interested in examining the interaction of MA components and the alternative evaluation scheme in the context of the CBSP, we assembled a set of genetic operators that features a diverse assortment of approaches for shaping the evolutionary process. This section introduces each of those operators and discusses what they offer in terms of creating a successful MA for our problem. It also presents some particularities in our MA implementation, specifically regarding mutation and local search.

4.3.1 Fitness function

Most of the decisions in the evolutionary process of the MA are made in terms of fitness, therefore the definition of the evaluation scheme is relevant. In the previous chapter we exposed the discrimination related issues that the problem's objective function can introduce in search algorithms. The research regarding the proposal of an alternative evaluation function yielded promising results, with function f_3 helping simple local search algorithms to significantly improve compared with how they perform when using the conventional CBS function and even allowing them to achieve better solutions than those reported in the literature. We are interested in observing the effect that function f_3 has over a multi-trajectory evolutionary algorithm, such as the MA, and examining how the operator's interaction with the fitness function reflects on the overall performance.

For efficiency, both the CBS function and function f_3 were implemented to compute the fitness in an incremental fashion whenever suitable. Specifically, full calculation of all cyclic distances occurs only after the initialization and crossover procedures. Since the operators for mutation, inversion and local search modify small sections of the chromosome, the fitness values are calculated in an incremental way, as described in Section 3.3.1.

4.3.2 Selection

Selection approaches often focus on creating a dependency between the fitness of individuals and their chances to produce offspring. Typically, the fitter an individual is, the higher its chances to reproduce. However, the exclusion of the individuals with worse fitness can result in limiting the scope for exploring new areas of the search space that are not immediately proven better.

The selection scheme is the operator for which we considered the highest number of alternatives, four in total: stochastic, roulette, random and binary-tournament. Selection shapes decisions that occur later in the generation, it determines which individuals would even be considered for crossover, mutation and local search, and in this way it influences the search directions that would be further explored. This does not mean that the rest of operators are not relevant, but instead that selection has the capacity to either overshadow or highlight the effects of other operators. For example, if the individuals produced by mutation are not picked by an over pressuring selection, it is almost as if they were not in the population. The first two selection strategies are stochastic selection and roulette. Both assign deterministic expected values to each individual and then conduct a drafting process. The expected values are set in function of the relative fitness of individuals with respect to the rest of the population. Higher expected values correspond to better fitness and thus more opportunities for an individual to be chosen. Since we are dealing with minimization, the expected value $ev(P_i)$ for an individual P_i is computed over its normalized fitness $\hat{f}(P_i)$. The min-max normalization of fitness is depicted in Equation 4.1

$$\hat{f}(P_i) = \frac{f(P_i) + f(P_{worst})}{\overline{f}} + 1.1\,, \tag{4.1}$$

where $f(P_i)$ is the fitness of an individual, $f(P_{worst})$ is the fitness of the worst individual in the population and $\overline{f}$ is the population average fitness. Equation 4.2 shows the calculation of expected values, where 2μ is the number individuals to be drafted in order get μ couples that would produce one descendant each. Therefore, the sum of the expected values is equal to 2μ.

$$ev(P_i) = \frac{2\mu \hat{f}(P_i)}{\sum_{j=1}^{\mu} \hat{f}(P_j)} \tag{4.2}$$

The drafting process for stochastic selection picks each individual as many times as the integer part of its expected value indicates. Then, any remaining gap on the total number of selected individuals is filled by using the floating part of the expected values, in a probabilistic fashion, to decide if the correspondent individual will be chosen once more.

In the roulette selection, the expected values set the size of the roulette portion that corresponds to each individual. It follows that fitter individuals have larger portions assigned and more chances to be picked. Each drafting occurs simply by spinning the roulette.

Both stochastic and roulette selection rely strongly on the fitness of the individuals. Stochastic selection combines the direct use of expected values with independent probabilities, since the floating part does not depend of fitness. Roulette selection may be the most restrictive regarding fitness, because implements fixed fitness dependant probabilities for each individual and the final set of picks would reflect the expected values quite closely.

The random selection does not consider fitness. Instead, all individuals have a uniform probability of being selected. Since we are doing that 2μ times, it can be assumed that every individual would be chosen 2 times on average.

Binary tournament has a different approach. It does not require expected values and the fitness distribution plays an underlying role. It works similarly to the random selection, in the sense that all individuals have uniform probabilities to be drafted, but instead of picking just one individual per draft, it picks two, compares their fitness and selects the one that is fittest. This introduces fitness preferences, where the fittest individual among the population will never be defeated, but it keeps opportunities to the rest of individuals to be chosen if the tournament pairs them with an equal or worse opponent. The worst individual in the population has chances to be picked if it is paired with itself or a fitness equal in the tournament.

Both stochastic and roulette selection involve the calculation of expected values that can be performed in linear time with μ steps for the average fitness calculation, μ steps for the normalization and μ steps for the expected value assignation. This whole process would take approximately 3μ steps. The complexity of stochastic selection is determined by the operation of deciding the number of times the individuals are picked. This is linear as well, and it stops when 2μ individuals have been chosen, so the corresponding complexity is $\mathcal{O}(2\mu + 3\mu) \approx \mathcal{O}(\mu)$. Roulette is the most complex among the four selection operators. It performs 2μ spins, with the cost of determining the winner of a spin being at most μ steps. Since this would result in a quadratic expression, we omit the linear number of steps for the expected values. Its complexity is then $\mathcal{O}(2\mu(\mu)) \approx \mathcal{O}(\mu^2)$. The random and binary tournament selection are the simplest ones, performing 2μ random drafts. The comparison between the drafted individuals for the binary tournament selection is a constant operation, so we have a complexity of $\mathcal{O}(2\mu) \approx \mathcal{O}(\mu)$ for both.

4.3.3 Crossover

The considered crossover operators are permutation based in order to produce feasible chromosomes under the previously described solution encoding, while the differences between them allow us to examine the effect that implicit mutations have on overall performance. Implicit mutations can be introduced during the crossover operation, resulting in an offspring individual that has genes in different positions than the ones they occupied in any of the parents. On one hand, implicit mutations are a way of introducing genetic diversity within crossover, but on the other hand, their combined effect with the actual mutation could result too disruptive.

The cyclic crossover operator [92] employs the notion of cycles between two permutations. By definition, these cycles do not overlap. Therefore, a new chromosome can be obtained by inheriting alternatively different cycles from each parent. Furthermore, every gene in the resulting chromosome will be in the same position that it was in the parent individual from whom the cycle was inherited. This avoids the occurrence of implicit mutations during crossover.

Cyclic crossover uses a mapping for determining the cycles between permutations that takes n^2 steps to be created. Then, the cycles can be inherited in linear time with respect to their lengths, which at most sum n, for a complexity of $\mathcal{O}(n^2 + n) \approx \mathcal{O}(n^2)$.

The order-based crossover defines a random section of the chromosome where the offspring individual directly inherits the corresponding genes from one of the parents. The other parent individual provides the genes to fill the remaining part of the offspring's chromosome. Since the result must be a permutation, these genes are inherited in the same relative order they appeared in the parent's chromosome. In contrast with the cyclic crossover, the order-based crossover allows potential implicit mutations limited to a section of the chromosome.

The complexity of order-based crossover is $\mathcal{O}(c + n(n - c)) \approx \mathcal{O}(n^2)$, where c is the number of genes directly inherited from one of the parents and it takes at most n steps to find the next free gene in the other parent's chromosome.

4.3.4 Mutation

The role of mutation is to introduce new genetic material in the population, with the purpose of preventing that the lack of diversity causes the search to converge prematurely. However, not all diversification is equal, some mutations can lead to further fitness improvements, others can result in loosing fitness. Genes that negatively affect the fitness would eventually be purged from the population by the selection or survival mechanisms. Then, why would it be a problem if the mutation operates without regarding fitness? To begin with, it could drag the search away from promising regions. For example, in rugged fitness landscapes where relatively close solutions exhibit abrupt fitness variations a reckless mutation would cause the average fitness of the population to drop, thus reducing the chances for further improvements. If we account for mutation affecting several individuals per generation, the resources employed in producing individuals that lead to no improvements and the eventual purging of poor genes became a potentially significant waste of often limited budgets in terms of time, number of evaluations or maximum number of generations.

The mutation operators we decided to include are meant to represent different strategies for mutation. The first is a classical insertion operation that takes into account the cyclic feature of the embeddings that the individuals represent. The focus of the second and third operators, reduced 3-swap and cumulative swap, is to also seek fitness improvements, under different levels of restrictiveness and number of affected genes.

The commonly used cyclic insertion is applied with probability $prob_m$ for each individual and modifies at most half of the genes. We define it in terms of what it does on regards to the embedding, as shown in the example of Figure 4.1. Let u' and v' be distinct randomly chosen host vertices. Recall that under our solution encoding, labeling ρ_i encapsulates an embedding by assigning guest vertices as labels to host vertices, in such a way that the label $\rho_i(u')$ is the guest vertex assigned to host vertex u'. The insertion mutation reassigns guest vertex $\rho_i(u')$ to host vertex v', and displaces the labels of vertices in between u' and v' in a clockwise or counterclockwise fashion, depending on which alternative affects the smaller number of labels. The insertion mutation does not consider fitness, but it thus will help us contrast the mutation operators that do it, as well as the ability of the rest of operators to compensate the potential drops in fitness.

With at most $n/2$ affected labels per individual, and if every affected node had the maximum degree, this would cause all the edges in the graph to be reevaluated. Since the maximal degree is $n-1$, then the complexity is $\mathcal{O}(n/2 + n/2(n-1)) \approx \mathcal{O}(n^2)$.

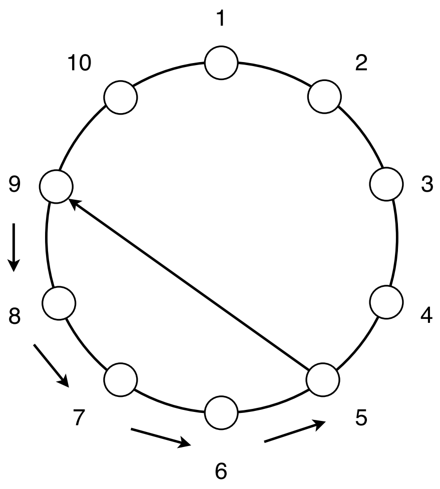
The design of the reduced 3-swap mutation is meant to diversify in a smart way, by combining the random picking of three genes to be modified and a fitness oriented choice. There are up to five different ways on which these swappings can be performed. The reduced 3-swap mutation computes the potential changes in fitness that each of the swappings would induce and performs the one that improves fitness the most, or the one that decrements it in the lesser amount, if there is not an improving one. This operator functions like a small exploration of the 3-swap neighbourhood, hence its naming. It follows that the solution created by reduced 3-swap is at a Hamming distance equal to three from the original solution, and that the same resulting solution could be achieved by two consecutive swaps with a common element. Since there are three affected nodes with at most $n-1$ neighbours each, assuming that those neighbours do not overlap, the complexity of evaluating the five mutation candidates is $\mathcal{O}(5(3(n-1))) \approx \mathcal{O}(n)$.

While the insertion mutation has no fitness preferences and reduced 3-swap considers fitness, but still allows non improving mutations, the cumulative swap operator performs only strictly improving mutations. This mutation operator executes $n/2$ iterations, choosing in each of them a pair of distinct random genes that are swapped with probability $prob_m$. However, each single swapping is only accepted if the fitness of the individual was improved by it. The cumulative swap is the only one that employs the mutation probability $prob_m$ at the gene level and also the one with the most variability on the number of affected genes. If none of the swaps it attempts results in a fitness improvement, there are no mutations at all, but if all $n/2$ attempts, under probability $prob_m$, are successful swaps, then the whole chromosome could be modified, therefore its complexity is $\mathcal{O}(n + \frac{n(n-1)}{2}) \approx \mathcal{O}(n^2)$.

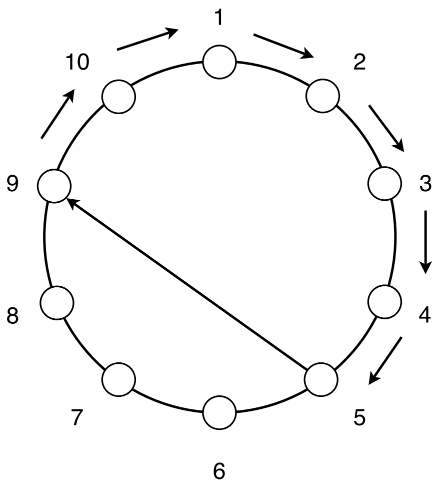
The mutation operators previously exposed are complemented by a fixed inversion operator that works independently. The inversion operator serves as an extra diversification layer that is not affected by the particularities of the mutation operators. It has its own probability $prob_i$, so individuals unaffected by mutation may still have a second chance, and mutated individuals can be modified further. The inversion picks a random section of the chromosome and inverts the order of the genes that are included in it, as shown in Figure 4.2. It can affect at most $n/2$ genes in a similar fashion than the insertion mutation, so its complexity is $\mathcal{O}(n/2(n-1)) \approx \mathcal{O}(n^2)$.

4.3.5 Survival

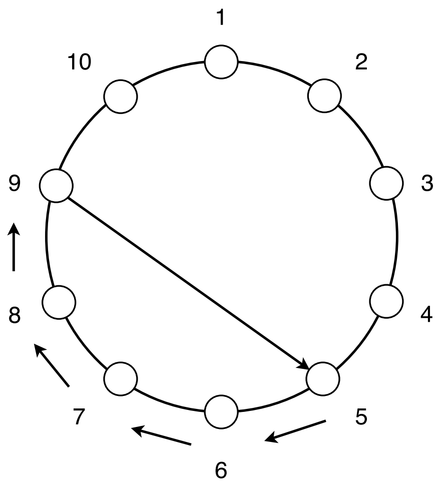
The survival strategies considered in this work are (μ, λ) and $(\mu + \lambda)$. Both are opposite ends of the spectrum between survival based on fixed life spans and pure fitness elitism. The (μ, λ) strategy replaces the population of parents with the offspring individuals, regardless of their fitness. In contrast, under $(\mu+\lambda)$ the fittest μ individuals among parents and offspring are the ones surviving.



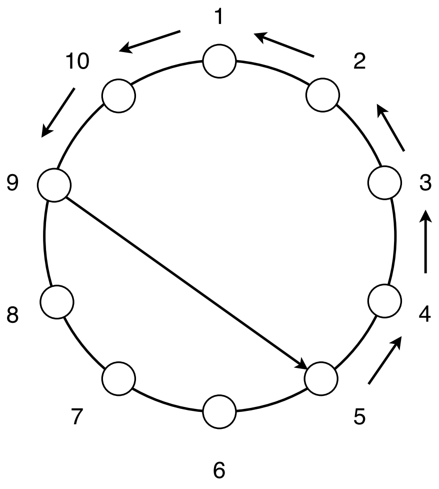
(a) Counterclockwise displacements for $u' = 5$ and $v' = 9$.



(b) Clockwise displacements for $u' = 5$ and $v' = 9$.

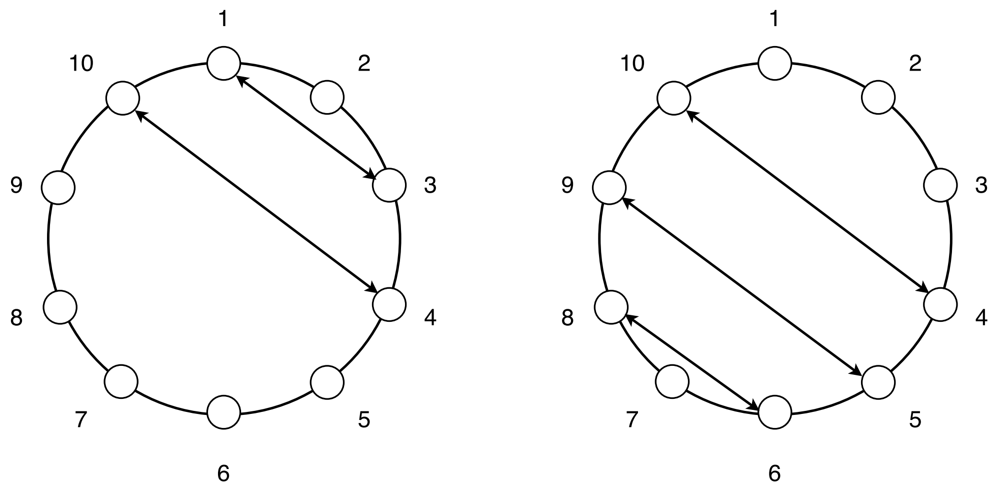


(c) Clockwise displacements for $u' = 9$ and $v' = 5$.



(d) Counterclockwise displacements for $u' = 9$ and $v' = 5$.

Figure 4.1: Insertion mutation. Numbers represent host vertices. Example in Figure 4.1(a) corresponds to counterclockwise insertion when $u' < v'$, performing five steps. Also for $u' < v'$, clockwise insertion will perform seven steps, as shown in Figure 4.1(b), therefore counterclockwise insertion is preferred. If instead $u' > v'$, clockwise insertion will perform five steps, while seven steps will be required by counterclockwise insertion, as shown in Figures 4.1(c) and 4.1(c). In this case, clockwise insertion is preferred.



(a) Clockwise inversion. (b) Counterclockwise inversion.

Figure 4.2: Inversion over φ_i. In Figure 4.2(a) a clockwise inversion between vertices $u = 10$ and $v = 4$ would perform two exchanges of labels. Figure 4.2(b) shows the respective counterclockwise inversion which performs three exchanges of labels, therefore clockwise inversion is preferred.

On one hand, allowing potentially fit individuals to leave the population after only one generation would cause the search to rely more on selection and crossover for managing to pass their genes to new offspring individuals. On the other hand, the proliferation of the genes of the fittest individuals throughout multiple generations is more likely to result in a genetically uniform population, specially in late search stages where new improvements become rarer.

The complexity of the (μ, λ) survival is simply $\mathcal{O}(\mu n)$ for directly replacing μ parents, each with n genes, by the offspring. Meanwhile, the $(\mu+\lambda)$ survival a more computationally demanding approach, because it requires sorting parents and offspring by fitness. Its complexity is $\mathcal{O}(\mu n + \mu \log \mu) \approx \mathcal{O}(\mu \log \mu)$, taking into account the sorting process carried on by a quicksort algorithm.

We considered that contrasting both strategies was interesting, from the point of view of gaining insight on the ability of the rest of operators for creating fitter offspring and introducing diversity in a smart way.

4.3.6 An intermittent non intensive local search approach

The local search approach incorporates design particularities that make it intermittent and less intensive regarding the number of individuals it tries to improve and the iterations it spends trying to do so.

At its core, the local search phase consists of a steepest descent, similar to the one that was employed for assessing alternative evaluation functions, described in Chapter 3. It also employs the 2-swap operator, but with a first-improvement move strategy. However, it has two main differences. The first one is that it is not employed on the whole population, but only on the fittest individual after survival P_{best}. The second one is that its execution is limited to a maximum number of iterations $tries$. In this way, the local search phase of the MA does not always produce a local optimum. Instead, it settles for a potentially improved individual that could be achieved within its limited budget.

Considering a complete neighborhood exploration, the local search visits at most $tries(\frac{n^2-n}{2})$ neighbouring solutions per generation. The number of affected edges that must then be reevaluated for each of them is no more than $2(n-1)$, if the swapped vertices have both maximum degree and their adjacent vertices do not overlap. Therefore, the complexity of the local search in the worst case is at most $\mathcal{O}(2(tries(\frac{n(n-1)}{2}))(n-1)) \approx \mathcal{O}(n^3)$.

The idea is that the remaining operators of the MA complement the work of the local search by exploring areas that may be passed over if the output was a local optimum. Since the individual entering local search is the fittest after the survival, is very likely that through selection and crossover its genes, or even complete copies of its chromosome, would be replicated into the next generation's offspring. Some of the mutation operators also incorporate their own fitness oriented behaviours that take inspiration from search and can add to the exploration effort.

When the local search is reached again in the following generation, there are two possible scenarios. Either P_{best} is the same chromosome from the previous generation, or it is a further improved one. If it is the first scenario, the local search phase can continue exploring exactly where it left in the previous generation, until its budget ends or it reaches a local optimum. In the later scenario, the local search restarts from the new P_{best} individual, trying to improve it within the time budget.

If the evolutionary process continuously replicates the genes of a population filled with local optima, improvements are only possible if the genetic operators eventually produce a solution that happens to be in the attraction basin of a new improving local optima. However, the genetic operators are often likely to direct the search towards the surroundings of the fittest individual, so they could be reducing the chances of exploring other areas. Therefore, a softer non intensive local search approach that does not fill the population with locally optimal solutions may help to avoid getting trapped by them.

4.3.7 MA complexity

Table 4.1 summarizes the worst case complexity for the set of operators and the two evaluation functions. From there we can determine that the most demanding MA would combine function f_3, the roulette selection, the cyclic crossover, any mutation different from reduced 3-swap and the $(\mu + \lambda)$ survival. In contrast, the least demanding would be combining the CBS function, random or binary tournament selection, order-based crossover, insertion or cumulative swap mutation and the (μ, λ) survival. By adding the complexity of the operators in each MA configuration we could find their respective complexities, but it is noticeable that all of them would include a cubic term for the evaluations during local search and a quadratic term for fitness evaluation, therefore all the MA are $\mathcal{O}(n^3 + \mu n^2)$, even the ones that use linear operators for selection, mutation or survival.

Table 4.1: Summary of the worst case complexity for the set of operators and fitness functions employed in the MA.

Operator		Main operations	Worst case complexity	
Fitness function	CBS function	Evaluating e cyclic distances	$\mathcal{O}((n^2-n)/2)$	quadratic
	function f_3	Weights recalculation and evaluating e weighted cyclic distances	$\mathcal{O}((n^2-n)/2+n)$	quadratic
Selection	stochastic	Selecting 2μ individuals based on the integer and floating parts of their expected values	$\mathcal{O}(\mu)$	linear
	roulette	Finding the winner for 2μ roulette	$\mathcal{O}(\mu^2)$	quadratic
	random	2μ random drafts	$\mathcal{O}(\mu)$	linear
	binary tournament	mu tournament matches	$\mathcal{O}(\mu)$	linear
Crossover	cyclic	Finding and inheriting cycles between permutations	$\mathcal{O}(n^2)$	quadratic
	order-based	Finding and inheriting unused genes in the correct order	$\mathcal{O}(n^2)$	quadratic
Mutation	insertion	Inserting a gene, evaluating up to $n/2$ displacements	$\mathcal{O}(n^2)$	quadratic
	reduced 3-swap	Evaluating five 3 swaps	$\mathcal{O}(n)$	linear
	cumulative 2-swap	Evaluating at most $n/2$ 2-swaps	$\mathcal{O}(n^2)$	quadratic
	inversion	Evaluating at most $n/2$ 2-swaps	$\mathcal{O}(n^2)$	quadratic
Survival	(μ, λ)	Replacing μ individuals, each one having n genes	$\mathcal{O}(\mu n)$	linear
	$(\mu + \lambda)$	Sorting 2μ by fitness and maintaining the fittest μ	$\mathcal{O}(\mu \log \mu)$	quasilinear
Local search	first-improvement	Up to $tries$ complete neighborhood explorations, with $2(n-1)$ affected edges per neighbor	$\mathcal{O}(n^3)$	cubic

4.4 MA operators and solution quality performance

The complete factorial experimental design that was employed allowed for the examination of the interactions between the genetic operators as well as to contrast the effect of the alternative fitness function on a varied collection of MA configurations. The four operators for selection, two for crossover, three for mutation, two survival strategies and the two alternatives for fitness function yield a total of 96 operator configurations.

For the experiments within this chapter we used a set of 40 instances that represent a diverse assortment of graph topologies that are commonly studied in the literature of GEP and the CBSP. The set includes Harwell-Boeing matrices, Cartesian products, random graphs, paths, cycles, wheels and powers of cycles.

Table 4.2: Input parameter values for the MA algorithms.

Parameter		Tested	Final value	Parameter		Tested	Final value
Population size	μ	10,20,50,100	20	Inversion rate	$prob_i$	0,1	0.240
Crossover rate	$prob_c$	0,1	0.788	Local search iterations	$tries$	5,10,15,20	10
Mutation rate	$prob_m$	0,1	0.543	Fitness function calls	T	-	4.0E+08

Table 4.3: Keys used to identify the operators involved in the MA configurations. For example, S1_C1_M1_SS1_V1 describes an operator configuration that combines stochastic selection, cyclic crossover, (μ, λ) survival and the CBS function to evaluate fitness.

Operator	key	Operator	key
Stochastic selection	S1	*Insertion* mutation	M1
Roulette selection	S2	*Reduced triple-swap* mutation	M2
Random selection	S3	*Cumulative swap* mutation	M3
Binary tournament selection	S4	(μ, λ) survival	SS1
Cyclic crossover	C1	$(\mu + \lambda)$ survival	SS2
Oder-Based crossover	C2	CBS function	V1
		Function f_3	V2

The stop criteria for each execution was a maximum number of fitness function evaluations. Table 4.2 lists the parameter values we use for the experiments with the different MA configurations. The values were assigned after performing preliminary tests supported by the irace utility [72] for automatized parameter tuning. When assessing the performance of the MA versions, we employ the O-RMSE metric introduced in Appendix A.2, which is calculated considering also results achieved by the GVNS and MACH algorithms from literature, and the methodology for testing statistical significance described in Appendix A.3. Tables comparing results of algorithms include either the upper bound or optimal value for each instance, calculated according to formulas presented in Section 2.3.1.

4.4.1 A global discussion of operator interactions

Ratter than discussing each operator configuration on an individual basis, our intention is trying to observe general tendencies regarding the behaviour of groups of MA that use a specific operator and evaluation function, and then contrasting the global tendencies with particular cases. For example, to analyze the selection operators we can look at how the groups of MA using each alternative perform and what does that imply about the helpfulness of said alternatives. Such an approach is intended to identify strengths in the operators and their combinations. We put special attention on the role of the fitness function by framing the performance variations of each MA group under the conventional CBS function and the alternative function f_3. For this analysis the MA were sorted into groups that share common features. MA belonging to the same group employ the same fitness function and one of the genetic operators for selection, crossover, mutation or survival. Global tendencies in the relationship between operators and solution quality can be identified by examining the O-RMSE across MA groups, while the RMSE distributions for single MA provide contrasting evidence for particular cases.

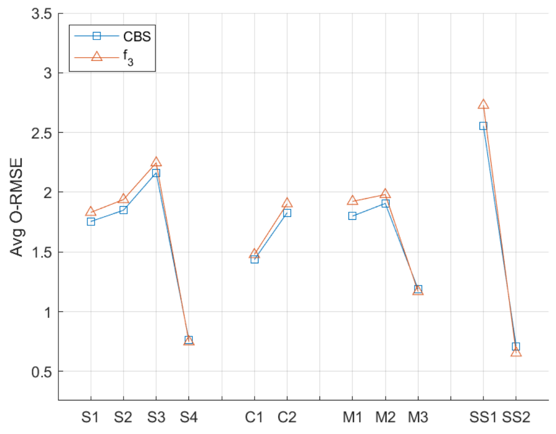
Figure 4.3 summarizes the O-RMSE variations for subsets of MA that employ each of the genetic operators variants when their fitness function is either the CBS function or f_3. Both the average (Figure 4.3(a)) and median (Figure 4.3(b)) of the O-RMSE are presented, as they give complementary perspectives to better understand the variations in performance and the scope of the general tendencies.

Selection and survival operators tend to be the most determinant operators for the success of the search, as it is shown by the average and median O-RMSE values of MA groups employing binary tournament selection and $(\mu+\lambda)$ survival, which correspond to the lowest O-RMSE values for any of the considered groups of MA. In both cases, the average and median O-RMSE are even better under function f_3. For the remaining genetic operators, crossover and mutation, the respective groups of MA have less cohesive behaviours, reflected by the differences between the average and the median of the O-RMSE, but in general, cyclic crossover and insertion mutation appear as the operators that help their respective MA groupings perform better.

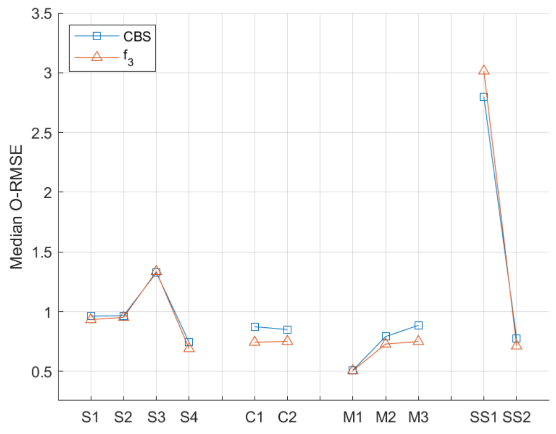
Across all the MA groups, function f_3 has the effect of lowering the group's O-RMSE median but increasing its average. This seems to indicate that function f_3 has a polarizing effect, often helping good performance MA to become better, but also causing the MA with poor performance to worsen. For example, in Figure 4.4, lets define the *worst* MA as those where the maximum non outlier RMSE value is over the 3% mark. If we compare the group of the worst MA under the CBS function with the group that has the same genetic operator configurations but instead uses function f_3, we found that the later has higher RMSE values per instance. In contrast, most of the best performing MA under the CBS function have better RMSE values reflected in more compact boxes and less disperse outliers. To understand why this happens we look at what makes the worse MA different from the rest: the use of (μ, λ) survival with a selection different from binary tournament, which are arguably the most harmful combinations among our genetic operators.

Survival (μ, λ) lets the offspring population directly replace their parents, regardless of their fitness, thus leaving selection on charge of most fitness decisions. Clearly, stochastic, roulette and random selection are not good enough for the task. Random selection has no fitness considerations at all; stochastic and roulette depend too much on expected values and with survival (μ, λ) making the population to rapidly change, they will tend to favor a different search direction on each generation.

Function f_3 was designed to help distinguish between neutral CBS solutions, but such distinctions are not helpful if used inadequately. Under function f_3, CBS neutral implicit and explicit mutations can become improvements, so the order-based crossover, the fitness oriented mutation and local search are more likely to perform movements that once evaluated, contribute to rapid genetic changes. Those changes will not result in CBS improvements, since the search directions they would have lead to will not be explored for more than a generation. Binary tournament selection provides the MA with a more reliable method to follow promising and diverse search directions, therefore the guidance of function f_3 is actually exploited and typically results in RMSE improvements.



(a) Average values



(b) Median values

Figure 4.3: Comparison of average and median O-RMSE values across MA with particular operators, grouped by their fitness function.

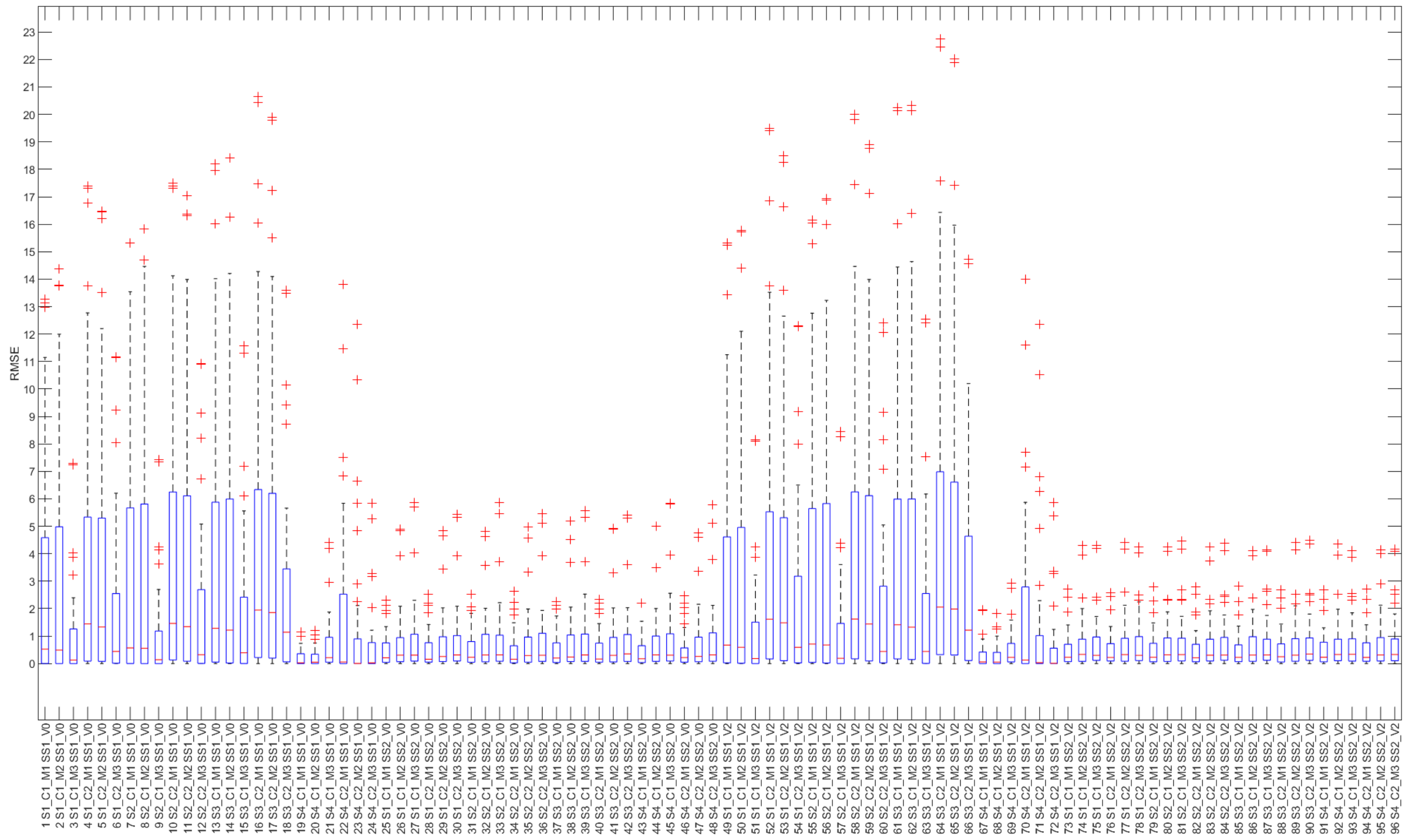


Figure 4.4: RMSE distributions for the 96 MA versions over the set of instances. Each version is lexicographical numerated according with the operator keys listed in Table 4.3, which are used to identify their operator configurations.

4.4.2 The top five MA

The hidden strengths of some other operators are also better employed when paired with the binary tournament. For example, the survival (μ, λ) is more characteristic of poor performing MA, the evidence of that is the large average and median O-RMSE for the group of MA that employ it. However, among the top five MA with the lowest O-RMSE values, listed in Table 4.4, the first four of them use (μ, λ) survival. These results show that the specific combination of binary tournament, cyclic crossover and either insertion or reduced 3-swap mutation creates the conditions for a diverse population where good genes are effectively inherited without the need of keeping around the individuals for several generations.

Selection by *binary tournament* ensures that individuals chosen for crossover are fit, without imposing an excessive selection pressure, so they can also be diverse. This helps to prevent the offspring population from getting too homogeneous. While the changes introduced by mutation are generally considered necessary for the exploration of new search areas, the results reveal that implicit mutations during crossover are not helpful for the overall performance of our MA. The implicit mutations during order-based crossover cause changes in the chromosome that can not be directly controlled or accounted for, so their effect combined with further actual mutations results too disruptive. Regarding O-RMSE values for the top five MA, the algorithms using insertion perform very similarly to the ones using reduced 3-swap. For example, MA-19 and MA-20 differ only on the mutation operator and both have close O-RMSE values and so do MA-68 and MA-67. The difference on execution time among the top five MA are linked to these mutation operators. The MA versions using reduced 2-swap complete the maximum number of fitness function evaluations quicker, due to the smaller number of affected genes.

In order to better understand how the algorithms in the top five MA compare to each other and to determine which one is better, we examined at their solution quality results in more detail. Figure 4.5 presents the distributions of their RMSE values and Table 4.5 a pair-wise assessment of statistical

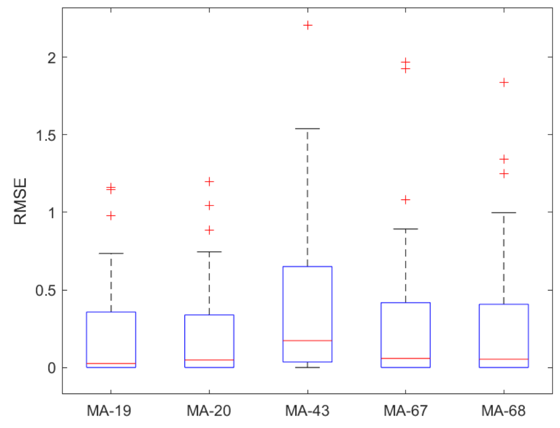
Table 4.4: O-RMSE ranking for the Top-5 MA versions, MACH and GVNS.

Rank	Algorithm	Operator configuration	O-RMSE (%)	Average execution time (s)	Time to local opt. (s)
1	MA-19	S4_C1_M1_SS1_V1	0.229	185.14	57.01
2	MA-20	S4_C1_M2_SS1_V1	0.227	230.24	68.66
3	MA-68	S4_C1_M2_SS1_V2	0.285	246.05	134.70
4	MA-67	S4_C1_M1_SS1_V2	0.287	294.64	158.31
5	MA-43	S4_C1_M1_SS2_V1	0.427	690.91	508.10
	MACH		0.496	7.72	
	GVNS		0.776	900.1	

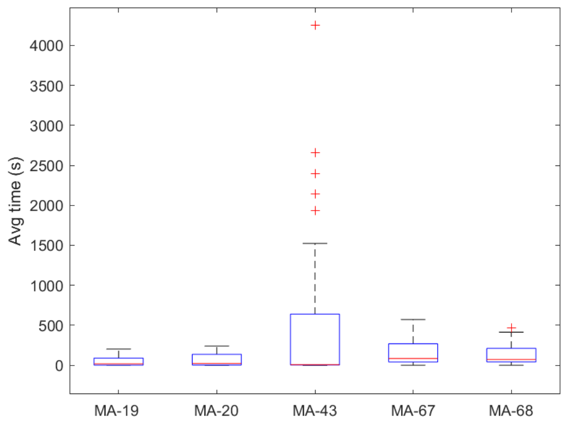
significance.

Among the top-five MA, MA-43 was the worst performing, with its notoriously worse solution quality and higher execution time. MA-43 has the largest median and average RMSE values and in the comparisons this algorithm very rarely defeats any of its pairs. It is remarkable than MA-43 differs from the first ranked MA-19 only in the survival scheme. While as previously discussed, focusing survival on fitness typically helps our MA to achieve lower O-RMSE values, there is a certain point in the search process at which it stops being helpful and instead causes the algorithm to converge prematurely. Since this type of survival will not allow a non improving individual to enter the population, it can accelerate convergence in early stages of the search, but as the search continues, the population is no longer able to explore enough. On top of that, the sorting process required by the $(\mu + \lambda)$ survival causes MA-43 to be by far the most demanding in execution time.

MA-67 and MA-68 are the respective function f_3 counterparts of MA-19 and MA-20. These four algorithms have close average O-RMSE rates, although the ones using the alternative function present higher outliers and require more execution time. In the Chapter 3 we show that function f_3 was able to help the SD and ILS algorithms to achieve better results, here we found that it is often the same for our MA implementations, but not in all cases. For example, MA-68 (ranked third according to O-RMSE) is able to get similar or significantly better solutions than MA-19 (ranked first) in 30 out of 40 instances. Moreover, MA-68's victories concentrate in Harwell-Boing instances, which are the bigger in order and size. Yet the remaining 10 instances where MA-19 defeats MA-67 are enough to increase its RMSE median and average stats, pushing it to the 3th ranking. The



(a) RMSE values.



(b) Average execution time.

Figure 4.5: Distribution of RMSE values and average time in seconds to the best-found solution for MA in the top five group.

explanation for this phenomenon relates to how function f_3 interacts with the local search. Our previous experiments with ILS and SD used the best-improvement strategy, the local search phase of the MA use instead first-improvement. We implemented first-improvement because we wanted to prevent the population from being quickly filled with local optima. The resulting side effect is that when the local search finds a neutral CBS neighbor solution, under function f_3 it can accept it as the first improvement. These neutral CBS steps that the local search takes are responsible of keeping MA-67 and MA-68 from reaching better search regions as fast as MA-19 and MA-20. However, we still remark that the differences on average O-RMSE for these algorithms are small and that there is a fair amount of statistically significant ties among them.

The MA version that dominates the instance by instance statistical comparisons is MA-20. Even when MA-20 ranked in second place, it is always the overall winner of all the statistically significance matches where it participates, including the one against the first ranked MA-19. Among all these matches, MA-20 is only defeated on seven instances. MA-20 offers the best compromise between good solution quality performance, execution time and statistical significance of the results and thus it was chosen to compare it against the state-of-the-art algorithms.

Table 4.5: Statistical analysis for comparing the performance of the Top5 MA group.

Instances	$\lvert V\rvert$	$\lvert E\rvert$	19 / 20	19 / 43	19 / 67	19 / 68	20 / 43	20 / 67	20 / 68	43 / 67	43 / 68	67 / 68
path100	100	99	−	−	−	−	−	⋆	−	+	⋆	−
path200	200	199	−	⋆	+	+	⋆	+	+	+	⋆	−
cycle100	100	100	+	+	⋆	+	+	−	+	−	⋆	+
cycle200	200	200	−	+	+	+	+	+	+	⋆	⋆	−
wheel100	100	198	+	+	−	+	+	−	+	−	⋆	+
wheel200	200	398	+	+	+	+	+	+	+	⋆	⋆	−
cyclePow100-2	100	200	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
cyclePow200-2	200	400	⋆	+	+	⋆	+	+	⋆	⋆	⋆	−
cyclePow100-10	100	1000	⋆	+	⋆	+	+	⋆	+	−	⋆	+
cyclePow200-10	200	2000	⋆	⋆	+	⋆	⋆	+	⋆	+	⋆	⋆
c9c9	81	162	⋆	⋆	⋆	⋆	⋆	⋆	⋆	−	⋆	⋆
c9k9	81	405	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
k9k9	81	648	⋆	+	⋆	+	+	⋆	⋆	−	⋆	⋆
p9c9	81	153	⋆	+	+	⋆	+	+	⋆	−	−	⋆
p9k9	81	396	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
p9p9	81	144	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
jgl011	11	49	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
ash85	85	219	⋆	+	⋆	+	+	−	⋆	−	−	+
curtis54	54	124	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
ibm32	32	90	+	+	⋆	+	+	−	⋆	−	−	+
will57	57	127	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
impcol_b	59	281	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
impcol_d	425	1267	−	−	⋆	−	−	+	⋆	+	+	−
nos4	100	247	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
nos6	675	1290	−	+	+	−	+	+	+	−	−	−
494_bus	494	586	−	+	+	−	+	+	+	−	−	−
662_bus	662	906	−	+	+	−	+	+	+	−	−	−
685_bus	685	1282	−	+	+	−	+	+	⋆	−	−	−
can_24	24	68	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
can_144	144	576	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
can_292	292	1124	−	+	+	−	+	+	+	−	−	−
can_445	445	1682	−	+	+	−	+	+	+	−	−	−
can_715	715	2975	−	+	⋆	−	+	+	⋆	−	−	−
bcspwr01	39	46	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
bcspwr02	49	59	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
bcspwr03	118	179	−	+	⋆	−	+	+	⋆	−	−	−
bcsstk01	48	176	+	+	⋆	+	+	−	⋆	−	−	+
bcsstk06	420	3720	−	+	⋆	−	+	+	⋆	⋆	−	−
dwt_503	503	2762	−	+	⋆	−	+	+	+	−	−	−
dwt_592	592	2256	−	+	⋆	−	+	+	⋆	⋆	−	−
+			5	28	12	10	28	18	12	4	1	6
−			15	2	2	13	2	5	1	24	21	17
⋆			20	10	26	17	10	17	27	12	18	17
Overall winner			MA-20	MA-19	MA19	MA-68	MA-20	MA-20	MA-20	MA-67	MA-68	MA-68

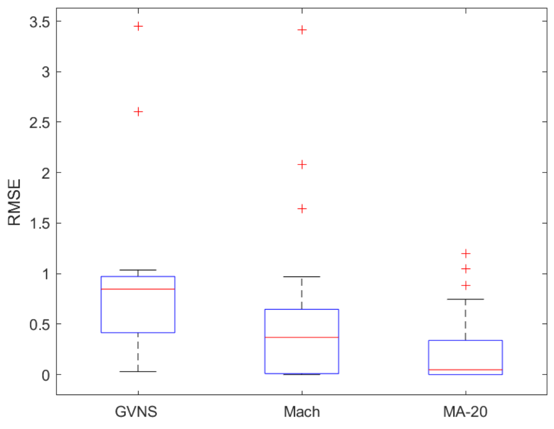
4.5 The best performing MA and the state-of-the-art algorithms

After comparing the MA in a general way and the top-five MA in more detail, in this section we discuss how MA-20 compares with the algorithms form the problem's literature described in Section 2.3.2. For the experiments with GVNS we established 900 seconds as stop criterion. The case of Mach was a little different. To be able to compare actual solution qualities, this constructive heuristic was allowed to run until complete solutions were generated. Similarly to the MA, GVNS and Mach were executed 50 times in the same computing platform described in Appendix A.4.

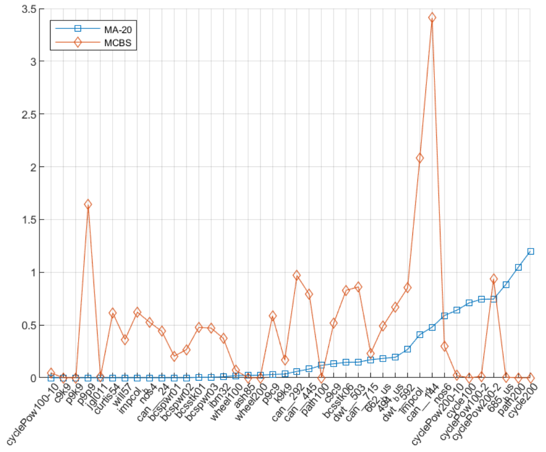
Figure 4.6 compares the performance of MA-20, GVNS and Mach, presenting the RMSE distributions for the three algorithms, and for more focus in the two closest competitors, the instance by instance RMSE values. Table 4.6 lists the instances, their size, order and upper bound/optimal value, as well as the specific solution costs that Mach and MA-20 were able to obtain, the corresponding average and standard deviation and the average execution time. It also includes the pair-wise comparison between Mach and MA-20.

In the comparisons of this section, GVNS is by far the worst performing algorithm, it has a large execution time that is not justified by the quality of its solutions, which correspond to the largest O-RMSE and very high outliers. On the other extreme, MA-20 was able to consistently achieve better solutions than any of the other two algorithms for the selected benchmark instances. Comparing the O-RMSE values, we found that MA-20 reduces the O-RMSE of Mach by more than 50%. For around a half of the instances, MA-20 provides the best solutions in almost all cases, hence its close to zero median and 75% of RMSE values under the RMSE median of Mach.

In Figure 4.6(b) we present the RMSE values for each instance in ascending order for MA-20, with the purpose of examining the cases where MA-20 is more or less successful and how that compares to Mach results. MA-20 gets statistically better results than Mach in 27 instances, and similar for



(a) RMSE distributions



(b) RMSE values per instance for MA-20 and MACH.

Figure 4.6: Comparing the performance in terms of RMSE for MA-20 and the algorithms from the literature, GVNS and MACH.

another 7, leaving just 6 instances where MACH has better results. Those instances belong to the topologies powers of cycles, cycles and path, which are highly regular topologies with low densities. MA-20 is instead consistently better on larger graphs. We further explore the differences on topology difficulty in Chapter 5, but here we can say that the instances such as paths and cycles are a curios case. An optimal embedding can be constructed very easily by means of a DFS enumeration of guest vertices, but they result oddly difficult for search algorithms as the SD, ILS and our MA. This is the reason why MACH can solve that type of instances very quickly, since the first phase of the MACH heuristic is a DFS with a priority criterion [47]. However, that design approach ends up catering to specific graphs and causing MACH to have larger performance variations across topologies. In contrast, MA-20 is a demonstrably more robust algorithm that experiments less performance variation across graph topologies, even for the instances that are less suitable for it.

4.6 Conclusions

In this chapter we expose the research work on the use of our alternative fitness function on different search methods, in this case memetic algorithms under different configurations of genetic operators.

Table 4.6: Comparison analysis of the performance of MA-20 and MACH.

				UB/	MACH				MA-20				MA-20
Graph	$\|V\|$	$\|E\|$	*d*	*Opt**	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	/MACH
path100	100	99	0.02	99*	99	99	0	0	99	105.5	9.75	61.63	−
path200	200	199	0.01	199*	199	199	0	0.01	352	404.7	31.34	124.70	−
cycle100	100	100	0.02	100*	100	100	0	0	100	140.6	58.69	23.43	−
cycle200	200	200	0.01	200*	200	200	0	0.01	362	424.64	85.51	100.06	−
wheel100	100	198	0.04	2600*	2600	2600	0	0.01	2600	2635.6	46.57	38.82	−
wheel200	200	398	0.02	10200*	10200	10200	0	0.07	10368	10436.4	94.88	155.83	−
cyclePow100-10	100	200	0.04	300*	5598	5711.68	67.76	0.04	300	458	159.32	2.74	⋆
cyclePow100-2	200	400	0.02	5500*	300	302.72	2.31	0	600	997.28	354.87	70.13	−
cyclePow200-10	100	1000	0.202	600*	11042	11200.64	76.52	0.11	5500	5500	0.00	30.27	+
cyclePow200-2	200	2000	0.101	11000*	600	602.4	2.21	0.01	11000	15767.4	5221.62	3.65	⋆
c9c9	81	162	0.05	873	991	1269.22	130.71	0.01	873	965.8	70.31	0.60	+
c9k9	81	405	0.125	7434	1809	1809	0	0.01	1809	1809	0.00	7.11	⋆
k9k9	81	648	0.2	8370	9424	9541.1	53.74	0.02	8280	8444.48	242.20	3.45	+
p9c9	81	153	0.047	7434	794	794	0	0	745	751.02	14.21	1.59	+
p9k9	81	396	0.122	7362	1728	1728	0	0.01	1728	1728	0.00	6.37	⋆
p9p9	81	144	0.044	720	944	1268.1	162.45	0	516	516	0.00	14.52	+
jgl011	11	49	0.891	4708	142	142	0	0	141	141	0.00	0.00	+
ash85	85	219	0.061	1705	1214	1395.34	126.53	0.14	913	931.92	21.56	12.85	+
curtis54	54	124	0.087	743	448	622.54	86.12	0.03	411	411	0.00	8.40	+
ibm32	32	90	0.181	1841	493	539.72	23.31	0.01	405	406.68	3.20	3.45	+
will57	57	127	0.08	4215	408	433.32	43.86	0.04	335	335	0.00	4.12	+
impcol_b	59	281	0.164	134935	2462	2841.06	214.84	0.07	1822	1822	0.00	0.24	+
impcol_d	425	1267	0.014	6237	23995	35083.34	5557.16	13.46	13011	16567.46	2420.23	143.50	+
nos4	100	247	0.05	218010	1181	1448.04	264.25	0.06	1031	1031	0.00	2.30	+
nos6	675	1290	0.006	72517	35658	48573.8	4660.16	2.12	15600	17396.2	1226.55	194.59	+
494_bus	494	586	0.005	150169	4873	5701.84	410.44	9.95	5160	5600.6	220.86	130.66	⋆
662_bus	662	906	0.004	219863	12956	15354.5	1319.83	31.75	11176	12589.84	719.90	200.17	+
685_bus	685	1282	0.005	425	14300	17723.38	2083.32	37.46	15541	16849.06	658.83	182.57	+
can_24	24	68	0.246	20881	216	253.06	15.55	0.01	182	182	0.00	0.12	+
can_144	144	576	0.056	82333	2250	2258.94	6.11	0.02	1776	2516.32	748.99	8.42	⋆
can_292	292	1124	0.026	187543	23288	25903.86	1731.32	7.1	15127	15781.26	1082.28	119.26	+
can_445	445	1682	0.017	532525	41259	51235.26	4795.67	18.74	27865	28903.22	498.09	150.33	+
can_715	715	2975	0.012	460	91646	109431.58	10123.94	79.88	67316	70951.26	2001.58	201.39	+
bcspwr01	39	46	0.062	737	101	114.48	8.08	0.01	98	98	0.00	0.72	+
bcspwr02	49	59	0.05	5325	158	179.4	19.47	0.02	148	148	0.00	1.08	+
bcspwr03	118	179	0.026	2156	766	927.02	70.56	0.26	662	664.92	1.19	102.37	+
bcsstk01	48	176	0.156	391532	1147	1336.48	107.99	0.02	936	938.32	3.62	23.50	+
bcsstk06	420	3720	0.042	391532	65017	83263.24	7489.16	30.87	53534	60387.2	5917.65	240.32	+
dwt_503	503	2762	0.022	348012	55067	67128.5	6003.72	32.69	40617	43240.72	3061.27	174.60	+
dwt_592	592	2256	0.013	334452	33659	44864.18	4826.16	44.03	30272	32162.52	1798.42	196.89	+
O-RMSE					0.22%				0.49%				
Avg time					7.72%				68.66%				
+													27
−													7
⋆													6
Overall winner													MA-20

We introduced a complete factorial study on the combination of four types of selection, two crossover operators, three types of mutation and two survival strategies. The choosing of these operators was oriented to cover a variety of approaches for inheriting the genes of promising individuals, maintaining a diverse and fit population, preventing premature convergence and accelerating the computations. The four selection operators cover the use of directly proportional fitness based preferences, the absence of said preferences, and a more limited scope for them and how these alternatives combine with the survival strategies that focus on fitness or disregard it. The crossover alternatives allow us to generate feasible and easy to evaluate solutions, as well as to address the effect of implicit mutations. We explored the introduction of fitness considerations within the mutation phase as a way of smart diversification and paired it with a non intensive local search approach as a mean to prevent premature convergence by allowing the evolutionary process to explore around local optima before directly introducing them in the population.

Our main findings were that the combination of survival and selection was the factor that affected the performance the most. While the $(\mu + \lambda)$ MA versions were on average better, the use of binary tournament revert this tendency by allowing MA using survival (μ, λ) to achieve better solutions in less time, specially when cyclic crossover was also present to prevent implicit mutations. These three operators (binary-tournament selection, cyclic crossover and survival (μ, λ)) were consistently present in the top five best performing MA, together with insertion and reduced 3-swap mutation.

Regarding the use of function f_3 for an alternative fitness scheme, it was observed that it had a polarizing effect, typically helping MA that were already good under the CBS function to achieve even better results, but also aggravating the problems of MA that were already poor performing. Even when the highest average O-RMSE rank reached by an algorithm using function f_3 was the third place, that algorithm (MA-68) was able to provide equal or significant better results than the first ranked algorithm (MA-19) in most of the instances when considered on an individual basis. We also identify the fist-improvement strategy in our local search as the potential cause limiting the helpfulness of function f_3.

While we choose MA-20 as a compromise between O-RMSE evaluation, statistical significance and execution time, we remark that the top three MA (MA-19, MA-20 and MA-68) had very similar results and therefore any of them is able to provide better results than the Mach algorithm for most of the instances. To a lesser extent, the fourth and fifth ranks (MA-67 and MA-43) are also able to do so. Overall, MA-20 achieved statically better results than Mach for 27 graphs and equal for another 7, out of the total 40. The 6 remaining cases were identified as particular topologies inherently easy for Mach, but not for search methods such as SD, ILS or our MA.

Although the full factorial experimental design employed when comparing MA versions was informative, it demanded high amounts of both time and computing power. Outstanding results were achieved by a particular set of operators, but as it was discussed in the previous section, the success of the best MA is less affected by graph topology in comparison with Mach, but it is still not completely independent. Operators and combination of operators outside the top five ranking, such as the cumulative swap mutation, can offer particular strengths on different phases of the search, so it is worth exploring the use of a hyperheuristic approach to automatically vary operators as a way of improving robustness under varying topologies by combining the advantages of the operators. The next chapter introduces our implementation of the Dynamic Multi-Armed Bandit [25, 32] approach to automatize the selection of operators during the execution of the MA.

5 Dynamic Multi-Arm Bandit

5.1 Introduction

One of the challenges often encountered when trying metaheuristic approaches for an optimization problem is that there is a variety of alternatives for the components of the technique in question. Some cases to exemplify this are the genetic operators for evolutionary computing, the neighborhood definitions for local search approaches and the cooling schemes for simulated annealing. The nature of the problem can help reducing the scope for alternatives, since some techniques are intended for particular types of problems. For example, approaches focused in numerical optimization problems, such as PSO, may not be among the first options when dealing with a combinatorial problem. Also, operators that produce feasible solutions under a particular solution encoding may be preferred. This was the case for the permutation-based crossover and mutation operators employed in the MA discussed in Chapter 4. However, even under those cases, it is not always known before hand which operators are better suited for a particular problem or even a specific type of instance.

As it was observed in our experiments with the different operator configurations for the MA, the interactions between operators are both complex and determinant for performance, with some combinations being more successful over different instances or at different stages of the search process [110, 111]. For example, MA using the cumulative 2-swap are likely to do better at the beginning of the search, due to the fitness focus of this mutation operator, but it becomes rarer to come by improving mutations in late stages of the search and that produces stagnation. Something similar happens with MA using the survival strategy $(\mu + \lambda)$, which at first helps keeping a high fitness population, but ultimately contributes to slowing down the improvement rate, as happened with MA-43. It was also observed that, while our best MA is able to provide statistically significant improvements with respect to MACH and all other MA in the top five MA, there is still some variation among instances and its RMSE values are not always equal to zero. That indicates that there is some other algorithm that got a better solution at least once, so there is margin for further improvement with respect to the performance of MA-20. Besides, there exists the issue with the alternative evaluation function, which was regularly better than the CBS function for guiding the SD and ILS algorithms, but that was not always the case for the MA.

Instead of trying another metaheuristic or testing new operators, which would take extra time in design and development, the approach was to use what had been already developed for the MA in a new way. The idea was to access the combined strengths of the operators by using an Adaptive Operator Selecton (AOS) [33] hyperheuristic method for automatically changing the operator configuration during execution time. Hyperheuristics allow to gain generality regarding performance variations across instances [17] and even better results, by using low level heuristics in an adequate sequence, than what could be achieved by any of those low level heuristics if used independently.

The problem of selecting among the MA configurations during execution was modeled for a hyperheuristic based on the multi armed bandit (MAB). The MAB original formulation comes from the game-theory area [3, 65], being described in terms of casino bandit machines. The bandit machines have arms, each with an associated probability for yielding an all-or-nothing reward. The objective is to pick an arm to play, so that the accumulated reward over time can be maximized. In practice, the rewards are not restricted to all-or-nothing and the distribution of reward probabilities is unknown, so the MAB approaches [3] use statistical information about the past rewards obtained by the arms for calculating estimations of reward probabilities, then play the arm more likely to give the best reward and the actual result becomes feedback for future estimations. One can see how the MAB relates to the AOS, by equating the bandit arms with the operators or algorithms to select. The approach we implemented is a MAB dynamic variant that was designed for scenarios where the reward probabilities change over time, as we considered that this is the case for our MA. The dynamic multi-armed bandit (DMAB) [25] recognizes those changes and it can adjust the data for reward estimations accordingly.

This chapter presents the implementation of the DMAB hyperheuristic and the results that were achieved when using the whole set of operators. To determine the helpfulness of the alternative evaluation in this scenario, it was also observed what happens if function f_3 is removed from the operator set. The results were compared with those obtained by the best performing MA configuration, MA-20, and by MACH, over to the same set of instances used in the previous chapter.

The rest of the chapter consists of the explanation of the DMAB functionality in Section 5.2, where its main components are introduced and its complexity is discussed. Section 5.3 presents the performance comparison with the best MA and the algorithms from the problem's literature. Section 5.4 examines how the interactions among operators are different when they are analized within the DMAB or in the individual MA; and how those differences reflect on their results. Finally, the chapter is closed in Section 5.5 by summarizing the main findings regarding the use of DMAB to choose among the available MA configurations.

5.2 The dynamic multi-armed bandit

MAB approaches rely on confidence estimations, meant to represent a notion of the underlying reward probabilities for the arms [3]. These estimations are calculated based on empirical reward evidence gathered from the past occasions where an arm was played. The rewards provide a measure of how much improvement was gained after playing an arm [27]. If an arm has been able to achieve high rewards in the past, then the confidence estimations will reflect it and the arm will have more chances of being played in the future. At the beginning of the process, there is no reward data available to estimate the confidence on the bandit's arms, therefore the first step is to collect it by playing all of them.

An issue with the MAB is that the reward probabilities may change over time and it could take significant time for the confidence estimations to adjust accordingly [9]. The inclusion of a dynamic component in the DMAB is a response to that issue, by incorporating a simple PH-Hinkley (PH-test) statistical test [53] to recognize when the reward obtained by the played arm is atypical. This is interpreted as a sign that the underlying reward probabilities have shifted, therefore the reward data from previous iterations is no longer useful for producing accurate confidence estimations. New reward data is then acquired by playing all the arms, as at the beginning of the DMAB process. As a result, the DMAB can adjust its confidence estimations faster, making it suitable for dynamic scenarios.

The DMAB was chosen because it requires little modifications over the existing implementation of the MA and it has relatively few components and parameters regarding the assignation of rewards, the strategy for estimating confidence, and the triggering of the PH-test.

For the DMAB implementation the set of K bandit's arms A represents the different MA by their configuration of operators, in the form $A_i = \{s, c, m, ss, f\}$. It was approached this way in order to manage alternatives for several MA components of different type. It was also known that the success of our MA is often more linked to operator interactions ratter than to a single operator.

Therefore, by equating an arm to an operator configuration, the rewards account for the effect of the whole MA and not just individual operators.

Population P was defined in the same way than it was done for the MA, employing the same encoding and initialization (see Section 4.2.1). When playing all arms for their initial rewards at the beginning of the process or after a PH-test triggering, the operations are carried on over population P', a copy of population P. This allows that the original population P remains unchanged, so that all arms can begin with the same initial solutions and the rewards are fairly compared.

In the following iterations, the rewards are used for calculating the confidence estimations. The arm with the best confidence, arm A_s, is played over population P, now actually affecting it in a permanent way. The obtained reward is used in the PH-test [53], to detect if there is an abrupt change with respect to previous rewards for arm A_s. If that is the case, there is a mechanism for restarting the process from scratch, setting the rewards, the confidence estimations, and the number of times the arms have been played to their original zero values.

The DMAB stops when it mets the stop criteria, reporting the best found solution g.

5.2.1 Reward assignation

The rewards capture the improvement that an arm was able to accomplish after been played, in this case we based the rewards on the CBS actual value for the best individual in population P. This focuses the evaluation of arms, and the MA they represent, on the true objective of our problem, which is minimizing the CBS.

This is expressed in Equation 5.1, where rr_i is the raw reward for arm A_i, and f_{old} and f_{new} are the CBS values for the best individual before and after playing the arm.

$$rr_i = \frac{f_{old} - f_{new}}{f_{old}} \times 100\,. \tag{5.1}$$

We refer to the value rr_i as a raw reward because it is not used directly, but within a more

Algorithm 2 DMAB Algorithm

```
1: $P \leftarrow$ initializePopulation($P$)
2: $g \leftarrow P_{best}$
3: Set confidence and number of times arms have been played to zero
4: for $i \leftarrow 1$ to $K$ do
5:    $P' \leftarrow P$
6:    $P' \leftarrow playArm(A_i, P')$
7:    $r_i \leftarrow$ initial reward for $A_i$
8: end for
9: repeat
10:    Compute confidence for all arms
11:    $A_s \leftarrow selectArm(r)$
12:    $P \leftarrow playArm(A_s, P)$
13:    $r_i \leftarrow$ reward for $A_i$
14:    if PH-test is triggered then
15:       Set confidence and number of times arms have been played to zero
16:       for $i \leftarrow 1$ to $K$ do
17:          $P' \leftarrow P$
18:          $P' \leftarrow playArm(A_i, P')$
19:          $r_i \leftarrow$ initial reward for $A_i$
20:       end for
21:    end if
22:    $g \leftarrow$ fitter individual among current $g$ and $P_{best}$
23: until stop criteria is met
24: return $g$
```

elaborated method for reward assignation. The extreme value-based reward [31] is an assignation scheme based on a notion of credit. It maintains a record of the raw rewards obtained by an arm the last WR times it was played, functioning as a memory of how good the arm has performed historically. Under this scheme, the arms receive the reward r_i as a credit equal to the best raw reward in the record.

$$r_i = \max_{j=1}^{WR}\{rr_j\}\,. \tag{5.2}$$

As part of the parameter tuning process, we tested several values for the length record WR, including $WR = 1$, which is equivalent to use the raw reward value directly.

5.2.2 Confidence estimations

The DMAB strategy for selecting an arm is a variant of the upper confidence bound (UCB1) [3]. It consists in assigning a higher confidence to the arms that have performed well, but sometimes giving preference to arms that have not been used very often. In this way UCB1 manages the exploitation and exploration of arms [25]. Its calculations are simple, requiring only the empirical reward, calculated as the average of previously received rewards for arm A_i and the number of times every arm has been used. The balance between the exploitation of reliable arms and the exploration of underused arms is adjusted by a parameter C. Equation 5.3 shows the confidence assignation by UCB1:

$$confidence_i = empRew_i + C\sqrt{\frac{2\log\sum_{j=1}^{k} plays_j}{plays_i}}\ , \tag{5.3}$$

where $empRew_i$ is the empirical reward for arm A_i (i.e., the average reward obtained by it), $plays_i$ is the number of times arm A_i has been played, and C is the parameter regulating the preference of trusted arms or underused ones.

5.2.3 Page-Hinkely test

The DMAB manages the detection of abrupt changes in the underlying reward probabilities by incorporating the Page-Hinkely test [25]. The test is employed to determine if the reward assigned to an arm is significantly different from the previous rewards attained by it. An arm getting an atypical reward is a sign that the confidence estimations are lacking accuracy. The PH-test examines how the standard deviation of reward values for arm A_i behaves through time, triggering if at time t the difference between the maximal standard deviation $maxDev_{i,t}$ and its average $avgDev_{i,t}$ is beyond a certain threshold. The test has two parameters, δ and λ. The first, δ, has the purpose of helping to deal with scenarios where the underlying reward probabilities change slowly, by adding a small shift on the average standard deviation, as shown in Equation 5.4, where $empRew_i$ is the empirical reward equal to the average of rewards for A_i and r_i is the current reward.

$$avgDev_{i,t} = avgDev_{i,t-1} + (empRew_i - r_i + \delta) \tag{5.4}$$

Then, the maximal standard deviation for arm A_i's rewards at time t is calculated according to Equation 5.5.

$$maxDev_{i,t} = \max(avgDev_{i,t}, maxDev_{i,t-1}) \tag{5.5}$$

The second parameter, λ, is the threshold for the test triggering if:

$$maxDev_{i,t} - avgDev_{i,t} > \lambda\,.$$

5.2.4 DMAB complexity

Updating the confidence estimation for an arm A_i that has just been played requires some information: the reward r_i, the average of the past rewards $empRew_i$, the number of plays $plays_i$ for the arm and the total plays across all arms. All these values can be calculated or read from memory in linear or constant time as follows. The raw reward rr_i is calculated in constant time, then the reward register for the extreme value-based reward is updated in linear time using a queue and the maximal reward value r_i can be determined in WR steps. Then, the average reward for the arm is updated in constant time as $emphiricalRew_i = (plays_i - 1(emphiricalRew_i) + r_i)/plays_i$. Notice that its initial value is set as $emphiricalRew_i = 0$. The number of plays per arm $plays_i$ and the total number of plays are kept in memory, so those values are also updated and accessed in constant time. Therefore, the confidence estimation for the last arm played has a linear complexity $\mathcal{O}(WR)$ with respect to the maximum length of the reward history WR.

Determining if the PH-test is triggered is also easily done in constant time, following the expressions in Equations 5.4 and 5.5, which require as input the current reward r_i and the average reward $empRew_i$ in order to update the average and maximal standard deviation of the rewards for arm A_i.

As demonstrated, the DMAB itself is very efficient in the operations for reward assignation, confidence estimation and the PH-test triggering. The complexity will then depend more on how complex are the algorithms implemented as the arms. It was established that all the MA have in common the cubic term for the incremental evaluation of edges affected during local search and a quadratic term for the fitness function (see Section 4.3.7), therefore playing any arm has a worst case complexity of $\mathcal{O}(n^3 + \mu n^2)$.

During the initialization, all K arms are played over a copy P' of the initial population P and assigned its first rewards and confidence estimations. The complexity of the arm's initialization is the accumulated complexity of all the MA, plus the steps for the first confidence estimation and the creation of population P'. Since each arm has only being played once, the reward is assigned in constant time independently from the length of the reward registry WR, the confidence estimation is performed in constant time, and the copy process of population P into P' is linear, taking μn steps. Therefore the complexity of playing all K arms would be $\mathcal{O}(K(n^3 + \mu n^2 + \mu n)) \approx \mathcal{O}(K(n^3 + \mu n^2))$. This complexity also applies for the arm restarting process that occurs after the PH-test is triggered.

For the worst case, we could assume that after playing an arm the PH-test triggers, resulting in a complexity of $\mathcal{O}(T(K+1)(n^3 + \mu n^2))$, where T stands for the total number of iterations.

5.3 Performance of MA alternating operators with DMAB

The combination of the MA with the DMAB hyperheuristic was named as DMAB+MA. This section addresses the experiments to measure the performance of this algorithm and compares it with the previous MA and the algorithms from the CBSP literature [47, 116]. The results cast light on some of the issues we identified regarding the helpfulness of the alternative evaluation scheme in the MA, the role of the other MA operators and the design of DMAB+MA itself. The experiments took place in the same experimental platform and over the same instance set we employed for the MA, with 50 runs per instance and 600 seconds as stop criterion.

5.3.1 Parameter settings and the extreme value-based reward

The first remarkable results about DMAB+MA concerns the use of the extreme vale-based reward approach [31]. Under this rewarding system, the reward assigned to an arm is a credit based on the maximal raw reward value in a record of length WR. In the parameter tuning process for DMAB+MA, described in Table 5.1, we tested several values for the parameters of the MA and the DMAB. These included the values $\{1, 3, 5, 10, 15\}$ for the length of the extreme value-based reward's record. As shown in Table 5.1, the best value was $WR = 1$, which effectively shuts down the special functionality of the extreme value-based reward.

A large reward record can have an adverse effect on the confidence estimation, by preventing an arm with a recent good raw reward from being used, favoring instead an arm that was more successful in a window of WR iterations ago, but perhaps no longer. The extreme value-based reward may also be preventing the PH-test from recognizing the changes on underlying reward probabilities, since its reward information would proceed from some past iteration where the success of the arm was maybe very different. For example, large fitness improvements are more common to occur in the first iterations, and it is known that some of the operators, such as the cumulative swap mutation or the $(\mu + \lambda)$ survival strategy are helpful at the beginning of the search process, but they can cause stagnation later. With the extreme value-based reward the DMAB may be placing too much confidence on MA that achieved large rewards when doing so was relatively easier.

The DMAB relies on an appropriate placing of the confidence estimations and the evaluation of reward changes for managing arm usage. In the case of our implementation, keeping reward values updated seems to be more effective than giving credit to arms for their past good behaviour.

Table 5.1: Parameter settings for the DMAB+MA algorithm, chosen after 5000 tests with the irace utility [72].

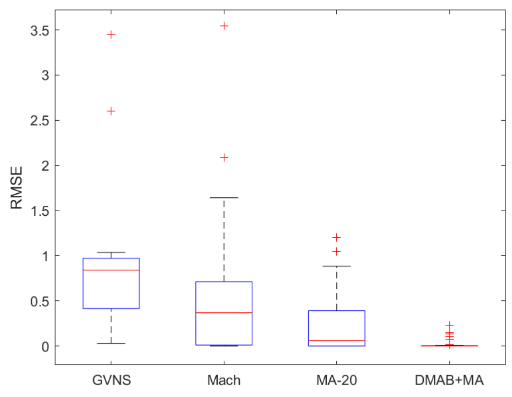
Parameter	Tested values	Final value	Parameter	Tested values	Final value
Crossover probability $prob_c$	$[0.01, 0.9]$	0.812	Length of the reward register WR	$\{1, 3, 5, 10, 15\}$	1
Mutation probability $prob_m$	$[0.01, 0.9]$	0.761	PH-test tolerance δ	$[0.05, 0.5]$	0.299
Inversion probability $prob_i$	$[0.01, 0.9]$	0.012	PH-test triggering threshold λ	$[15, 35]$	33.472
Scaling factor C	$[1, 10]$	7.138	Max. running time for DMAB s	–	600s

5.3.2 DMAB+MA for improving performance and generality

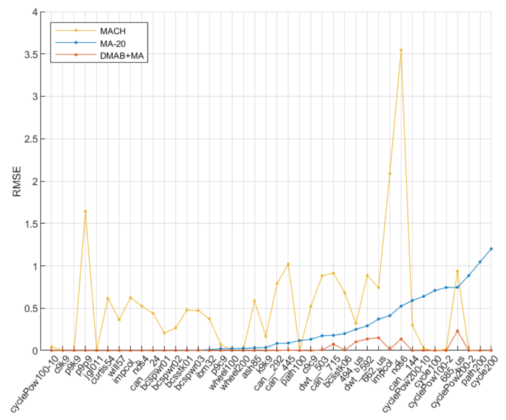
The main expectation for the use of DMAB with our MA is that, if there is a MA that can get a good solution for a certain instance at least once, then the DMAB+MA, having all operators available, should be able to produce comparable or even better solutions in a more consistent fashion. The examination of results for Mach and MA-20 in Chapter 3 showed that the performance of these algorithms was influenced by the guest graph topology. Because of their respective designs, the topologies that result struggling for them differ. Mach uses a depth-first style search, so it solves paths, cycles an wheels very quickly but then it struggles with the rest of instances. Meanwhile, MA-20 and the rest of MA, were designed with a general case approach. MA-20 is able to produce better results for graphs where Mach can not, but it is not as accurate reaching optimal solutions for paths, cycles and wheels.

The results for MA-20, namely the best and average CBS and the corresponding standard deviation, shown that the algorithm was able to find good solutions for the path, cycle or wheel graphs in some executions, but it lacked the consistency to do it more often. For example, for the path, cycle and wheel of order $n = 100$, MA-20 occasionally gets optimal cost solutions as reflected by the average and standard deviation columns in Table 5.3. It is also noticeable that for the aforementioned topologies, there were other MA with better RMSE rates than MA-20, such as MA-43. Since the RMSE values for MA-20 are not always zero, there are opportunities for further improvement of its results. Therefore, the DMAB+MA was proposed as a method combining strengths from the different MA in order to achieve optimal/best-known solutions more consistently and for a broader variety of graph topologies than any of the independent MA.

The evidence from the experiments suggests that incorporating the DMAB to manage the operator configurations of our MA resulted very effective for reducing the O-RMSE, lowering the solution cost, and sometimes also decreasing the amount of time required to reach the new best-known solutions. For example, out of the 30 instances for which some MA got a better solution than MACH, DMAB+MA achieved 10 further improvements. Moreover, DMAB+MA was able to produce the new 30 best-found results and the 10 known optimal results more often than any of the other considered methods. This is demonstrated by the global O-RMSE values in Table 5.2, with the O-RMSE of DMAB+MA being over 10 times smaller than the one of MA-20 and 24 times smaller than the one for MACH. Notice that, since the O-RMSE metric is relative to the best-known results, after DMAB+MA improved those results, the O-RMSE values for GVNS, MACH and MA-20 worsened accordingly.



(a) RMSE distributions.



(b) RMSE by instance for the three most relevant algorithms.

Figure 5.1: Distributions of RMSE for GVNS, MACH, MA-20 and DMAB+MA, and RMSE by instance for the three best performing algorithms.

The behaviour of MACH, MA-20 and DMAB+MA at the instance level is presented in Figure 5.1, contrasting their corresponding RMSE values sorted with respect to MA-20. Such sorting allows to examine if DMAB+MA can perform better on the instances where it was identified that MA-

Table 5.2: Results comparing DMAB+MA with MA-20 and the algorithms form the literature in terms of solution quality (0-RMSE) and running time.

Algorithm	O-RMSE (%)	Average execution time (s)	Time to best-found sol. (s)
DMAB+MA	0.021	600.0	128.7
MA-20	0.238	193.3	87.1
Mach	0.513	7.7	
GVNS	0.775	900.1	612.380

20 had issues, specifically, the paths, cycles and wheel topologies. It was found that in all the instances where either MA-20 or Mach had the ideal performance regarding solution cost, that being producing a best-know or optimal cost solution in all executions, DMAB+MA also succeeded. For most of the instances, MA-20 had already significantly improved Mach's results, as summarized in the statistical significance analysis in Table 5.4. DMAB+MA fulfills the purpose of lowering the O-RMSE values while also achieving the optimal solutions for the instances MA-20 could not. In the statistical comparison analysis, DMAB+MA defeats both MA-20 and Mach for most instances, the exceptions being ties mostly for graphs were the optimal had been reached.

While MA-20 was bested by DMAB+MA, it is worth to notice it is still useful as a reference metaheuristic, since Mach is only a constructive heuristic at an obvious disadvantage against metaheuristic approaches in terms of providing better solution quality. This is specially relevant when dealing with larger graphs from topologies such as the Harwell-Boeing, Cartesian products and random graphs.

5.4 Operator usage

This section discusses in more detail the inner mechanics of the DMAB+MA and how they result helpful for conducting the search process. It specifically focuses on the interaction among operators, how this relates to tendencies observed in the independent MA versions, and the role played by the alternative fitness function.

The average operator usage rate was examined in Figure 5.2. These graphics were created by

Table 5.3: Detailed performance comparison of MACH, MA-20, and DMAB+MA.

					MACH				MA-20 (S4_C1_M2_SS1_V0)				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	d	UB/Opt^*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
path100	100	99	0.02	99*	99	99	0	0	99	105.5	9.75	61.631	99	99	0.00	10.1206
path200	200	199	0.01	199*	199	199	0	0.01	352	404.7	31.34	124.7024	199	199	0.00	85.356
cycle100	100	100	0.02	100*	100	100	0	0	100	140.6	58.69	23.4264	100	100	0.00	14.8906
cycle200	200	200	0.01	200*	200	200	0	0.01	362	424.64	85.51	100.0636	200	200	0.00	54.4894
wheel100	100	198	0.04	2600*	2600	2600	0	0.01	2600	2635.6	46.57	38.818	2600	2600	0.00	2.5844
wheel200	200	398	0.02	10200*	10200	10200	0	0.07	10368	10436.4	94.88	155.8296	10200	10200	0.00	13.4392
cyclePow100-2	100	200	0.04	300*	300	302.72	2.31	0	300	458	159.32	2.7418	300	300	0.00	24.8558
cyclePow200-2	200	400	0.02	600*	600	602.4	2.21	0.01	600	997.28	354.87	70.1316	600	600	0.00	63.4302
cyclePow100-10	100	1000	0.202	5500*	5598	5711.68	67.76	0.04	5500	5500	0.00	30.2684	5500	5500	0.00	0.382
cyclePow200-10	200	2000	0.101	11000*	11042	11200.64	76.52	0.11	11000	15767.4	5221.62	3.6536	11000	11000	0.00	6.0316
c9c9	81	162	0.05	873	991	1269.22	130.71	0.01	873	965.8	70.31	0.5998	873	873	0.00	137.9754
c9k9	81	405	0.125	7434	1809	1809	0	0.01	1809	1809	0.00	7.113	1809	1809	0.00	23.4992
k9k9	81	648	0.2	8370	9424	9541.1	53.74	0.02	8280	8444.48	242.20	3.4542	8280	8280	0.00	294.0756
p9c9	81	153	0.047	7434	794	794	0	0	745	751.02	14.21	1.5936	745	745	0.00	302.1566
p9k9	81	396	0.122	7362	1728	1728	0	0.01	1728	1728	0.00	6.3714	1728	1728	0.00	53.2098
p9p9	81	144	0.044	720	944	1268.1	162.45	0	516	516	0.00	14.5194	516	516	0.00	41.559
jgl011	11	49	0.891	147	142	142	0	0	141	141	0.00	0	141	141	0.00	0.0002
ash85	85	219	0.061	4708	1214	1395.34	126.53	0.14	913	931.92	21.56	12.851	913	913	0.00	45.449
curtis54	54	124	0.087	1705	448	622.54	86.12	0.03	411	411	0.00	8.399	411	411	0.00	52.4954
ibm32	32	90	0.181	743	493	539.72	23.31	0.01	405	406.68	3.20	3.4466	405	405	0.00	0.2088
will57	57	127	0.08	1841	408	433.32	43.86	0.04	335	335	0.00	4.116	335	335	0.00	10.6124
impcol_b	59	281	0.164	4215	2462	2841.06	214.84	0.07	1822	1822	0.00	0.237	1822	1822	0.00	0.62
impcol_d	425	1267	0.014	134935	23995	35083.34	5557.16	13.46	13011	16567.46	2420.23	143.5012	12170	12277.52	169.45	330.8104
nos4	100	247	0.05	6237	1181	1448.04	264.25	0.06	1031	1031	0.00	2.2972	1031	1031	0.00	44.6782
nos6	675	1290	0.006	218010	35658	48573.8	4660.16	2.12	15600	17396.2	1226.55	194.5918	11496	12823.08	824.31	324.0696
494_bus	494	586	0.005	72517	4873	5701.84	410.44	9.95	5160	5600.6	220.86	130.6622	4496	4914.16	198.84	326.8424
662_bus	662	906	0.004	150169	12956	15354.5	1319.83	31.75	11176	12589.84	719.90	200.17	9238	10485.3	613.80	295.0838
685_bus	685	1282	0.005	219863	14300	17723.38	2083.32	37.46	15541	16849.06	658.83	182.5688	10254	11814.68	625.54	293.4828
can_24	24	68	0.246	425	216	253.06	15.55	0.01	182	182	0.00	0.1246	182	182	0.00	0.0302
can_144	144	576	0.056	20881	2250	2258.94	6.11	0.02	1776	2516.32	748.99	8.4198	1776	1776	0.00	44.3344
can_292	292	1124	0.026	82333	23288	25903.86	1731.32	7.1	15127	15781.26	1082.28	119.2576	15109	15125.9	7.04	284.8162
can_445	445	1682	0.017	187543	41259	51235.26	4795.67	18.74	27865	28903.22	498.09	150.3298	26634	26767.76	53.35	387.8466
can_715	715	2975	0.012	532525	91646	109431.58	10123.94	79.88	67316	70951.26	2001.58	201.3898	60349	64689.94	1392.98	309.81
bcspwr01	39	46	0.062	460	101	114.48	8.08	0.01	98	98	0.00	0.7214	98	98	0.00	2.6946
bcspwr02	49	59	0.05	737	158	179.4	19.47	0.02	148	148	0.00	1.0796	148	148	0.00	3.5354
bcspwr03	118	179	0.026	5325	766	927.02	70.56	0.26	662	664.92	1.19	102.3684	662	663.26	0.85	295.9958
bcsstk01	48	176	0.156	2156	1147	1336.48	107.99	0.02	936	938.32	3.62	23.4962	936	936	0.00	4.6114
bcsstk06	420	3720	0.042	391532	65017	83263.24	7489.16	30.87	53534	60387.2	5917.65	240.3212	51830	52085.5	152.42	292.7646
dwt_503	503	2762	0.022	348012	55067	67128.5	6003.72	32.69	40617	43240.72	3061.27	174.5952	37448	37703.96	149.62	378.481
dwt_592	592	2256	0.013	334452	33659	44864.18	4826.16	44.03	30272	32162.52	1798.42	196.8884	25076	27678.24	2254.86	294.4532

Table 5.4: Statistical analysis for comparing the performance MACH, the MA-20 y DMAB+MA.

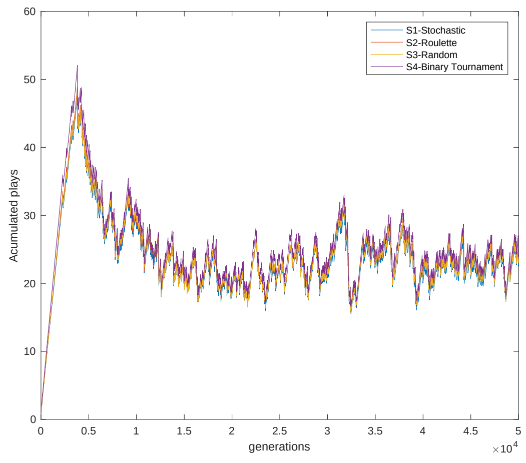
Instances	$\|V\|$	$\|E\|$	MA-20/ MACH	MA-20/ DMAB+MA	MACH/ DMAB+MA
path100	100	99	−	−	$\star$
path200	200	199	−	−	$\star$
cycle100	100	100	−	−	$\star$
cycle200	200	200	−	−	$\star$
wheel100	100	198	−	−	$\star$
wheel200	200	398	−	−	$\star$
cyclePow100-2	100	200	$\star$	−	−
cyclePow200-2	200	400	−	−	−
cyclePow100-10	100	1000	+	$\star$	−
cyclePow200-10	200	2000	$\star$	−	−
c9c9	81	162	+	−	−
c9k9	81	405	$\star$	$\star$	$\star$
k9k9	81	648	+	−	−
p9c9	81	153	+	−	−
p9k9	81	396	$\star$	$\star$	$\star$
p9p9	81	144	+	$\star$	−
jgl011	11	49	+	$\star$	−
ash85	85	219	+	−	−
curtis54	54	124	+	$\star$	−
ibm32	32	90	+	−	−
will57	57	127	+	$\star$	−
impcol_b	59	281	+	$\star$	−
impcol_d	425	1267	+	−	−
nos4	100	247	+	$\star$	−
nos6	675	1290	+	−	−
494_bus	494	586	$\star$	−	−
662_bus	662	906	+	−	−
685_bus	685	1282	+	−	−
can_24	24	68	+	$\star$	−
can_144	144	576	$\star$	−	−
can_292	292	1124	+	−	−
can_445	445	1682	+	−	−
can_715	715	2975	+	−	−
bcspwr01	39	46	+	$\star$	−
bcspwr02	49	59	+	$\star$	−
bcspwr03	118	179	+	−	−
bcsstk01	48	176	+	−	−
bcsstk06	420	3720	+	−	−
dwt_503	503	2762	+	−	−
dwt_592	592	2256	+	−	−
+			27	0	0
−			7	28	32
$\star$			6	12	8
Overall winner			MA-20	DMAB+MA	DMAB+MA

recording which arm was being used at a given iteration and the operators that were involved. Then, it was calculated an average of the accumulated number of times a MA employing a particular operator had been used across the first 50,000 iterations for all the runs. The resulting graphics also allow to visualize the frequency of the arm's restarts caused by the PH-test being triggered. Recall that when the PH-test triggers the statistical information about the arms, including the number of times that they have been played is set back to its initial zero values. Since these plots depict average values across executions, the restarts are denoted by the drops in the accumulated number of plays.

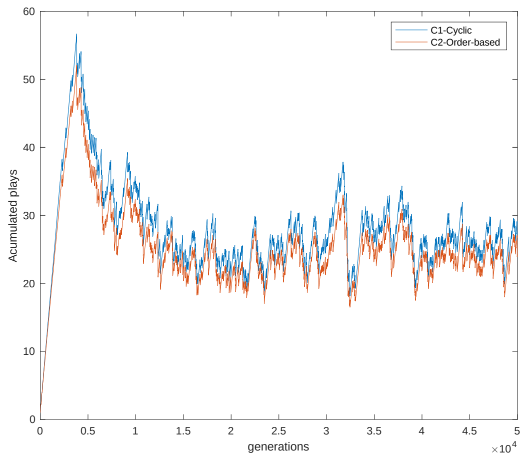
According to Figure 5.2, the most frequently employed operators are binary tournament selection, cyclic crossover, cumulative 2-swap mutation and (μ, λ) survival. Meanwhile, the fitness functions are employed with very similar frequency. The operator preferences observed in DMAB+MA match several of the tendencies for the most often employed operators among the top five MA configurations. Binary tournament selection and cyclic crossover were present in all top five MA, while the alternative evaluation function function was employed by two of them. However, there are also interesting differences, for which operators not often found in the top five MA are frequently used in the DMAB+MA.

These correspond to mutation and survival: cumulative swap mutation was not present in the top five MA, and the $(\mu + \lambda)$ survival strategy was employed only by MA-43, which performed significantly worse than the rest of the algorithms in the top five MA group. It can be argued that the new interactions created among the full set of operators available in the DMAB+MA are responsible for these differences, and ultimately, for the DMAB+MA's ability to produce better solution quality than the independent MA.

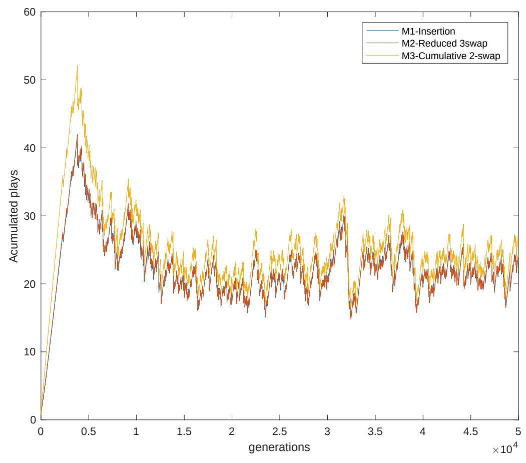
For the case of mutation, it was previously established that the fitness restrictiveness of the cumulative 2-swap mutation can help the search in early stages, when it is relatively easier to find improving mutations; but this becomes harder in later generations. Since non improving mutations will not be accepted by the cumulative 2-swap mutation, this operator can hinder the algorithm's ability to explore new areas and result in a worst performance. The independent MA with lowest



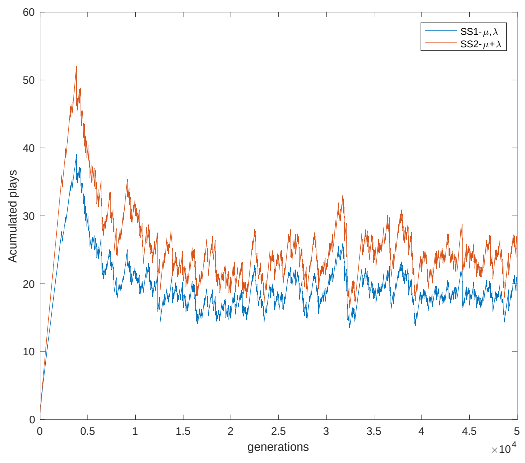
(a) Selection operators.



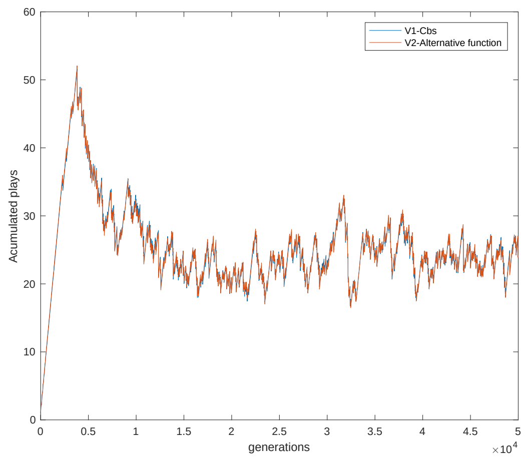
(b) Crossover operators.



(c) Mutation operators.



(d) Survival strategies.



(e) Evaluation functions.

Figure 5.2: Average number of times that arms employing particular operators were played along 50,000 generations.

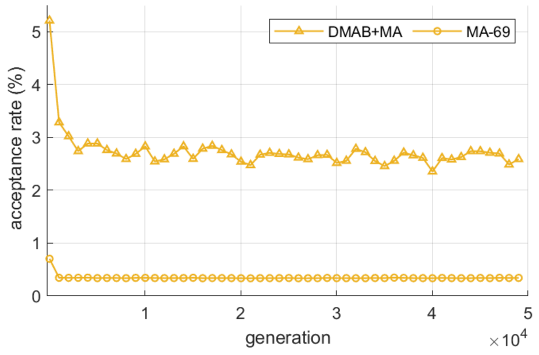
O-RMSE that used cumulative 2-swap mutation was MA-69 (O-RMSE = 0.520). It is worth noting that MA-69 differs from the algorithm in the fourth position of the top five MA, MA-67 (ORMSE= 0.2877), only in the mutation operator, yet MA-69 ranked in the position 21. Therefore, by using MA-67 as reference, the cumulative 2-swap mutation can be identified as the main cause of MA-69's poor performance.

The behaviour of the cumulative 2-swap mutation in MA-69 and DMAB+MA is showcased in Figure 5.3, comparing the acceptance rate for attempted mutation moves and the percentage change in CBS caused by affected mutations. Figure 5.3(a) demonstrates that the average acceptance rate for cumulative 2-swap mutations in MA-69 drops within the first 1000 generations for both algorithms, specially for MA-69.

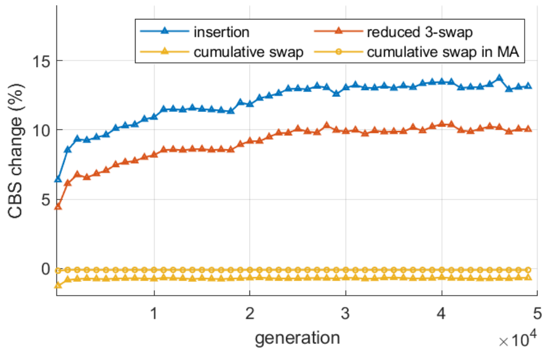
The fact that cumulative 2-swap is the preferred mutation operator in the DMAB+MA can be explained by its combination with the insertion and reduced 3-swap operators. Figure 5.3(b) compared the percentage change in CBS caused by mutations in MA-69 and the DMAB+MA. For the later, the three mutation operators are examined separately. The percentage of change in CBS was calculated as $(f_{new} - f_{old})/f_{old}$, where f_{old} and f_{new} are the CBS of the individuals before and after mutation. Negative values indicate a improvements, which reduce the CBS values.

It can be observed that the arms using the cumulative 2-swap operator can produce larger improvements during mutation in DMAB+MA than in MA-69. These improvements in fitness of mutated individuals can contribute to arms that use cumulative 2-swap receiving better rewards.

When the acceptance rate for cumulative 2-swap mutation drops, independent MA such as MA-69, loose part of their ability to explore new areas, preventing them from achieving better results when compared with MA using the other mutation operators. In contrast, in DMAB+MA, if the absence of mutation causes the fitness to stop improving, the arm's reward and confidence would be adjusted accordingly. Then, by switching arms, the DMAB+MA is capable of using insertion or reduced 3-swap mutation. In further generations, an arm using cumulative 2-swap can be employed again, potentially achieving new fitness improvements during mutation and a reward that result in it



(a) Mutation rate for cumulative swap.



(b) Change in CBS quality.

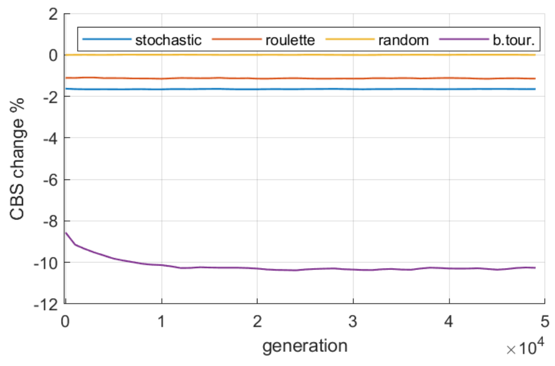
Figure 5.3: Comparison of the acceptance rate and change in CBS after mutation, for cumulative swap mutation in DMAB+MA and MA-69, which was the MA with smallest O-RMSE employing this operator, ranking in position 21.

being played again.

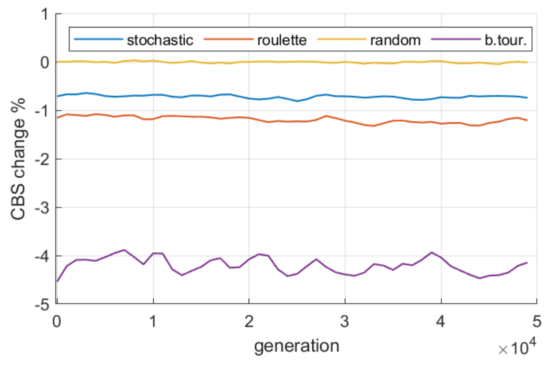
Another interesting interaction in the DMAB+MA can be identified for the survival strategy operators and their interaction with selection. As previously discussed, independent MA using the (μ, λ) survival strategy are not typical good performers, except when it is combined with binary tournament selection. These two operators appear together in the first four positions of the top five MA. The only MA in the top five that used the $(\mu + \lambda)$ survival was proven to have a significantly worse performance than the rest of MA in the top five group. This was attributed to the $(\mu + \lambda)$ survival strategy creating too much evolutionary pressure. Meanwhile, in the DMAB+MA, $(\mu + \lambda)$ survival is employed more often than (μ, λ), while binary tournament is preferred over the other selection operators, but not by a large margin.

Whether in independent MA or DMAB+MA, the binary tournament operator is more effective than the other selection operators for choosing a set of individuals with better average fitness respect to the population, as it is shown in Figure 5.4. For independent MA, this was computed using the best ranked MA that employs each selection operator.

Figure 5.5 shows the fitness of the best individual in the population and the fitness standard deviation for MA-19, MA-43 and DMAB+MA. Recall MA-43 is the best independent MA using the



(a) Change in CBS in independent MA.



(b) Change in CBS in DMAB+MA.

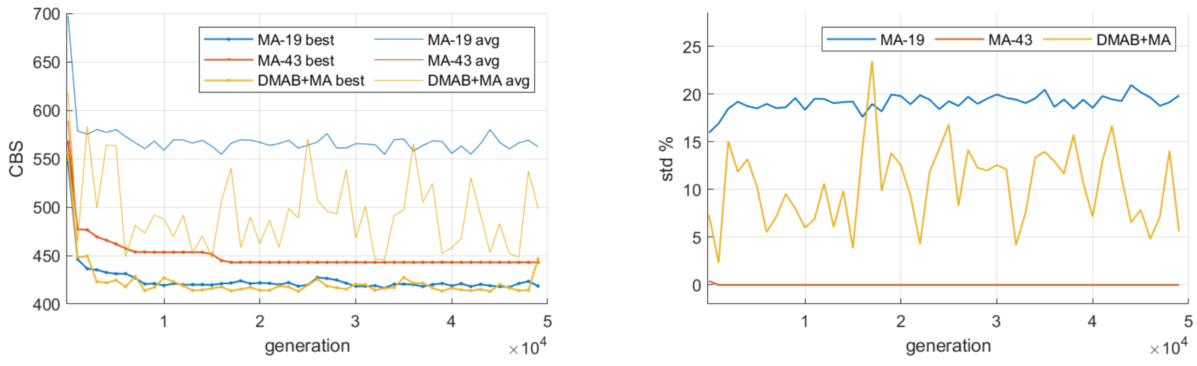
Figure 5.4: Change in average fitness of selected individuals with respect to the average fitness of the population, for independent MA and DMAB+MA. The independent MA correspond to those with the lowest O-RMSE MA that employ the corresponding selection operator.

$(\mu+\lambda)$ survival strategy and MA-19 has exactly the same operators, besides the survival. Comparing them to each other, and against DMAB+MA, provides insights on the effects of the approach for population updating. MA-43 does not allow individuals in the offspring population to survive, unless they are fitter than current individuals. This can cause the population to become too uniform in terms of fitness, as denoted by the equal to zero standard deviation of the population's fitness in Figure 5.5(a). Meanwhile, the algorithms that use the (μ,λ) survival, including MA-20 and the rest of the top five MA, can maintain populations of diverse fitness, but they rely on selection for preserving the genes of the fitter individuals.

Differently from independent MA, when the DMAB+MA changes arms, the effects of the two survival approaches over the population can combine to maintain a high fitness population and allow the incorporation of new genetic material.

5.4.1 The role of the alternative evaluation function in DMAB+MA

To close this discussion in the use of operators, we look at the fitness functions. In the MA it was found that function f_3 helps to achieve better results on many occasions, but can also slow down the local search phase by considering neutral CBS neighbours as the first improvement, as discussed



(a) Average and best fitness.

(b) Standard deviation for population fitness.

Figure 5.5: Fitness across generation for independent MA using each of the survival operators and the DMAB+MA, where both are available, for instance *curtis_54*. MA-43 is the MA with smallest O-RMSE that employs the $(\mu + \lambda)$ survival. MA-19 employs instead the (μ, λ) operator and the rest of its operator configuration is identical to that of MA-43: binary tournament selection, cyclic crossover, insertion mutation and the CBS function for fitness evaluation. MA-43 occupies the 1st ranked position in the top five MA, while MA-43 ranked 5th.

in Section 4.4.1. When it comes to the DMAB+MA, the fitness function is the operator for which there is the less notorious preference. In order to have more concluding evidence on the helpfulness of function f_3 in the DMAB+MA, an experiment was performed to evaluate the performance of this algorithm when the alternative evaluation function is absent. All the other operators were still available, for a reduced total of 48 arms out of the original 96. The stop criteria for this experiment remained as 600 seconds, as well as the size of the population $\mu = 20$. The rest of the parameters for this reduced version of DMAB+MA were tuned once more, using the same previously described methodology. Most of the assigned values, listed in Table 5.5, were similar to those of the original version, with two exceptions. The first one is a larger size of the reward register for the extreme value-based reward WR, which changed from $WR = 1$ to $WR = 15$. The second is a lower tolerance for the PH-test triggering threshold λ, from $\lambda = 33.472$ to $\lambda = 18.958$.

At first glance, it was observed that the DMAB+MA has a worst performance when it is only guided by the CBS function. The changes in the assignation of parameter values for the DMAB+MA without function f_3 focused on parameters related to the reward calculation and the test for abrupt

Table 5.5: Parameter settings for the DMAB+MA algorithm without function f_3.

Parameter	Tested values	Final value		Parameter	Tested values	Final value	
		original	without f_3			original	without f_3
Crossover probability $prob_c$	$[0.01, 0.9]$	0.812	0.632	Length of the reward register WR	$\{1, 2, 3, 5, 10\}$	1	15
Mutation probability $prob_m$	$[0.01, 0.9]$	0.761	0.738	PH-test tolerance δ	$[0.05, 0.5]$	0.299	0.202
Inversion probability $prob_i$	$[0.01, 0.9]$	0.012	0.245	PH-test triggering threshold λ	$[15, 35]$	33.472	18.958
Scaling factor C	$[1, 10]$	7.138	7.686	Max. running time for DMAB s	–	600s	600s

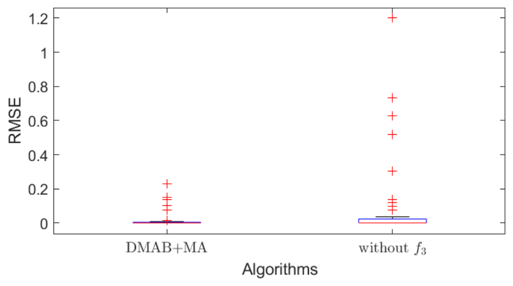
change in reward value. Larger sizes for the reward record WR cause the reward values to variate slower, and a lower threshold for the PH-triggering can recognize variations of smaller magnitude as abrupt. Therefore, it is interesting to discuss how the reward mechanism in the DMAB+MA can be influenced by the fitness functions that are available.

Function f_3 affects the decisions taken by the selection, mutation, survival and local search operators, specially when dealing with equal CBS solutions. For example, after playing an arm that uses function f_3, it can occur that the fitness of the best individual has been improved, while the CBS remains equal. In such cases, the corresponding arm gets a reward equal to zero (recall rewards are calculated from CBS values because the goal is to choose a MA with an operator configurations that can actually reduce the CBS). Even if this results in a lower confidence estimation for the arm in question, the neutral CBS changes that it performed over the population remain, and may help the DMAB+MA to produce fitter individuals with actual CBS improvements in future generations. In this way, function f_3 fulfills the role for which it was designed, to provide guidance for choosing among a group of equal cost solutions.

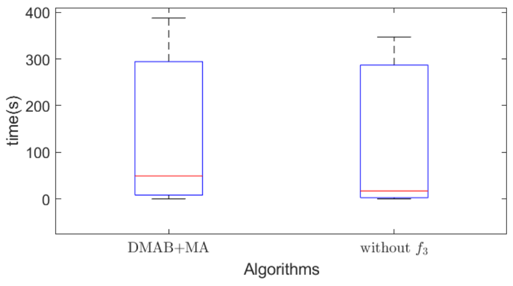
When function f_3 is not available, the search is less able to reach better areas by navigating through neutral areas. With only the CBS function guiding the DMAB+MA, it still performs better than MA-20, demonstrating how powerful the combined use of the genetic operators can be. However, it is still not as good as the original DMAB+MA that can also combine function f_3 and the CBS function. It can be observed that without function f_3, there is larger dispersion of RMSE values and more outliers, indicating different performance variations among instances.

The time at which the reduced DMAB+MA version stopped improving, presented in Figure

5.6(b), is shorter when compared to that of the original version. Considering also its worse RMSE, it seems likely that removing function f_3 caused the reduced version to become stagnant in areas of worse quality.



(a) RMSE distributions.



(b) Average time to the best-found solution.

Figure 5.6: Performance regarding solution quality and expended computing time for DMAB+MA without function f_3.

5.5 Conclusions

This chapter presented an hyperheuristic approach to take advantage of the alternative evaluation function and all the other MA components that were previously developed/implemented, in a new way that was capable of obtaining better results and more consistently. The problem of alternating between operator configurations was expressed as a multi-armed bandit, with the arms representing the different MA versions, and approached it with a DMAB implementation. The DMAB was the chosen hyperheuristic because it offers the capacity of dealing with dynamic scenarios where some operators are more or less useful at different moments, as it was determined that happened with the operators of the independent MA. The DMAB also had the advantage of being easy to integrate within the previous MA implementation while having few parameters to tune.

Our DMAB implementation employed the extreme value-based reward assignation [31], which has the purpose of considering how good an arm has performed over a window of its WR most recent plays, instead of just on the very last one. The estimation of confidence for the arms was carried out by the UCB1 [25] strategy that seeks to balance between exploiting arms with high average rewards and exploring the use of arms that are not often played.

The experiments with the DMAB+MA were executed over the same set of instances employed previously with the independent MA versions, comparing its results with them, GVNS [117] and MACH[47]. The first finding was that the extreme value-based reward was turned off by the parameter tuning process, which assigned the length of the arm's reward record as $WR = 1$, making a case for basing confidence estimations on up to date raw reward values.

By employing the DMAB, it was expected to solve the issues observed with MACH and the individual MA. Specifically, the aim was to develop a general approach algorithm, one that could consistently reach the best-known results, with more independence from the guest graph topology. DMAB+MA succeeded in all the cases where MA-20 and MACH did it, and more importantly, in those where they failed. Out of the ten instances with known optimal values in the set, DMAB+MA found optimal solutions for all of them. The MA had already produced new best-know solutions for the 30 instances with unknown optimal CBS, but some of them were not reached with enough frequency. The DMAB+MA matched and surpassed all the individual MA versions in that sense, producing the new best-found results with close to zero RMSE values, and even further improving 10 of them.

For a closer look on why the DMAB+MA works this well, it was examined the average usage of the operators within the DMAB, contrasting how their combinations affected the MA performance and how that changes when operators of the same type can complement each other. The results hinted that the fitness oriented mutation of the cumulative 2-swap mutation became more useful when its restrictiveness can be eased by the other mutation operators. Also that the DMAB+MA does not need to rely as much on the binary tournament selection for keeping the population fit when the (μ, λ) lets the parent population go. Instead, the $(\mu + \lambda)$ survival strategy, being used in different moments of the search process, helps to compensate the potential loose of fitness in other generations. At the same time, the $(\mu + \lambda)$ search strategy is not employed so often that it creates the stagnating problem as in the MA. Special attention was set on the role of the fitness function. It was experimentally determined that with the same time budget and under the same conditions, the DMAB+MA without the use of the evaluation function f_3 has a poor performance, despite having the advantage of managing fewer arms. This showed that the alternative function is very important for pointing out the search in worthy directions that will not be explored otherwise, and thus, crucial for the success of the DMAB+MA.

The next chapter focuses on the fitness landscape analysis of the CBS. In order to better comprehend the effects that the alternative evaluation function has over it and how MA-20 and DMAB+MA navigate through it, the fitness landscape is studied using several techniques and contrasted with algorithm performance and a group of instance features measured by graph metrics.

6
The fitness landscape of the CBSP

This chapter revolves around the fitness landscape analysis of the CBSP, by taking into consideration three important aspects: the fitness function, certain graph features, and the instance difficulty for the metaheuristic algorithms proposed in this work. The analysis of the fitness landscape of a problem is a mean for understanding its search space and the challenges it can present in relation to the algorithms for solving it. In the fitness landscape analysis literature there is a large, continuously growing, collection of techniques intended to sample, measure, and visualize its different aspects, such as the neutrality [100], ruggedness [93], structures near the optimum [52], etc. The techniques are abundant and sometimes it can exist redundancy among them, to the extent that selecting uncorrelated fitness landscape metrics can be itself a challenge [127]. Providing a comprehensive overview of the fitness landscape analysis literature is out of the scope of this work. The reader can refer to several works that deal with that matter for a more complete perspective [76, 77, 96, 100].

There was three complementary goals behind analyzing the fitness landscape of the CBSP: a) getting a deeper understanding of the problem and further investigating how the proposed alternative fitness function may affect it, b) how the fitness landscape's characteristics are linked to instance features and the problem's difficulty for the metaheuristics proposed in this work, and c) to gain a new perspective on the way these metaheuristics behave, and how their success relates to the search processes being conducted thought specific search areas. This chapter begins by presenting, in Section 6.1, various fitness landscape analysis techniques applied to the CBSP fitness landscape, induced by the traditional CBSP evaluation function and by the alternative evaluation function proposed in Chapter 3. A subset of uncorrelated fitness landscape features was selected to further examine its relationship with instance features and their difficulty. Then, a study for the identification of uncorrelated graph metrics that adequately describe the features of an instance is described in Section 6.2. Section 6.3 deals with the analysis of the interaction among fitness landscape features, characteristics of the problem instances, and search performance by applying principal component analysis (PCA) [43] and exploratory factor analysis (EFA) [41]. Section 6.4 consists of a study of search trajectory networks (STN) for MA-20 and DMAB+MA (the algorithms introduced in Chapters 4 and 5). The STN method is an analysis tool enabling the study of the algorithm's dynamics, based on the visualization and characterization of the search process as a directed weighted graph. Finally, Section 6.5 summarizes the main findings of this chapter.

6.1 The fitness landscape of the CBSP under the alternative fitness function

Function f_3 for the CBSP was designed to help the search methods to discriminate between equal cost solutions by introducing a refined definition of what makes a potential solution better than another one, which assesses a solution not only with the total sum of its cyclic distance, but also by considering its cyclic distances. This section studies the fitness landscape of the CBSP, the changes that the alternative fitness function f_3 introduces on it, and how this may affect its difficulty. In order to determine if the alternative function f_3 significantly changes some fitness landscape features, and which is the nature of these changes a series of specialized analysis were conducted. These include measuring the neutrality [133], determining the variation on the number and cost of global and local optima, autocorrelation analysis [135], fitness distance autocorrelation [60], fitness clouds [134] and negative slope coefficient [132]. The instance set for the experiments presented consists of 90 graphs covering all the topologies listed in Section A.1 for a variety of orders and sizes. It includes a subset of 38 small instances with order $n = \{10, 11, 12\}$ and size $8 \leq e \leq 43$ and 52 larger instances. A list of these 90 instances can be found in Appendix B. Fitness landscape analysis techniques that require a piori knowledge of the global optima [1] were exclusively conducted over the subset of the 38 smaller graphs. The techniques for which the global optima are not required were applied to the whole set.

6.1.1 Neutrality

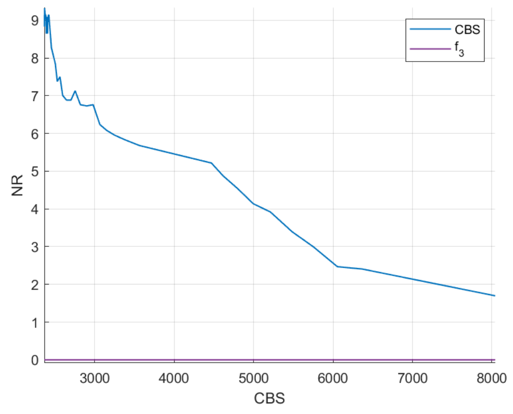
The motivation for proposing an alternative evaluation function was to improve the results of algorithms by reducing the number of solutions that have the same cost. Function f_3 was determined to achieve this goal, while maintaining consistency with the problem's objective. The assessment of the potential of discrimination for function f_3 (see Section 3.4.1) resulted in the ideal relative entropy value ($RE = 1$) for all the instances considered, meaning that in the sampled set of solutions there was none of them with the same fitness evaluation value. Since fitness plateaus consist of connected equal cost solutions, it was conjectured that a reduction in the amount of solutions sharing fitness values was already an indicator of a potential reduction in neutrality. The matter is further addressed here by examining how neutral is the neighborhood of a solution with respect to its fitness, under both the conventional CBS function and the function f_3. This is particularly interesting for high quality solutions, such as the local optima, since reducing the neutrality around them has the potential of opening new routes out of areas where the search process could get trapped otherwise.

The numbers of better, worse and equal fitness neighbor solutions, under the CBS function and

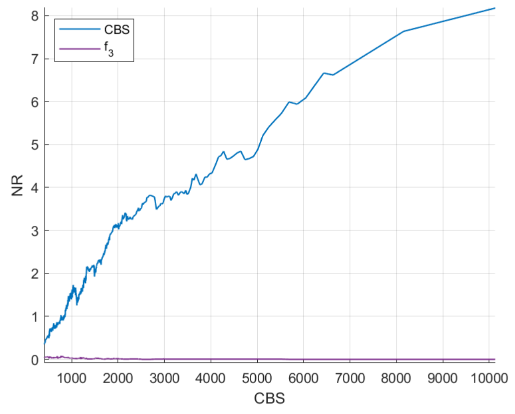
[1] Collected using exhaustive search

function f_3, were recorded for a sample of 100,000 potential solutions visited during 50 runs of a CBS guided ILS, with each run ending after visiting 20,000 solutions. This was repeated over the 90 instances in the set. As expected, under function f_3 the sampled solutions presented fewer neutral neighbors, with most of the equal CBS solutions changing into worsening or improving. However, neutrality is not completely absent, as it can be seen that a reduced percentage of neutral neighbors is still present. Figure 6.1 illustrates changes in the neutrality ratio [133], the percentages of equal fitness neighbors of a solution, with respect to the CBS cost. Under function f_3, the neutrality is significantly reduced, with less variation, and only slightly more often found in the neighborhood of fitter solutions rather than in that of the poorer ones. Recall that under function f_3 the equal CBS value solutions share the same integer part of the cost, which is their CBS and differ only in the floating point part, calculated based on their cyclic distances distribution. Therefore, as the CBS value lowers, there are naturally fewer distinct cyclic distances distributions. Having the same cyclic distance distribution means having as well the same evaluation under function f_3, therefore neutrality can be slightly higher around the fitter solutions. Yet, the neutrality ratio for function f_3 is still generally smaller than it was originally under the conventinal CBS function. This holds even for graph topologies like the Cartesian products involving complete graphs, as it is exemplified in Figure 6.1(a), where lower cost solutions had originally more neutrality.

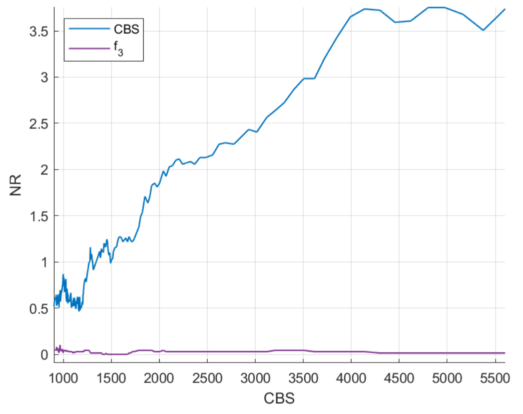
It is also relevant to observe how function f_3 rearranges the equal CBS relationships, particularly around the local optima. For this, we look at the average neutral percentage of the neighborhood of the the local optima that were visited during the ILS. Figure 6.2 represents how the percentages of non deteriorating transitions on the neighborhood of CBS local optima change when the alternative evaluation is in place. First, it was found that in average, around 2.5% of the neighbors of local optima are neutral under the CBS function. Then, when evaluated by function f_3 most of the neighbors became either improving or worsening, and the neutrality within the neighborhood was reduced to approximately 0.5%. This demonstrates that function f_3 significantly reduces the neutrality and it has, in effect, created new non neutral transitions among solutions that used to be part of a plateau



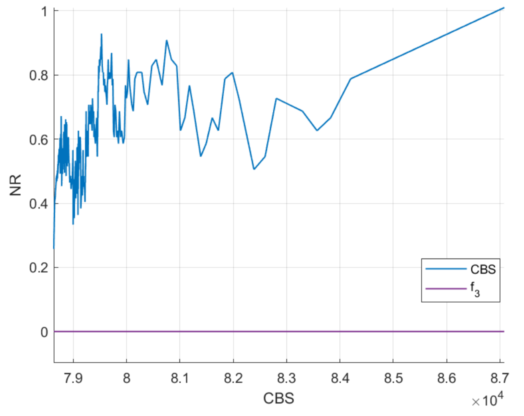
(a) *p9k9*.



(b) *path200*



(c) *bcpwr03*.



(d) *rand100p7*

Figure 6.1: Neutrality ratio in relation to fitness values for both evaluation functions across several instances. Overall, the neutrality ratio was found to be significantly reduced by function f_3 (p-value = 6.8862e-17), while correlated to the CBS values (corr=0.61).

of local optima.

Figure 6.3 shows how the new transitions affect the local optima and the structure of the landscape around them. Under function f_3, the average distance walked to find a local optima, referred as the descent distance, became larger and the cost of local optima also exhibit less variability. While the difference were not determined statistically significant with respect to the CBS function, these small changes can be exploited to benefit the search. For example, consider a CBS local optimum lo and two of its neutral neighbors lo_b and lo_w that became respectively better and worse

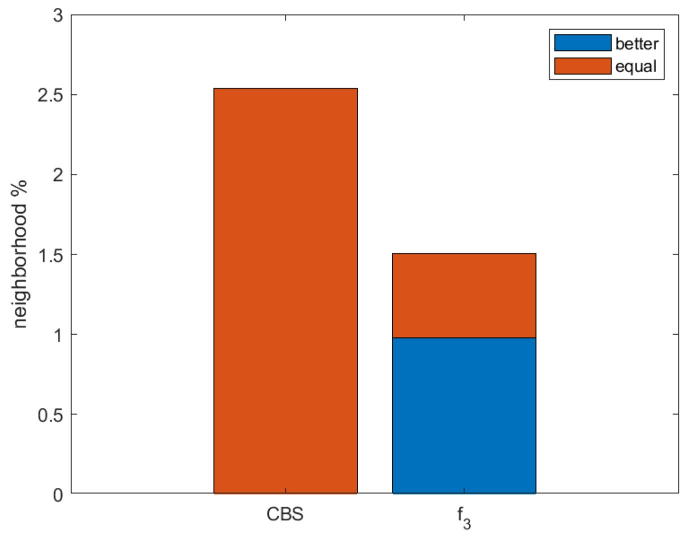
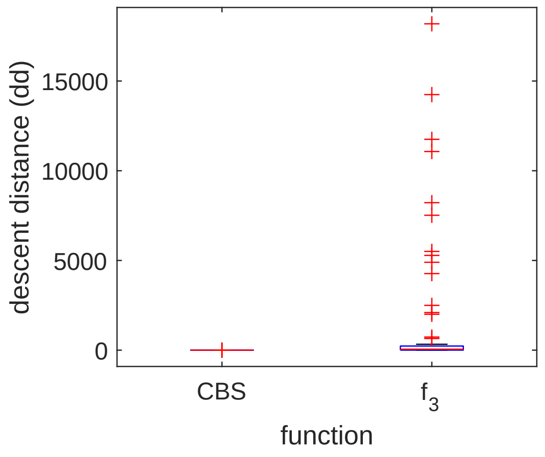


Figure 6.2: Average change in the percentages of non deteriorating transitions from CBS local optima after being evaluated by the alternative function.

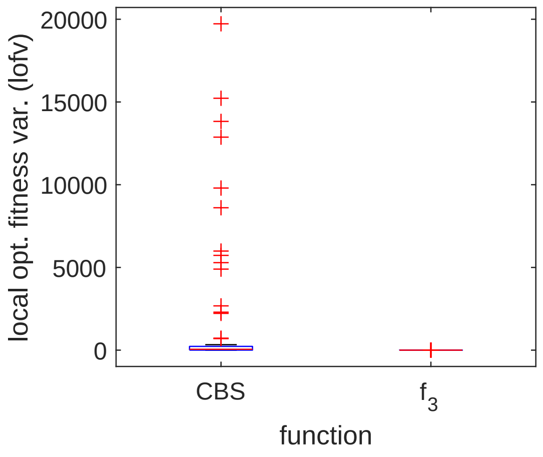
when function f_3 is employed. Not only a new improving transition from lo to lo_b has been created, there is also another one from lo_w to lo. Where a local search process guided by the CBS function could have stooped in solution lo_w, the alternative evaluation scheme will allow it to reach lo, then lo_b and eventually the potentially improving neighbors of lo_b.

6.1.2 Global and local optima

For understanding how the alternative function f_3 affects the fitness landscape it was consider relevant to examine the sets of global and local optima. From the results discussed in the previous section, it can be inferred that the number of local optima must be reduced by function f_3, since it was shown that some of the neutral neighbors of local optima became worsening solutions. A lower number of local optima is a good sign that function f_3 helps to produce better results, since a high number of them is a common indicator of a problem instance difficulty [50]. It was of particular interest to determine if the alternative evaluation scheme creates cases where CBS global optima solutions become suboptimal. It can be assured that if two global optimal solutions have the same distribution



(a) Average descent distance, (p-value=0.0171, corr=0.73)



(b) Local optima fitness variance (p-value = 0.9452, corr=0.99)

Figure 6.3: Average length of the steepest descents during the ILS and variance of the fitness of the local optima.

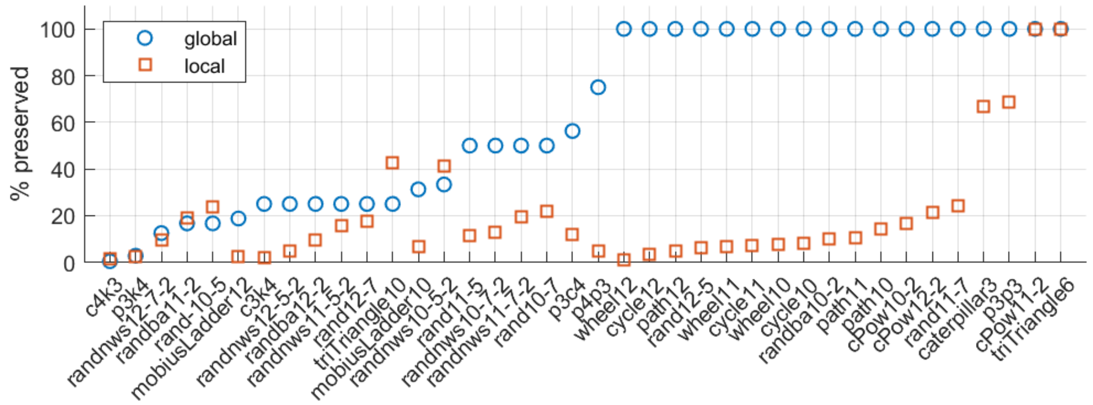
of cyclic distances, then they will have the same evaluation value assigned to them by function f_3. Then, it arises the question of how likely are the global optima to have the same cyclic distances frequencies. Again, in order for this to happen, the solutions not necessarily have the same cyclic distances for the same edges, but rather the same number of occurrences of each of those cyclic distances of magnitude k.

Quantifying changes that affect the global optima presents similar challenges to the ones associated to producing optimal solutions in the first place. Determining which is the optimal value and in how many different ways it can be achieved is a demanding task only practicable for small instances. In order to provide at least an idea of how function f_3 affects global and local optima, an exhaustive sampling was performed for obtaining all the possible embeddings for the 38 smaller graphs in the instance set. While these results are limited to graphs of order $n \leq 12$, they provide evidence that the number of local optima is reduced, as well as information on how much the number of global optima can change. Table 6.1 lists information regarding the global and local optima in the fitness landscape induced by both evaluation functions, including the optimal values (Opt*), the absolute numbers of unique global and local optima and the percentages of those that function f_3

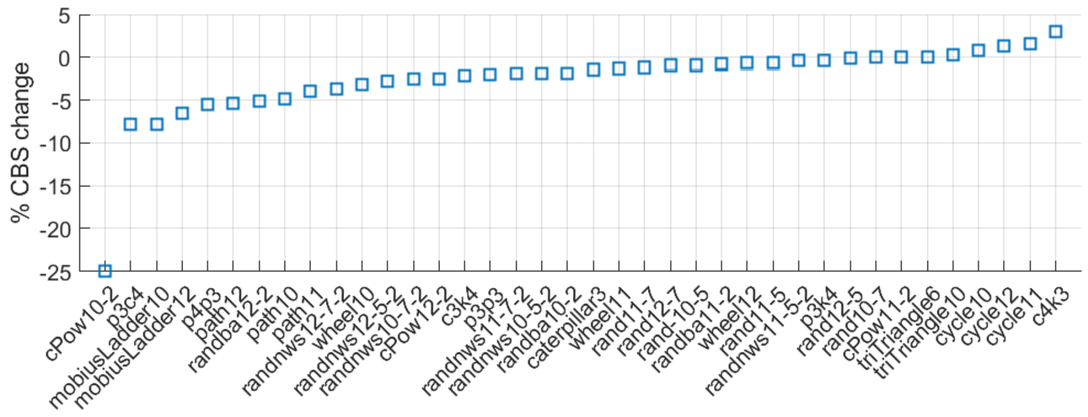
Table 6.1: Analysis of the effects of function f_3 in the number of global and local optima, with respect to the CBS function, for exhaustively solved instances of order $n \leq 12$. The comparison includes the optimal value (Opt*), the number of global (ngo) and local optima (nlo) per function and the percentage of them that was preserved by function f_3.

Instance	$\|V\|$	$\|E\|$	Opt*		ngo			nlo		
			CBS	f_3	CBS	f_3	%	CBS	f_3	%
path10	10	9	9	9.45	10	10	100.00	900	130	14.44
path11	11	10	10	10.46	2	2	100.00	448	48	10.71
path12	12	11	11	11.46	2	2	100.00	1740	86	4.94
cycle10	10	10	10	10.50	10	10	100.00	6430	510	7.93
cycle11	11	11	11	11.50	2	2	100.00	2886	202	7.00
cycle12	12	12	12	12.50	2	2	100.00	10266	326	3.18
wheel10	10	18	35	35.62	90	90	100.00	5490	420	7.65
wheel11	11	20	41	41.61	20	20	100.00	3160	212	6.71
wheel12	12	22	48	48.60	22	22	100.00	20240	210	1.04
cPow10-2	10	20	30	30.75	10	10	100.00	60	10	16.67
cPow11-2	11	22	33	33.75	2	2	100.00	2	2	100.00
cPow12-2	12	24	36	36.75	2	2	100.00	122	26	21.31
p3p3	9	12	19	19.52	72	72	100.00	144	99	68.75
p4p3	12	17	29	29.52	16	12	75.00	564	26	4.61
p3c4	12	20	40	40.53	128	72	56.25	1476	174	11.79
p3k4	12	26	58	58.60	1728	48	2.78	2928	72	2.46
c4k3	12	24	52	52.54	2592	12	0.46	10944	154	1.41
c3k4	12	30	72	72.65	384	96	25.00	11688	240	2.05
caterpillar3	9	8	12	12.35	108	108	100.00	324	216	66.67
triTriangle6	6	9	12	12.63	6	6	100.00	6	6	100
triTriangle10	10	18	32	32.61	120	30	25.00	330	140	42.42
mobiusLadder10	10	15	25	25.51	320	100	31.25	2800	190	6.79
mobiusLadder12	12	18	30	30.51	128	24	18.75	3274	80	2.44
rand10-5	10	20	41	41.60	60	10	16.67	590	140	23.73
rand10-7	10	27	56	56.78	20	10	50.00	320	70	21.88
rand11-5	11	25	55	55.66	8	4	50.00	260	30	11.54
rand11-7	11	43	108	108.93	2	2	100.00	58	14	24.14
rand12-5	12	37	93	93.75	2	2	100.00	486	30	6.17
rand12-7	12	43	116	116.80	8	2	25.00	1188	210	17.68
randba10-2	10	16	33	33.49	40	40	100.00	1620	160	9.88
randba11-2	11	18	37	37.48	24	4	16.67	136	26	19.12
randba12-2	12	20	41	41.52	8	2	25.00	492	48	9.76
randnws10-5-2	10	15	26	26.52	30	10	33.33	340	140	41.18
randnws11-5-2	11	20	38	38.58	8	2	25.00	140	22	15.71
randnws12-5-2	12	16	28	28.48	8	2	25.00	1112	52	4.68
randnws10-7-2	10	19	35	35.61	20	10	50.00	1010	130	12.87
randnws11-7-2	11	20	38	38.60	4	2	50.00	134	26	19.40
randnws12-7-2	12	19	35	35.50	16	2	12.50	330	32	9.70
p-value						0.025			6.89e-12	
correlation						0.1577			0.1988	

preserves. The last rows present the statistical significance and the coefficient of correlation between the measurements. In general terms, there was almost no evidence of correlation between the original numbers of local and global optima and those preserved by function f_3. It can also be concluded



(a) *Global and local optima percentages, respect to the search space size.*



(b) *Change on the average cost of the local optima.*

Figure 6.4: Changes in the number of local optima and their fitness produced by the alternative evaluation scheme on several small instances.

from the p-values that the global optima are not significantly affected by function f_3, while the local optima indeed are significantly fewer. Figure 6.1 illustrates the changes in the size and CBS quality of the sets of local optima for both functions. Since function f_3 creates smaller and, overall, fitter sets of local optima when their actual CBS is considered, it is justifiable to say that the new improving transitions created by the alternative evaluation can indeed make the landscape easier to search.

6.1.3 Fitness distance correlation

The fitness distance correlation (FDC) studies the relationship between the fitness of the solutions and their distance to the nearest global optimum [56, 96]. The result is a single number between -1 and 1 that encapsulates the problem's difficulty based on the global structure of its fitness landscape. For problems considered as *easy*, the closer a solution is to the global optimum, the better its fitness should be. One considerable disadvantage of the FDC technique is that it requires knowledge of the global optima, which often are not attainable for all problem instances. For this reason, it was only evaluated for the 38 smaller instances in the set. For each of the instances, all the global optima under the original CBS function and function f_3 were obtained in an exhaustive listing of the permutations in lexicographical order. The interchange distance [21] was employed for measuring the distance from the solutions to their nearest global optimum.

Let $s_i \in S$ be a set of solutions, while f and d are, respectively, their fitness values and their distance to the nearest global optimum. The correlation value r for all solutions $s_i \in S$ was calculated as:

$$\text{fdc} = \frac{cov(f,d)}{f_\sigma d_\sigma}, \tag{6.1}$$

where $cov(f,d) = \frac{1}{|S|}\sum_{i=1}^{|S|}(f(s_i) - \overline{f})(d(s_i) - \overline{d})$ is the covariance between the fitness values and distance values, $\overline{f}$ and $\overline{d}$ are the averages for fitness and distance, and f_σ and d_σ are their corresponding standard deviations.

In the case of minimization problems, like the CBSP, a value of fdc $= 1$ represents the ideal case where solutions with smaller cost values have smaller distances to the optimum. The fdc value is employed to classify problems into three classes [56, 60], depending on a threshold between -0.15 and 0.15:

- fdc ≤ -0.15: the problem is misleading, fitter solutions are not closer to the global optimum,
- $-0.15 <$ fdc < 0.15: the problem is difficult, almost no correlation exists between fitness and

distance to the optimum,

- fdc ≥ 0.15: the problem is straightforward, the fitter solutions are closer to the global optimum.

Table 6.2 presents the FDC values and the average distance to the optimum for each of the considered instances. It also reports the statistical significance and correlation between these values. For the instances where the fitness landscape for function f_3 has a reduced set of global optima, it can be observed that the average distance to the nearest optimum became larger. On the one hand, the new distances are highly correlated with the values measured for the original set of global optima and there was no evidence that they are significantly different. On the other hand, that changes in distance and fitness were enough to alter the FDC values in a significant way. This seems to suggest that the FDC is very sensible to small variations. For almost all the instances, the results for the original fitness landscape had values fdc ≥ 0.15, which correspond to straightforward or *easy* problems, while for the fitness landscape under the alternative evaluation function the fdc values lowered. This is interpreted as the problem becoming slightly more *deceptive*, in the sense that high fitness solutions have smaller correlations to the reduced set of global optima preserved by function f_3.

6.1.4 Autocorrelation

The correlation structure of the fitness landscape is an indicator of its ruggedness by reflecting of the degree on which the fitness of a solution varies in regard to the fitness of other nearby solutions with a certain step distance [126, 135]. Fitness landscapes with strongly correlated solutions are thought to be smoother, and thus easier for search algorithms.

The autocorrelation [135] is typically studied through a time series based analysis. Let y be a time series of the fitness values of solutions within a random walk of length K. The estimation of the correlation ρ_i between the fitness values y_k and y_{k+i} of two solutions separated by i steps within the random walk is:

Table 6.2: Comparison of fitness distance correlation values (fdc) and average distance to the nearest optimum (ado), for functions CBS and f_3, in exhaustively solved instances of order $n \leq 12$. The last four rows show the results of the statistical significance analysis (p-value) and correlation coefficient for the fdc and ado measures between both evaluation functions.

			CBS		f_3	
Instance	$\lvert V \rvert$	$\lvert E \rvert$	fdc	ado	fdc	ado
path10	10	9	0.240	5.05	0.239	5.05
path11	11	10	**0.139**	6.34	**0.138**	6.34
path12	12	11	**0.114**	7.25	**0.113**	7.25
cycle10	10	10	0.258	5.05	0.257	5.05
cycle11	11	11	0.183	6.34	0.182	6.34
cycle12	12	12	0.151	7.25	**0.150**	7.25
wheel10	10	18	0.319	4.03	0.318	4.03
wheel11	11	20	0.289	4.78	0.288	4.78
wheel12	12	22	0.245	5.46	0.244	5.46
cyclePow10-2	10	20	0.306	5.05	0.306	5.05
cyclePow11-2	11	22	0.223	6.34	0.223	6.34
cyclePow12-2	12	24	0.189	7.25	0.189	7.25
c3k4	12	30	0.337	4.34	0.304	4.84
c4k3	12	24	0.325	3.73	0.199	6.08
p3c4	12	20	0.266	4.92	0.251	5.17
p3k4	12	26	0.418	3.77	0.263	5.32
p3p3	9	12	0.343	3.30	0.342	3.30
p4p3	12	17	0.234	5.99	0.219	6.16
caterpillar3	9	8	0.394	3.06	0.393	3.06
triTriangle6	6	9	0.802	2.05	0.802	2.05
triTriangle10	10	18	0.388	3.83	0.351	4.41
mobiusLadder10	10	15	0.365	3.44	0.311	3.94
mobiusLadder12	12	18	0.238	5.03	0.206	5.78
rand10-5	10	20	0.295	4.10	0.193	5.04
rand10-7	10	27	0.268	4.70	0.241	5.05
rand11-5	11	25	0.151	5.61	**0.113**	6.01
rand11-7	11	43	0.208	6.34	0.209	6.34
rand12-5	12	37	0.153	7.25	0.153	7.25
rand12-7	12	43	0.161	6.45	**0.125**	7.25
randba10-2	10	16	0.223	4.39	0.223	4.39
randba11-2	11	18	0.179	4.89	**0.129**	5.87
randba12-2	12	20	0.173	6.48	0.159	7.25
randnws10-7-2	10	19	0.251	4.71	0.232	5.05
randnws11-7-2	11	20	0.226	5.81	0.177	6.34
randnws12-7-2	12	19	0.219	5.95	0.159	7.25
randnws10-5-2	10	15	0.279	4.45	0.213	5.05
randnws11-5-2	11	20	0.195	5.64	0.174	6.34
randnws12-5-2	12	16	0.157	6.48	**0.137**	7.25
p-value fdc						1.8344e-11
p-value ado						0.2043
correlation fdc						0.7352
correlation ado						0.9168

$$\rho_i = \frac{\sum_{k=1}^{K-i}(y_k - \overline{y})(y_{k+i} - \overline{y})}{\sum_{k=1}^{K}(y_k - \overline{y})^2} \tag{6.2}$$

where $\overline{y}$ is the mean of the fitness values.

The autocorrelation length is the number of steps between two solutions within the random walk before their fitness values become uncorrelated. It is defined as the largest integer l such ρ_l is still different from zero. It assumed that the correlation decreases exponentially in function of the number steps, so the significant margin is bounded as $|\rho_l| < 2/\sqrt{K}$ following [55].

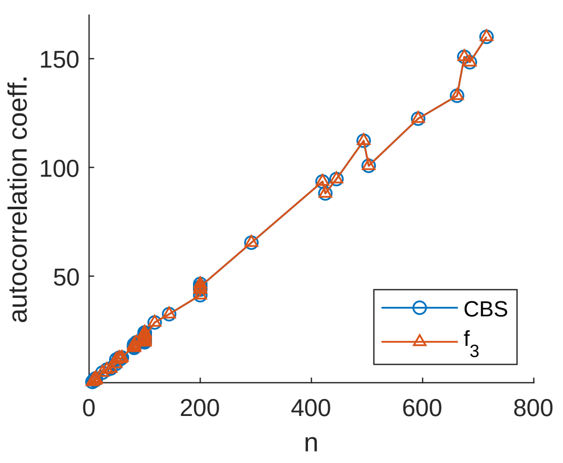
Experiments for measuring the autocorrelation and correlation length for the CBSP were performed using random walks of length $K = 10n$. A new random solution was obtained by employing the 2-swap operator that exchanges the labels of two random vertices. For each of the walks the autocorrelation r_i was calculated according to Equation 6.2. The *overall* autocorrelation at step i was calculated as the average correlation among all the walks at step i. The correlation length is then calculated with respect to the overall autocorrelation.

The autocorrelation coefficient is a derived metric to measure how fast the autocorrelation decays after one step. It is calculated as:

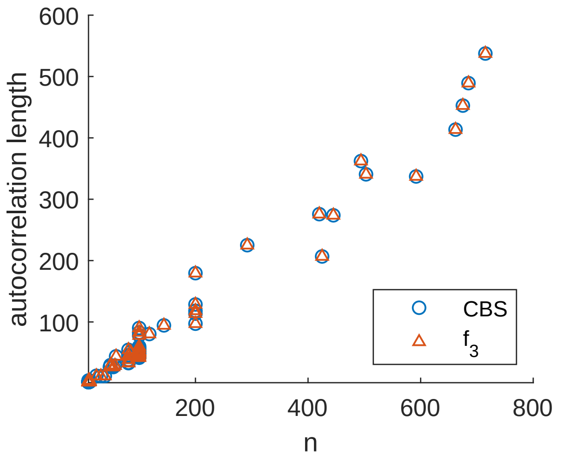
$$\xi = \frac{1}{1 - \rho(1)}\,, \tag{6.3}$$

with values closer to one indicating less ruggedness.

Figure 6.5 shows that the autocorrelation coefficient and the autocorrelation length are have almost identical values regardless of the evaluation function. The results of the the statistical significance test and the correlation for these metrics supports the idea that their values remain very similar. This further demonstrates that function f_3 is effective in reducing neutrality, while the global structure of the landscape, as far as the autocorrelation analysis can capture it, is not significantly disrupted by the use of function f_3 Figure 6.6 presents the autocorrelation for eighth instances of order $n = 100$. Since the results do not differ between functions, only those corresponding to the original evaluation function are presented. In general, it seems that the autocorrelation becomes insignificant after around $n/2$ steps. .



(a) Autocorrelation coefficient (p-value = 0.9771 and corr = 1).



(b) Autocorrelation length (p-value = 1 and corr = 1).

Figure 6.5: Comparison of autocorrelation coefficient and autocorrelation length for the CBS function and function f_3. Since very similar values were obtained for both functions, they overlap.

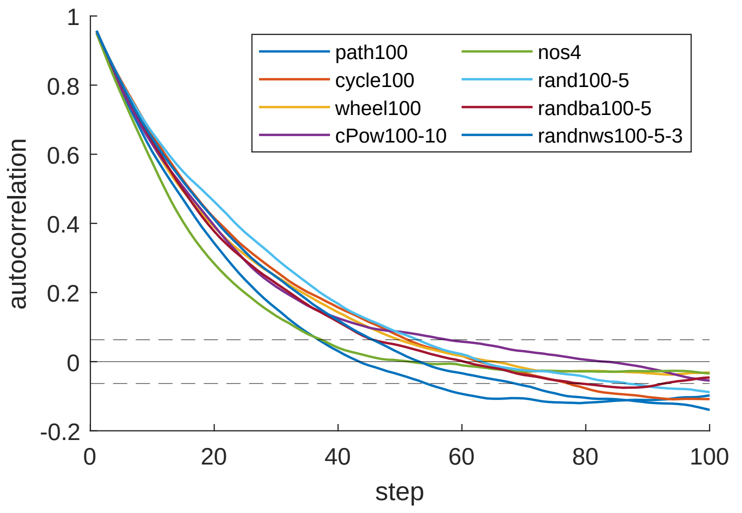


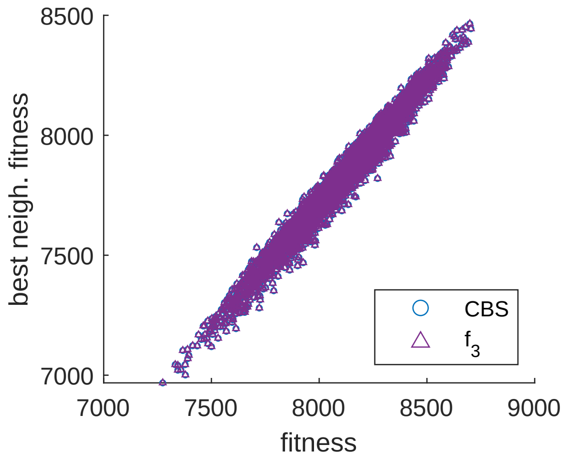
Figure 6.6: Autocorrelation for several instances of order $n = 100$, for the CBS function. Results for function f_3 are omitted, since they were similar.

6.1.5 Fitness clouds and negative slope coefficient

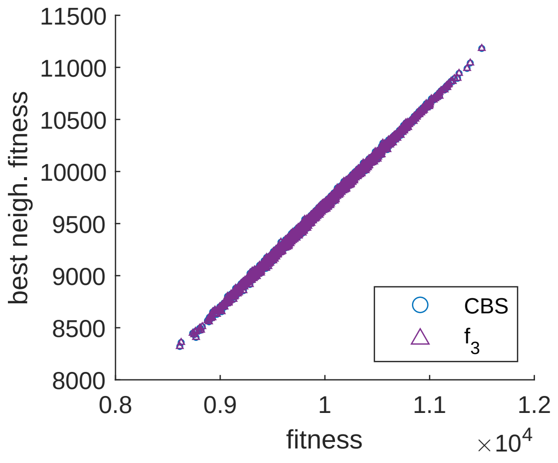
A fitness cloud [134] is a scatter plot for observing the relationship between the fitness values of a group of solutions and the fitness of the new solutions that resulted after applying a neighborhood operator to them. The fitness cloud consists in a group of points were the abscissa values correspond to the fitness of parent solutions and the ordinate values are fitness of the offspring solutions. The purpose of this method is to reflect the ability of a particular operator to produce fitness improvements, i.e., its evolvability, as it allows to identify if fitter solutions are more likely to produce fitter offspring, thus implying the problem may be easy for solving it with the use of evolutionary algorithms.

Figure 6.7 shows the fitness clouds of the CBS function and function f_3 for the 2-swap operator. Two samples per instance were created, one for the CBS function guiding the sampling and another for the alternative fitness function f_3. Each sample contains 250,000 pairs of solutions created by 50 runs with 5,000 pairs per run. Parent solutions, whose fitness corresponds to the abscissa values were generated by Metropolis-Hastings sampling. The ordinate axis shows the fitness values of their offspring solutions, which are their best neighbor obtained by the 2-swap operator. While the previous experiments with steepest descent and ILS (in Chapter 3) showed that the alternative fitness function f_3 causes the steepest descent to have longer descent distance and reach better solutions, it can be easily seen that the general shape of the fitness clouds for both evaluation functions is very similar. This can be attributed to the differences in sampling for the fitness cloud, where the offspring solutions are the best-neighbor after only one step.

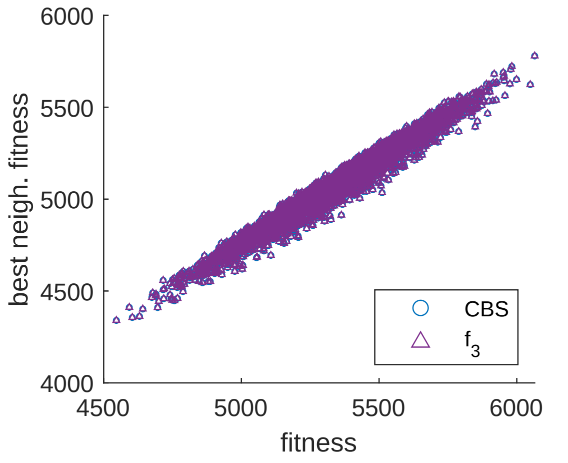
The negative slope coefficient (NSC) [131, 132], is a metric that derives from the fitness cloud analysis. It is a method to quantify the results of fitness clouds into a single numerical value that can be more directly interpreted. It is calculated by partitioning the fitness cloud into m slices according to m equal length intervals defined over the abscissa axis. A slice consists of all the points whose abscissa value is between the predefined interval. Then, for each slice the average of the abscissa



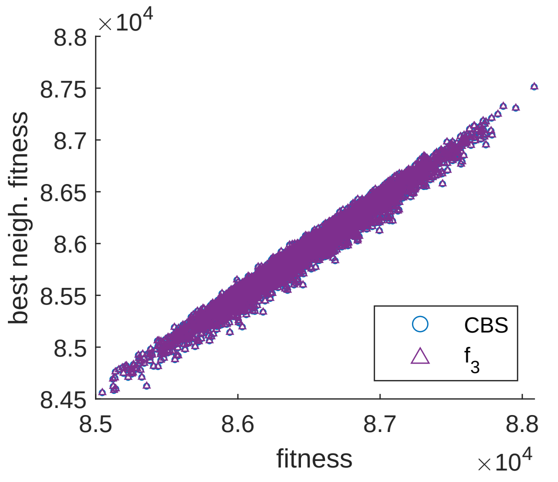
(a) *p9k9*.



(b) *path200*



(c) *bcpwr03*.



(d) *rand100p7*

Figure 6.7: Comparing the fitness cloud obtained by the 2-swap operator and both studied fitness functions.

x_i and ordinate values y_i are calculated, with $0 \leq i \leq m$. The resulting values correspond to the average of fitness values before and after applying the operator. The slope for a slice is calculated as follows.

$$z_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad 0 \leq i \leq m - 1\,. \tag{6.4}$$

The negative slope coefficient is equal to the sum of the slopes with negative values.

$$\text{nsc} = \sum_{i=1}^{m-1} \min(z_i, 0)\,. \tag{6.5}$$

The magnitude of the nsc value is an indicator of problem difficulty in terms of evolvability, where smaller values indicate it is harder to produce improving solutions, and the maximum possible value, nsc $= 0$, corresponds to an easy problem.

Figure 6.8 presents the NSC results for the instance set. The values are very similar and highly correlated between both functions. The previously changes introduced by function f_3 (reduced neutrality, lowered cost of local optima and decrease in their number) do not seem to significantly affect the NSC results. However, they were observed to help produce significantly better results. Furthermore, the NSC had values linked to better evolvability for 24 out of the 90 instances.

6.1.6 Selection of fitness landscape features

The previous sections presented a study of how the fitness landscape features were affected by the alternative fitness function. These results were analyzed to determine which features were able

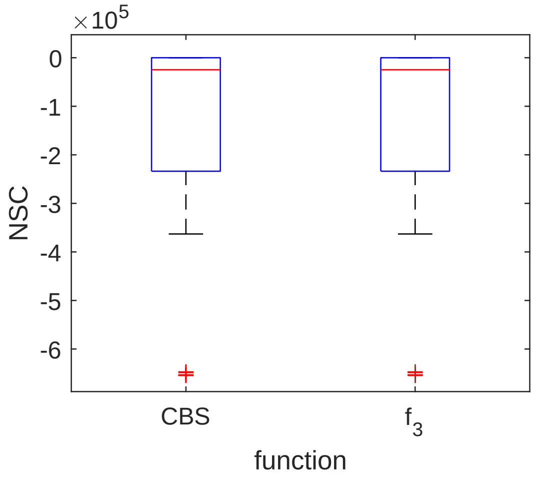


Figure 6.8: Negative slope coefficient among the instance set, evaluated for both functions (p-value=0.9520, corr = 1).

to capture changes in the fitness landscape between both evaluation functions, and therefore can be considered as more relevant for further study. Three criteria were defined in order to select fitness landscape features and deciding if their values for both functions are necessary or generally redundant. The first criteria for reducing the feature set was that, if a feature shows not a significant difference between its values for the CBS function and for function f_3, and these values are highly correlated, then there is no that much relevant new information to obtain from having both set of values participating. Therefore, it was assumed that the feature could be represented good enough solely by its measurements for the CBS function. The second criteria was to eliminate features that are not practical in the general case, since they require a priori knowledge of the global and local optima set. It was only possible to obtain it by exhaustive enumeration, for the fraction of the instance set with order $n \leq 12$, so further analysis including that type of features would exclude most of the instances. The first two criteria were used to remove most of the results for the fitness landscape features for function f_3. Finally, the third criteria was based in a correlation study between the remaining features, in order to identify a subset of uncorrelated features that can represent the fitness landscape.

Table 4.1 summarizes the results for the statistical significance analysis of the measured features and their correlation, specifying which features were selected, based on the first two criteria. Among the fitness landscape features, only the neutrality ratio (neu), the number of local optima (nlo), and fitness distance correlation (fdc) values are significantly affected when the alternative fitness function is implemented. The number of local optima (nlo) and fitness distance correlation (fdc), however, were eliminated by the second criteria, since they require knowledge about the global and local optima. Features like the autocorrelation (ac1 and ac10), autocorrelation length (al), autocorrelation coefficient (acf) and the negative slope coefficient maintained almost identical values regardless of the function. The values of these features were judged to be represented adequately enough by the values measured for the CBS function only. The descent distance (dd), the variance on the fitness of local optima (lofv), the number of global optima (ngo) and the average distance to the optimum

Table 6.3: Selection of the measurements of the fitness landscape features for the CBS function and function f_3, based on the first two criteria.

Feature	sampling	p-value	stat. significant	correlation	selected
neu - neutrality ratio	ILS	6.8862e-17	yes	0.61	CBS and f_3
dd - descent distance	ILS	0.0171	no	0.73	CBS
lofv - local optima fitness variance	ILS	0.9452	no	0.99	CBS
ngo - number of global optima	exhaustive	0.0253	no	0.15	none
nlo - number of local optima	exhaustive	6.8977e-12	yes	0.19	none
fdc - fitness distance correlation	exhaustive	1.8344e-11	yes	0.73	none
ado - average distance to optimum	exhaustive	0.2043	no	0.91	none
ac1 - autocorrelation at step 1	random walk	0.9771	no	1	CBS
ac10 - autocorrelation at step 10	random walk	0.9795	no	1	CBS
al - autocorrelation length	random walk	1	no	1	CBS
acf - autocorrelation coefficient	random walk	0.9771	no	1	CBS
nsc - negative slope coefficient	Metropolis-Hastings	0.9520	no	0.83	CBS

(ado) were observed to vary between functions, yet, these variations were not larger enough to pass the significance threshold. Furthermore, in most cases they have some correlation to results for the CBS function. The number of global optima (ngo) and average distance to the nearest optimum (ado) also fall under the criteria excluding features that require to have a priory knowledge of the global and local optima. Therefore, they were removed.

Figure 6.9 shows the correlation matrix for the selected fitness landscape features, listed in Table 6.3. The asterisks in a cell indicate the significance level, with no asterisks meaning the correlation is not considered significant and three asterisks meaning the correlation is considered extremely significant. Since the correlation matrix is symmetrical, only the cells below the main diagonal are presented. There were high correlations for the autocorrelation based features: autocorrelation at step 1 (ac1), autocorrelation at step 10 (ac10), autocorrelation length (al) and autocorrelation coefficient (acf). The local optima fitness variance (lofv) is also positively correlated with all the autocorrelation features. The autocorrelation group of features has negative correlations with the negative slope (nsc) and the neutrality (neu), to a lesser extent with neutrality for the alternative function (neuf3). Meanwhile, the descent distance (dd) has in general low correlation with the rest of the features.

For the selection of fitness landscape features, two features were considered as highly correlated if the correlation value was greater than 0.80. The neutrality for both functions (neu and neuf3),

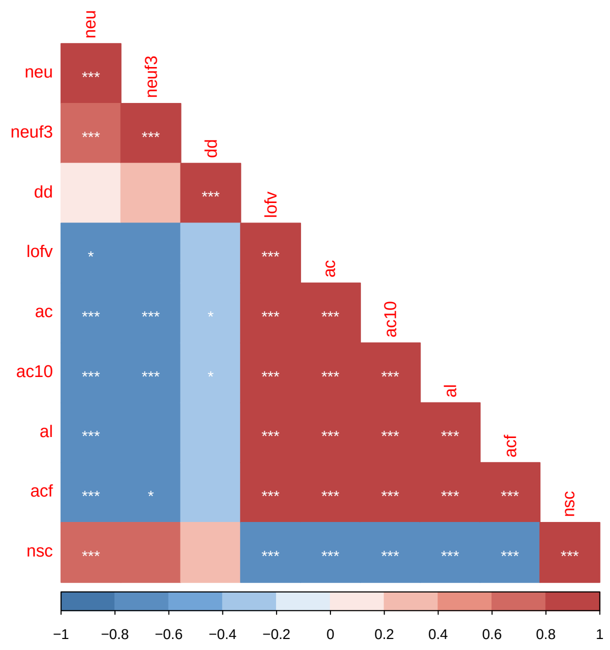


Figure 6.9: Pairwise correlation among the reduced set of fitness landscape features

descent distance and negative slope coefficient were selected because they have no correlation grater than 0.80 with any other feature. From the autocorrelation features, only the autocorrelation coefficient was selected.

6.2 Instance features

In order to investigate the links between features of the problem instances, their fitness landscape and the difficulty that such instances can present for our algorithms, it was necessary to establish a framework for instance description. Since the CBSP instances consist of graphs, a set of graph metric was chosen from the complex network analysis literature. These metrics include different aspects of the graphs, like their size, order, connectivity patterns and spectral metrics. The following metrics were measured over the set of graphs, employing the built-in resources of the NetworkX [45] library for Python. For multivalued metrics, the average and variance were included.

- **order (n):** The number of vertices of the graph.
- **size (e)**: The number of edges of the graph.
- **density (den):** The density of the graph, measuring how close the number of edges of the graph is to the maximum number of possible edges for its order.
- **node degree (deg and degv):** Average and variance of the number of edges for each vertex. It was normalized by the number of vertices.
- **maximal degree (degm):** Maximum degree of any vertex, normalized by the number of vertices.
- **endpoints (enp):** Number of the edges with degree equal to 1, normalized by the total number of edges.
- **average neighbor degree (avnd and avndv):** Average and variance for the degree of the neighbors of each vertex, normalized as a percentage of $n-1$.
- **degree centrality (dcen and dcenv):** Average and variance of the fraction of vertices connected to a vertex. Normalized by the maximum possible degree in a simple graph, $n-1$, where n is the number of nodes.
- **eccentricity (ecc and eccv):** Average and variance of the maximum distance from each vertex to any other vertex. Normalized by $e-1$.
- **radius (rad):** Minimum eccentricity. Normalized by $e-1$.
- **diameter (dia):** Maximum eccentricity. Normalized by $e-1$.
- **central vertices (cen):** Number of nodes in the center of the graph, which consists of the nodes with eccentricity equal to the radius. Normalized by the number of vertices n.

- **periphery nodes (pern):** Number of nodes in the periphery of the graph, which consists of the nodes with eccentricity equal to the diameter. Normalized by the number of vertices n.
- **node connectivity (ncon):** The number of nodes that, if removed, would cause the graph to become unconnected or trivial. It is equal to the number of disjointed paths in the graph. The Networkx library [45] calculates this in an approximated way [136].
- **clustering coefficient (clc):** The average of the clustering coefficients for each vertex. The clustering coefficient for a vertex is the fraction of possible cycles of length equal to three that involve that vertex.
- **closeness centrality (ccen and ccenv):** The clossenes centrality is the reciprocal of the average shortest distance from a vertex to the rest. The vertices that have a small average distance to the other vertices are then considered more *central*. Since it is multivalued, its average and variance across vertices are included. It is normalized by $(n-1)/(|G|-1)$.
- **between centrality (bcen and bcenv):** For each vertex, it is the fraction of all the shortest paths among two vertices that pass through it. Average and variance are included. The values were normalized by $2/((n-1)(n-2))$.
- **spectral radius (srad):** Largest magnitude eigenvalue of the normalized Laplacian matrix of the graph. The Networkx library [45] normalizes the Laplacian matrix as

  $$N = D^{-1/2}(L(D^{-1/2}))\,,$$

  where L is the Laplacian matrix and D is the degree matrix.
- **second largest eigenvalue (slev):** The second largest magnitude eigenvalue.
- **energy (ene):** Square sum of the eigenvalues.

6.2.1 Selection of graph features

For the graph metric selection, it was observed how the features relate to each other in terms of correlation, similarly to the correlation-based elimination process that was implemented for the fitness landscape features. Figure 6.10 shows the pairwise correlations between the graph features. Graph features were considered as highly correlated if the correlation coefficient was over 0.80.

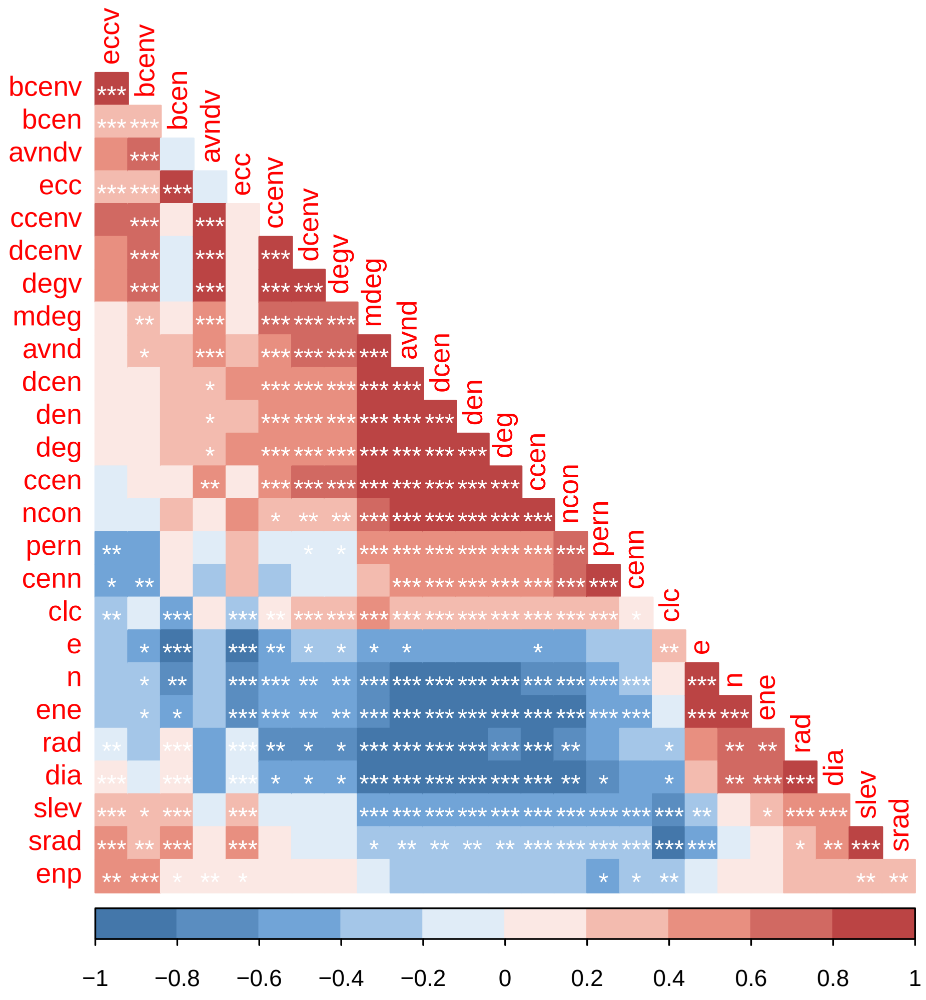


Figure 6.10: Correlation matrix among the graph metrics.

It was observed that several features focused on connectivity, such as density (den), degree (deg), maximal degree (mdeg), average neighbor degree (avnd), degree centrality (dcen), closeness

Table 6.4: Selected graph features and the features correlated to them that were eliminated.

selected feature	correlated features	selected feature	correlated features
n - order	e - edges	enp - number of endpoints	none
	ene - energy	ecc - eccentricity	none
den - density	deg - node degree		bcen - betweeness centrality
	avnd - average neighbor degree		bcenv - betweeness centrality variance
	dcen - degree centrality	eccv - eccentricity variance	none
	ncon - node connectivity	rad - radius	dia - diameter
	ccen - closeness centrality	cen - central vertices	pern - periphery nodes
degv - degree variance	dcenv - degree centrality variance	srad - sprectral radius	slev - second largest eigenvalue
	ccenv - closeness centrality variance	clc - clustering clustering	none

centrality (ccen), node connectivity (ncon) are strongly correlated to each other. Among these features, density (den) was selected because it is the easiest one to calculate. Several features inherently related were shown to be correlated. For example, the radius (rad) and the diameter (dia), which are the minimum and maximum eccentricity values. Another example was found in the number of central nodes (cenn) and periphery nodes (pern), which are the number of nodes with degree equal to the radius and the diameter, respectively.

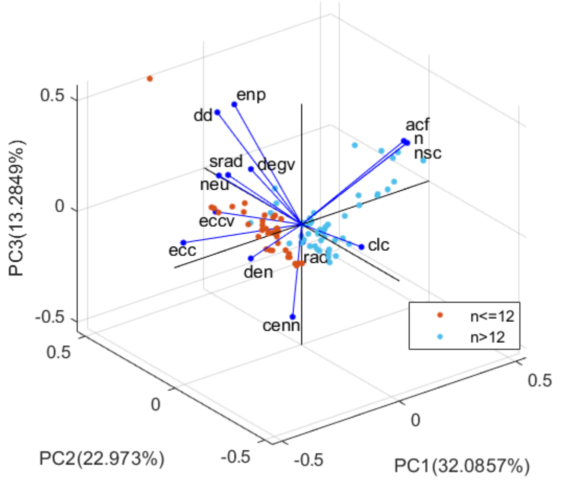
Table 6.4 presents a list of selected features altogether with the discarded features that were correlated to them. In total, ten graph features remained to describe the instance set.

6.3 Analyzing instance features, fitness landscape and problem difficulty

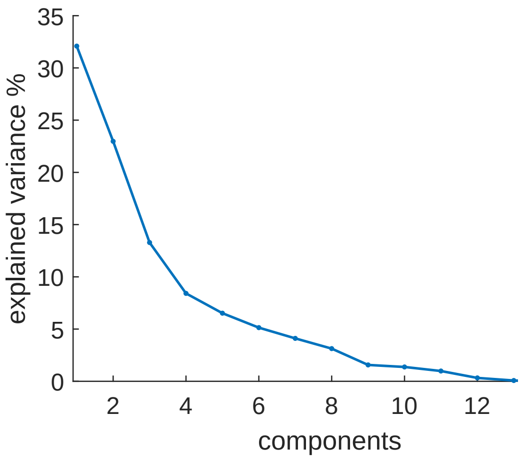
A challenge that arises when studying the interplay between a group of features is that it results difficult to obtain, identify and interpret useful information from the raw data, specially if there is a high number of dimensions. In this work, principal components analysis (PCA) [43] and exploratory factor analysis (EFA) [41] were employed as methods to extract information from the set of features previously introduced for fitness landscape, instance characteristics and algorithmic performance. These techniques help to deal with the high dimensionality issue, by reframing the raw data in a new way that expresses the combined variation among features. PCA works by extracting the eigenvectors of the covariance matrix. The eigenvectors are orthogonal to each other and they are built to maximize the covered variance across the original dimensions of the data. The importance of an eigenvector is given by its respective eigenvalue, which represents the amount of total variance in the data that the eigenvector explains. PCA's components are the eigenvectors sorted by their amount of explained variance, with the first being the most important. The first i-th components covering most of the variance can be employed to reduce the number of considered dimensions for applications like clustering, classification or data visualization. They are also useful to assist in defining the number of factors to extract in the exploratory factor analysis, which was its purpose in this work.

EFA, unlike PCA, distinguishes between the variance specific to a particular feature and the common variance. In EFA, the specific variances are discarded, considering the common variance shared across features as the one that is truly descriptive of their underlying relationships. These underlying relationships are meant to be captured by the extracted factors and their loadings. The factors can be understood as constructs or synthetic variables capturing latent internal attributes of the data that explain its variability and structure. The loadings are the correlation values between a factor and the original data features. These help in identifying which features of the data influenced a factor, in order to determine which features were involved in these latent internal attributes and their relationships. EFA can apply rotations in order to analyze the data from a different perspective that is deemed more suitable and helps to produce *simpler* results. The simplicity of the results means that, ideally, factors should have only a few unique significant loadings with the features and mostly close to zero loadings with the rest of features [64, 129]. Rotations can be orthogonal, suitable for cases where the factors are likely to be uncorrelated, or oblique, if the factors may be correlated [41].

The data set to analyze consisted of the four fitness landscape features and the ten graph features that were previously discussed, with 90 observations. The z-score normalization was applied, fitting the values into the range -1 to 1, with mean equal to zero. This normalization had the purpose of



(a) Loadings for three main components.



(b) Expalined variances per components.

Figure 6.11: Feature loadings for the three main components and percentage of the variance covered by each component. Together, the three main components explain 68.34% of the total variance.

adjusting the different magnitudes of the feature vectors into the same scale.

PCA was conducted separately for the traditional evaluation function and the alternative one, obtaining similar results, since the features only differed in the neutrality values (neu and neuf3). Figure 6.11 presents the first three principal components obtained from applying PCA to the feature matrix. It includes the loadings of features over these components, which are the weights representing how much a feature influences an eigenvector. The two subgroups of instances correspond to the graphs of size $n \leq 12$ that were solved exhaustively, and the larger graphs. In the plot, is possible to clearly identify the two distinct subgroups occupying differentiated regions. The loadings provide additional information about the features in regards to their correlation: loading vectors forming angles of less than 90° can indicate positive correlation, angles closer to 180° denote negative correlation and angles of around 90° would mean the metrics are uncorrelated.The amount of variance explained for each component is shown in Figure 6.11(b). Combined, the three principal components in Figure 6.11(a) explain 68.34% of the data variance. It was found that with half of the components, the first seven out of 14, the percentage of explained variance can be raised to 92.52%. From this result it was decided to extract seven factors in the exploratory factor analysis.

The results of EFA, extracting seven factors are presented in Table 6.5. These results were achieved using an oblique rotation, not granting that the factors would be uncorrelated. Therefore, the loadings represent regression coefficients instead of linear correlations. As shown in Table 6.5, EFA produced similar loadings for the original evaluation function and the alternative one. As for PCA results, their similarity is due to their respective data matrices differing only in the values of the neutrality ratio (neu) feature, which were significantly lower for function f_3, while correlated to their counterparts for the CBS function.

Factor F1 is linked to the order of the graph (n) and two fitness landscape features: autocorrelation coefficient (acf) and negative slope (nsc). Both of the fitness landscape features capture related aspects to the variation of fitness around a solution after the application of one operator. Higher values of autocorrelation and negative slope coefficient are related, respectively to larger gradients and better evolvability. Factor F1 shows that fitness landscape of larger instances has larger gradients and fitness values that improve gradually, in relation to a particular number of steps. This suggests that factor F1 may be an indicator of the presence of large attraction basins, the valleys at the fitness landscapes where the fitness improves towards a local optimum in the bottom of the valley. The correlation of factor F1 with the RMSE of DMAB+MA can then be an indicator that such basins can require more effort to escape due to their size and its relationship with the order of the graph. Therefore, the DMAB+MA may improve its performance if some of its operators, such as the local search phase o mutation, increased their number of operations over a solution in function of the graph's order. It is also relevant to remark that DMAB+MA has better O-RMSE values than MA-20 (0.015 and 0.1262 respectively), so the comparatively smaller correlation of MA-20 with factor F1 does not indicate better performance.

Factor F2 is influenced by the spectral radius and the clustering coefficient of the graph. However,

Table 6.5: Factor loadings with features, the cumulative proportion of variance (cv) explained by each factor and their correlation with solution quality (RMSE), for MA-20 and DMAB+MA.

	CBS							f_3						
feature	F1	F2	F3	F4	F5	F6	F7	F1	F2	F3	F4	F5	F6	F7
n	**0.99**	0.00	0.02	-0.02	-0.01	0.00	0.05	**0.99**	0.00	0.02	-0.02	-0.01	0.00	0.05
den	0.13	-0.03	-0.13	0.14	**0.66**	**0.60**	-0.30	0.13	-0.03	-0.13	0.14	**0.66**	**0.60**	-0.30
degv	-0.04	0.09	0.16	-0.10	-0.14	**0.73**	-0.02	-0.04	0.09	0.16	-0.10	-0.14	**0.73**	-0.02
enp	0.06	-0.07	**0.61**	0.21	-0.20	-0.10	-0.09	0.06	-0.07	**0.61**	0.21	-0.20	-0.10	-0.09
ecc	-0.14	-0.19	0.02	**0.49**	0.18	0.16	**0.46**	-0.14	-0.19	0.02	**0.49**	0.18	0.16	**0.46**
eccv	-0.01	0.14	-0.06	**1.06**	-0.15	-0.07	0.04	-0.01	0.14	-0.06	**1.06**	-0.15	-0.07	0.04
rad	0.19	0.03	0.03	0.09	0.05	-0.16	**0.63**	0.19	0.03	0.03	0.09	0.05	-0.16	**0.63**
cenn	-0.08	0.13	0.09	-0.20	**0.77**	-0.14	0.19	-0.08	0.13	0.09	-0.20	**0.77**	-0.14	0.19
srad	0.01	**-0.97**	-0.06	-0.13	-0.22	0.18	0.12	0.01	**-0.97**	-0.06	-0.13	-0.22	0.18	0.12
clc	-0.01	**0.94**	-0.02	-0.02	-0.10	0.38	0.12	-0.01	**0.94**	-0.02	-0.02	-0.10	0.38	0.12
neu	-0.08	-0.03	**0.48**	-0.03	0.25	0.21	-0.05	-0.08	-0.03	**0.48**	-0.03	0.25	0.21	-0.05
dd	0.03	0.04	**1.01**	-0.08	0.03	0.07	0.06	0.03	0.04	**1.01**	-0.08	0.03	0.07	0.06
acf	**1.01**	-0.01	0.01	0.00	0.02	0.03	0.04	**1.01**	-0.01	0.01	0.00	0.02	0.03	0.04
nsc	**0.99**	0.00	0.02	-0.02	-0.01	0.00	0.05	**0.99**	0.00	0.02	-0.02	-0.01	0.00	0.05
cv	0.22	0.35	0.47	0.58	0.67	0.76	0.81	0.22	0.35	0.47	0.58	0.67	0.76	0.81
MA-20	0.35	-0.07	-0.15	0.07	-0.09	-0.32	**0.65**	0.35	-0.07	-0.15	0.07	-0.09	-0.32	**0.65**
DMAB+MA	**0.73**	-0.06	-0.10	-0.16	-0.33	-0.27	0.00	**0.73**	-0.06	-0.10	-0.16	-0.33	-0.27	0.00

it is not affected by fitness landscape features and it does not correlate to the algorithm's performance. Because of this, it can be regarded as a problem independent relationship between graph features, with no evidence of it having a significant role in problem difficulty.

Factor F3 is influenced by the number of endpoints (enp), the neutrality ratio (neu and neuf3) and the descent distance (dd). The loadings are similar, regardless of the fitness function, showing that it relates to the rest of features in the same way. Endpoint vertices have the smallest worst case contribution to CBS, even if they are placed the farthest from their neighbors, they can contribute only with a cyclic distance of value $n/2$. This may result in fewer cases where applying the 2-swap operator causes larger, more abrupt fitness variations, perhaps even neutral ones. Factor F3 seems to indicate that instances presenting a larger number of vertices with degree equal to 1 may be a little easier for MA-20, while the intermittent local search may be suitable to jump ahead in the search, if the descent distance is large. Observing also the relationship of factor F1 with the descent distance and neutrality, it could be interpreted as an indicator of the size of plateaus. If this was the case, it would explain the smaller correlation of DMAB+MA performance with factor F3, since this algorithm has the alternative fitness function available, being better equipped to deal with neutrality.

However, an actual estimation of the size of plateaus would be necessary to confirm this.

Similarly to factor F2, factors F4 and F5 also reflect mostly relationships between graph features. Factor F4 captures the average eccentricity and its variance, implying it is more likely that the eccentricity across vertices varies more when the average is larger. Factor F5 is influenced by the graph density and closeness centrality, which suggest denser graphs have more vertices with eccentricity equal to the radius (minimum eccentricity), which makes sense, since a higher density would mean the graph has more edges and therefore, more paths available among vertices. Factor F6 is linked to the density and the variation on the degree of vertices. The correlation with MA-20 and DMAB+MA indicates that instances with these characteristics may make an instance easier.

The average eccentricity (ecc) and radius (rad) are the features influencing factor F7. These features are linked: the radius is the minimum eccentricity value for any vertex. The factor seems to capture the occurrence of high eccentricity, and according to its correlation with MA-20's performance, this algorithm has more difficulties dealing with problem instances with this characteristic. Meanwhile, the DMAB+MA shows zero correlation with factor F7, supporting the claim that is indeed less susceptible than MA-20 to characteristics of the graph, beyond the order.

6.4 MA-20 and DMAB+MA analysis through search trajectory networks

The search trajectory networks (STN) [89] are models for analyzing and comparing the behavior of metaheuristic algorithms over the fitness landscape of a problem. These models were employed in the CBSP research to further examine the differences in behavior between MA-20 and DMAB+MA and how face the challenges presented by the problem itself.

The STN models provide a framework for the visualization and comparative analysis of the search dynamics present in metaheuristic algorithms. The main idea behind this is to create a map of the journey of a metaheuristic algorithm through the explored regions in the search space. Such a map is built from recording the solutions that were the best-found at a given iteration, which provide a snapshot of how the search was being conducted, hence they are called representative solutions. When creating a STN model, the search space is divided into locations, each of them containing at least one representative solution. The search trajectories are the sequence of transitions performed by the algorithm between these locations. An STN is a representation of the search trajectories as a directed weighted graph, that is then analyzed, by employing network metrics, in order to extract relevant information about the underlying structure of a fitness landscape and how the algorithms navigate through it.

The STN derive from the local optima networks (LON) [90], a related fitness landscape analysis technique focused in the study of the transitions among local optima. While LON are usually built from samples of local optima obtained by local search algorithms, such as ILS, the STN do not require the representative solutions to be locally optimal. Hence, the STN are suitable for analyzing the behavior of a wider variety of metaheuristic algorithms.

Formally, given a metaheuristic algorithm M, an STN is built as a weighed, directed graph $STN_M(V_M, E_M)$ where:

- V_M is the set of **locations** visited by the algorithm.
- E_M is the set of directed edges, such that two nodes $a, b \in V_M$ are adjacent if the algorithm M performed a transition between solutions in the respective locations. The weight of the edge represents the number of times the transition from a to b occurred.

Similarly to LON [13], the STN models for the same problem instance can be merged in order to compare the search dynamics of two metaheuristics. The merged STN for two metaheuristic algorithms M_i and M_j is the union of the STN graphs STN_{M_i} and STN_{M_j} as $STN_{M_i,M_j} = (V_{M_i} \bigcup V_{M_j}, E_{M_i} \bigcup E_{M_j})$.

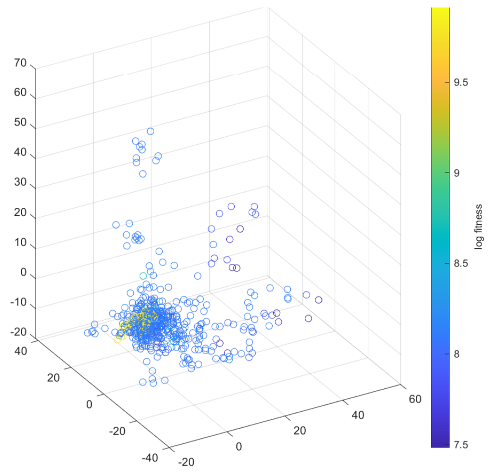
The search trajectories of an algorithm are sampled by recording the representative solutions visited by an algorithm during several independent executions for a problem instance. The frequency of the recording is controlled by a parameter.

6.4.1 Search space partitioning into locations

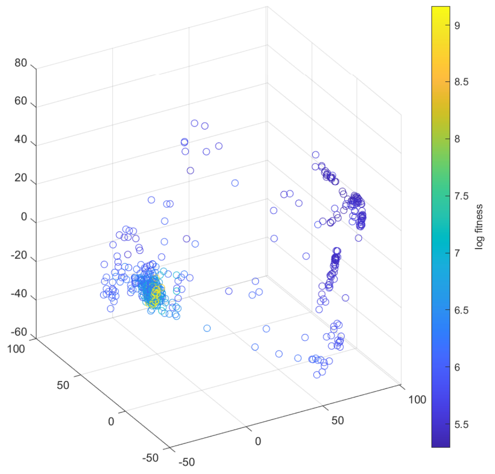
An important step for modeling the search trajectory samples into a network of locations that mets the definitions of nodes, edges and weights of the STN, is to have a mapping between single solutions in the sample and locations. While unique solutions can easily be mapped to unique locations, partitioning the search space in locations of larger size has several advantages. It improves the readability of the STN visualization, it allows to identify search regions with common characteristics and global tendencies in the search dynamics. The partitioning aspect of the STN modeling process implies that if an algorithm visits one location, it is likely to be able to reach the best solution in the location. Therefore, the partitioning definition must employ a notion of closeness between solutions. For example, for continuous optimization problems, a partitioning of the search space into hypercubes of regular size was induced by adjusting the precision of the solution encoding [89].

To apply the STN modeling for the CBSP, which has a combinatorial search space, we proposed the use of multidimensional scaling as the partitioning method that would define which solutions can be grouped into the same location. The multidimensional scaling is a dimensionality reduction technique that has been employed by several authors to visualize the distribution of solutions and to analyze the cartography of search spaces [66, 98]. In these works, the multidimensional scaling was employed to transform groups of solutions from their original n-dimensional representation to the Euclidean space. Within the transformation process, the distortion of pairwise distances among solutions is minimized, preserving the notion of which solutions are close to each other. Classical multidimensional scaling was applied to merged samples of solutions, which contain the solutions visited by the two algorithms along all runs for the same instance. This ensures that a solution that was visited by different algorithms, or in different runs, gets assigned the same three values in the reduced dimensions and can then be identify as the same solution during the creation of the STN. The pairwise distances were evaluated by employing once again, the interchange distance.

The location mapping definition for the STN was that two solutions belong to the same location if, after the multidimensional scaling, the integer parts of their three dimensions are equal. The distance preservation happening within the classical multidimensional scaling process ensures that solutions in a location are close to each other.



(a) *can144*.



(b) *path200*.

Figure 6.12: Sampled solutions mapped to the Euclidean 3D space by applying multidimensional scaling. Better fitness values (smaller CBS) are colored in dark violet.

Figure 6.12 exemplifies the results obtained by the modeling and partitioning steps, based on multidimensional scaling, for two instances, *can144* and *path200*. The fitness of each point in these plots corresponds to the average of the fitness values for all the solutions that share the same Euclidean position. A color map is used to represent with dark violet, solutions with small (better) fitness values, and with light yellow tones those having large fitness costs. This kind of visualization, previously used in [66, 98], is useful to reveal the fitness variations around specific known solutions, such as the best-found. However, they lack the trajectory component to reveal how the transitions among solutions occurred through the search process.

For instance, in Figure 6.12(b) there is clearly different zones of the search space which concentrate reduced groups of solutions with small fitness values (dark violet points, at the right), while there is a cluster of yellow points (i.e., large fitness cost points, at the left) surrounded by medium fitness cost points. The former represent a zone of the search space less frequently visited by the algorithm, while the latter one seems to be a more accessible region for the algorithms. From these plots, nonetheless, it is not possible to identify if the analyzed algorithm has followed a particular search trajectory connecting these two distant zones of the space. As we will see the STNs, presented next, represent a better alternative visualization that overcomes this drawback.

6.4.2 Metrics for analyzing search trajectory networks

As previously mentioned, in addition to the visual representation of the resulting STN model, its characteristics are quantified by examining it with a series of network metrics that help to summarize the main aspects of its structure, as well as to compare the portions of merged STN belonging to different metaheuristics. The network metrics employed in the STN analysis are the following.

- **nodes**: Number of nodes
- **edges**: Number of edges
- **end nodes**: Number of unique nodes without any edge going out from them. They represent the end of a search trajectory.
- **best nodes**: Number of unique nodes with a fitness equal to the best-found during the search trajectory.
- **in-strength**: Ratio between the sum of weighted incoming degree for best nodes and the sum of weighted incoming degrees for all end nodes.
- **shared nodes**: Number of nodes visited by more than one algorithm.
- **visited nodes (v. nodes)**: Number of nodes visited by algorithm A_i.
- **shortest path length (s. path)**: Length of the shortest path to the best node visited by algorithm A_i.

- **number of shortest paths (n. paths)**: Number of shortest paths to the closest best node visited by algorithm A_i.

The STN were plotted in R by employing the igraph package [24] and the force-directed layout algorithms Kamada-Kawai [63] and Fruchterman-Reinold [36]. Both layout algorithms try to create displays of the network that have:

- a roughly even distribution of vertices
- minimized crossings of edges
- approximately uniform length edges
- preservation of inherent symmetries, in such a way that similar subnetworks are depicted similarly as well.

6.4.3 Search trajectory networks results

The STN presented in this section were created by recording the best-found solution at a given iteration of the DMAB+MA and MA-20 algorithms, with a sample frequency of $5n$. The algorithms were run by 50,000 generations, with 10 runs per instance, which is a commonly standard number of runs for this type of technique [89, 91]. Previously generated information about the best CBS value that each algorithm was able to produce for each instance was employed to end their execution at the point where that cost is reached for the first time. Since the size of the graphs was a challenge for the partitioning method, the instances were 24 graphs of various topologies, with order $24 \leq 200$ and edges $68 \leq e \leq 2000$. This challenge is further discussed ahead.

Table 6.6 shows algorithm specific results of the STN analysis for the reduced group of instances. It presents the instance name, its order, size and the value of the optimum/best-known solution. The rest of the columns describe the portion of the merged STN belonging to each of the algorithms using the STN metrics previously described: the number of nodes, length of the shortest path to

Table 6.6: STN metrics produced by MA-20 and DMAB+MA algorithms over the complete set of selected instances.

Problem instance				DMAB+MA			MA-20		
name	$\|V\|$	$\|E\|$	opt/b-k	v. nodes	s. path	n. paths	v. nodes	s. path	n. paths
path100	100	99	99	304	11	9	376	6	9
path200	200	199	199	378	19	10	214	11	8
cycle100	100	100	100	343	3	10	363	15	10
cycle200	200	200	200	260	8	10	218	8	8
wheel100	100	198	2600	60	2	31	383	1	24
wheel200	200	398	10200	101	5	21	223	2	11
p9p9	81	144	516	89	0	88	263	1	88
c9c9	81	162	873	231	1	110	291	1	110
k9k9	81	648	8280	135	0	104	179	2	82
p9c9	81	153	745	82	1	112	223	0	80
p9k9	81	396	1728	561	3	101	141	2	91
c9k9	81	405	1809	252	1	126	117	0	109
cyclePow100-2	100	200	300	266	2	10	303	2	10
cyclePow100-10	100	1000	5500	32	2	9	103	0	8
cyclePow200-2	200	400	600	239	1	10	226	1	9
cyclePow200-10	200	2000	11000	34	2	9	114	1	9
can24	24	68	182	37	3	18	57	0	12
ibm32	32	90	405	78	1	82	217	0	73
curtis54	54	124	411	535	2	80	256	1	80
will57	57	127	335	393	0	90	222	2	90
ash85	85	219	913	280	2	54	304	1	54
nos4	100	247	1031	56	0	30	69	0	24
can144	144	576	1776	332	3	92	214	4	42
average				220.78	3.13	52.87	220.70	2.65	45.26

Table 6.7: Structural metrics for merged STN produced over the complete set of selected instances.

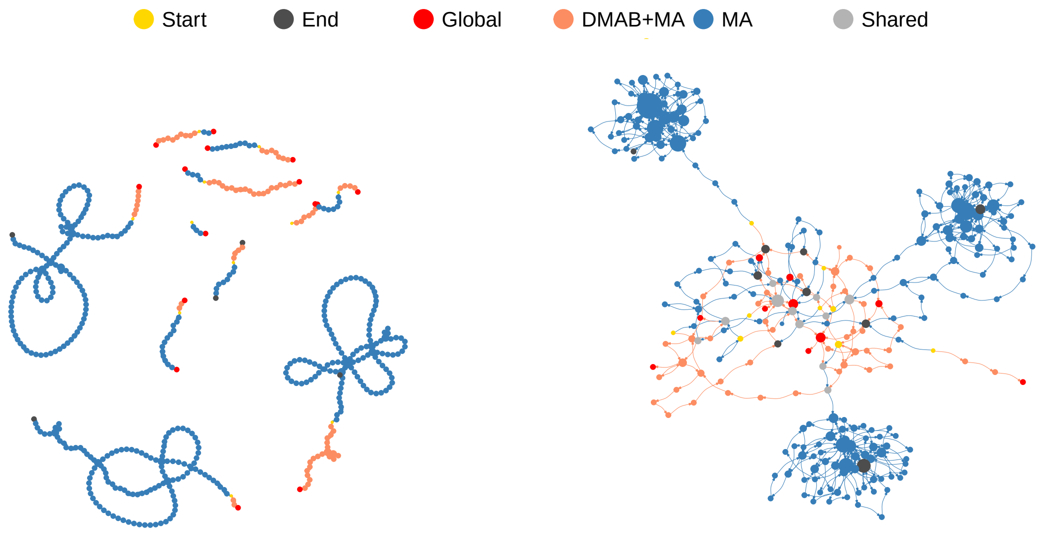
Instance	nodes	edges	shared nodes (%)		best nodes	end nodes	in-strength best	in-strength end
path100	654	706	26	(3.98)	1	9	0.524	0.476
path200	571	606	21	(3.68)	1	13	0.318	0.682
cycle100	683	739	23	(3.37)	1	9	0.550	0.450
cycle200	456	483	22	(4.82)	1	13	0.350	0.650
wheel100	429	432	14	(3.26)	11	9	0.550	0.450
wheel200	308	314	16	(5.19)	7	11	0.429	0.571
cyclePow100-2	551	578	18	(3.27)	1	10	0.476	0.524
cyclePow100-10	124	135	11	(8.87)	1	3	0.850	0.150
cyclePow200-2	448	461	17	(3.79)	1	12	0.400	0.600
cyclePow200-10	137	142	11	(8.03)	1	6	0.700	0.300
p9p9	303	450	49	(16.17)	11	9	0.588	0.412
c9c9	466	589	56	(12.02)	11	9	0.556	0.444
k9k9	295	315	19	(6.44)	14	6	0.704	0.296
p9c9	280	322	25	(8.93)	13	6	0.679	0.321
p9k9	653	851	49	(7.50)	11	9	0.351	0.649
c9k9	338	392	31	(9.17)	15	3	0.828	0.172
can24	83	80	11	(13.25)	15	2	0.900	0.100
ibm32	269	496	26	(9.67)	10	9	0.404	0.596
curtis54	646	1708	145	(22.45)	8	12	0.159	0.841
will57	529	1049	86	(16.26)	9	10	0.294	0.706
ash85	493	974	91	(18.46)	6	13	0.395	0.605
nos4	114	109	11	(9.65)	18	2	0.905	0.095
can144	520	551	26	(5.00)	11	9	0.500	0.500
average	406.52	542.70	34.96	(8.60)	7.74	8.43	0.540	0.460

the closest end node (s. path) and the number of paths of such length (n. paths). Shortest paths of length zero correspond to cases where the initial solution and the final one solution were close enough to be mapped into the same location by the search space partitioning method. These type of transitions would be represented as loops, however, for the sake of a clearer STN visualization, loop edges are excluded.

The structural metrics for the merged STN are introduced in Table 6.7. It includes the number of nodes and edges of the STN, the number of shared nodes, best nodes and end nodes. The strength for end nodes is marked in bold when it is equal or larger than the in-strength for best nodes. This can be considered as a difficulty indicator, since it implies that the search trajectories could be diverted to locations of inferior quality.

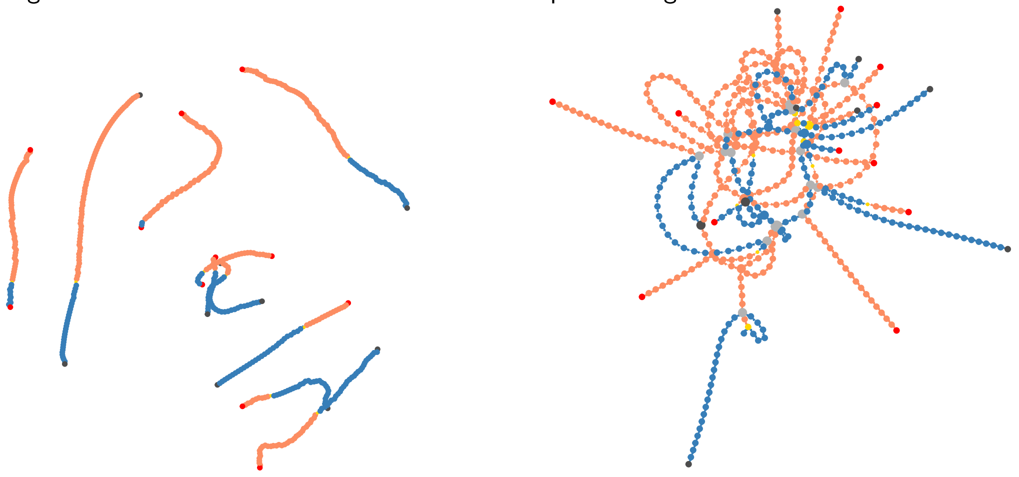
Figures 6.13 and 6.14 present a comparison of the merged STN before and after applying the partitioning method, for instances *p9p9*, *path200*, *can144* and *ibm32*. Before the partitioning, the STN locations contain only one single solution each. In most of the occasions, the absence of a search space partitioning method results in disjointed search trajectories that do not revisit locations. After using the multidimensional scaling to partition the search space, solutions close to each other were mapped into the same location, allowing to reveal the distinct patterns of visit of both algorithms around certain regions. Often, the DMAB+MA can be observed to avoid areas where MA-20 revisited constantly, as in Figures 6.13(b) and 6.14(d).

The STN of Figure 6.13(b) has three densely connected subgraphs that contain the largest nodes of the network, denoting locations that were visited several times by MA-20. There is only one path going out of each of these subgraphs, suggesting it is very difficult for MA-20 to abandon the locations in those regions of the search space, probably due to the presence of local optima with large attraction basins. The STN has shared nodes, visited by both algorithms, that are part of incoming paths towards those subrgaphs, but when DMAB+MA visited them, it took a different path than MA-20, which eventually lead to best nodes. In the central area of the STN there are best nodes with outgoing edges towards other nodes, meaning that the algorithm visited a location



(a) STN for instance *ibm32*, without partitioning.

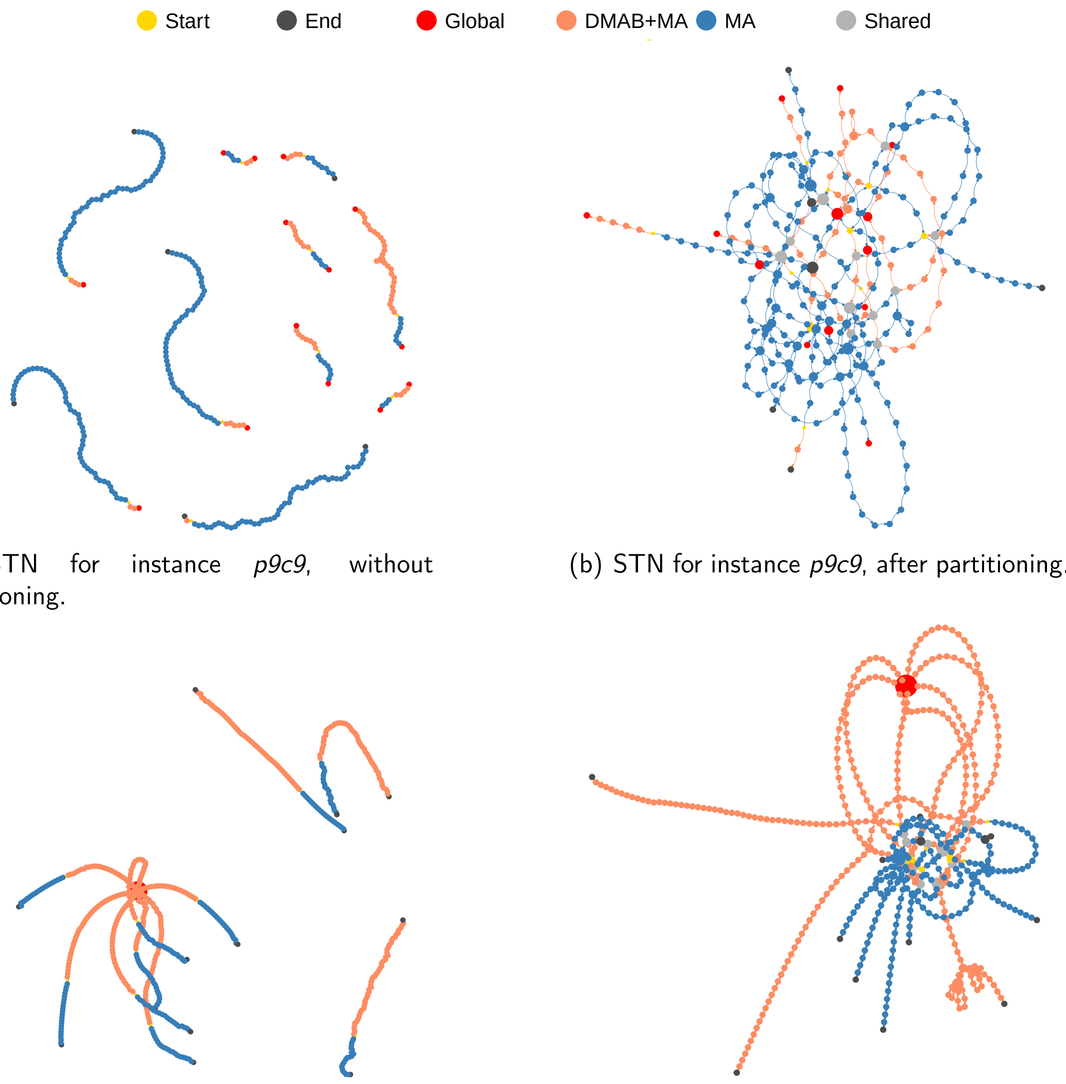
(b) STN for instance *ibm32*, after partitioning.



(c) STN for instance *can144*, without partitioning.

(d) STN for instance *can144*, after partitioning.

Figure 6.13: STN before and after applying partitioning for two representative instances: *ibm32* and *can144*.



(a) STN for instance *p9c9*, without partitioning.

(b) STN for instance *p9c9*, after partitioning.

(c) STN for instance *path200*, without partitioning.

(d) STN for instance *path200*, after partitioning.

Figure 6.14: STN before and after applying partitioning for two representative instances: *p9c9* and *path200*.

that contains a solutions with the best-known cost. However, despite being close to that solution, it did not found it. Most of this cases, also observed in the STN in Figure 6.14(b), correspond to MA-20, while in cases corresponding to DMAB+MA, the paths it took eventually lead to a different best node.

In the STN for instance *can144*, shown in Figure 6.13(d) there are some shared nodes, but the two algorithms take distinct paths out of these nodes, with only DMAB+MA eventually reaching best nodes. The STN subgraph for MA-20 has fewer nodes and its search trajectories are shorter, rarely ending in best nodes, showing that the best-found solution was visited early in the search. Another relevant observation is that the end nodes and best nodes seem to be distributed in different search space regions and that does not seem to be close to each other.

For instance *path200*, the STN contains only one best node, even before the partitioning process, corresponding to the location of a known optimum. Only seven of the DMAB+MA executions were able to converge towards this optimum. The post-partitioning STN shows that the trajectories of MA-20 and one of the DMAB-MA trajectories that did not reached the optimum have several cycles, suggesting hard to escape regions.

The STN analysis revealed that the search space for most of the CBSP considered instances contains multiple optimal/best-known solutions that are sparsely distributed across the fitness landscape. It also helped to identify that certain regions traversed by both studied algorithms are close to high quality areas, but it is usual that only the DMAB+MA has the ability to reach them, while the MA instead gets to worse quality end nodes. The evidence from this study helps to demonstrate that DMAB+MA is more efficient for conducting the search process, as observed in its shorter trajectories, avoidance of areas where MA-20 gets trapped, shorter, often more numerous, paths to the optimal/best-known and number of hits.

From the STN analysis, it is possible to infer information about the fitness landscape of the different instance topologies and their difficult. Instances from the path, cycle and power of cycle topologies had STN with single best nodes. Since the optimum is known for these topologies, those nodes are known to be locations containing optimal solutions. The in-strength metric for the best nodes can be an indicator of how hard is to reach the best solutions. For the path and cycle topologies, the in-strength for the best nodes becomes smaller for the larger instances. This suggest that the fitness landscape for those topologies has very few optimums. Meanwhile, for the power of cycle graphs, the in-strengh of best nodes is higher than that of the end nodes, and it increases for instances with more edges. For this reason, the path and cycle topologies could be considered harder. Something similar was observed for the wheel topology, showing higher in-strength of the best nodes as the number of edges grew, as well as a larger number of best nodes and more paths towards them.

Table 6.7 shows that several Cartesian product graphs (*p9p9*, *c9c9*, *k9k9*, *p9c9*, *p9k9*, and *c9k9*) have a higher percentage of shared nodes, which could hint at the presence of fitness landscape structures that conduct to high quality areas. The Cartesian products also had some of the highest numbers of best nodes, which were easier to reach by many paths leading to them, and consequently fewer end nodes. Something similar was observed on some of the Harwell-Boeing graphs, such as *can24* and *nos4*. However, this subset is more heterogeneous, consisting of disperse matrices from problems in diverse engineering areas. Because of this reason, they vary more, with instances like *curtis54* and *will57* with very small in-strength for best nodes, or *can144*, with equal in-strength for best nodes and end nodes.

6.5 Conclusions

This chapter explored three main aspects: the impact of the proposed alternative evaluation function on the fitness landscape of the problem, the relationship of fitness landscape features with instance characteristics and problem difficulty, and the search dynamics of MA-20 and DMAB+MA and what can they reveal about the challenges of the problem. For the first aspect, the fitness landscape was analyzed for both functions. The second aspect was addressed by studying the interplay between a group of selected fitness landscape features, graph metrics and the O-RMSE values with exploratory factor analysis. Finally, the third aspect consisted in modeling the search patterns of the algorithms as search trajectory networks to examine and compare them.

The results of the fitness landscape analysis shown that the alternative evaluation function significantly reduces the neutrality and the number of local optima, leading to larger descent distances in the ILS algorithm that ended in lower cost solutions. The overall cost of the local optima was also improved. While the number of global optima was reduced as well for several of the considered instances, this reduction was not proven significant. Meanwhile, the autocorrelation analysis did not demonstrate a significant variation between the decay in the correlation for solutions along random walks for the functions. The correlation between distance to the nearest global optimum, however, showed FDC values theoretically associated with more deceptiveness for function f_3. This was attributed to the reduced set of global optima causing the distances to the nearest one to become larger. The evolvability was analyzed with fitness clouds and the negative slope coefficient calculated from them. The results did not showed significant differences between the two functions. It was therefore concluded that, besides the neutrality reduction, and the marginal increase in the descent distances, most of the other aspects of the fitness landscape remain, overall, undisturbed by the alternative evaluation function.

Before the application of the exploratory factor analysis, the set of fitness landscape features, initially consisting of observations for both functions, was reduced with base in three criteria: their significant difference, correlation and practicability. Fitness landscape features that were not proven to have different values for function f_3 were represented only with their values for the original evaluation function, which can be assumed to be representative enough for both. The fitness landscape features that required a priory knowledge of the set of global optima were discarded. A correlation analysis between the remaining fitness landscape features showed that they relate to each other in the same way for both evaluation functions. After identifying highly correlated features, four fitness landscape features were selected: neutrality ratio (for both functions), autocorrelation coefficient, descent distance and negative slope coefficient. The graph metrics were also subject to an elimination process based on correlation, keeping 10 out of the initial 26. Together with the fitness landscape features, this resulted in 14 features.

Principal component analysis was employed for assessing the number of factors to extract with exploratory factor analysis. The number of factors was set to seven. The factors that resulted the most interesting, given their correlation with problem difficulty, were factors F1, F3 and F7. Factor F1 had large loadings for the order of the graph, the autocorrelation coefficient and descent distance. It was also the one that correlated the most with problem difficulty, expressed as the algorithm performance of MA-20 ad DMAB+MA in terms of RMSE values. Since it suggested large gradients that improve steadily, it was conjectured that it may be associated with the size of the attraction basins, which would be larger and harder to escape, for larger instances. The correlation with algorithms seems to support this. Factor F3 was influenced by neutrality ratio and number of endpoints. It had small correlation with the performance of algorithms, specially in the case of DMAB+MA. If this factor expresses the size of plateaus, this would explain its smaller correlation with DMAB+MA performance, since this algorithm has function f_3 available. Factor F7 seems to suggest that graphs with higher eccentricity result more difficult to MA-20, while DMAB+MA performance appeared uncorrelated to this factor. The remaining factors, F2, F4, F5 and F6 were observed to reflect relationships between graph metrics, with few relation to the algorithm's performance or fitness landscape features.

The final part of the chapter focused on modeling and analyzing search trajectory networks for DMAB+MA and MA-20. A new search space partitioning technique, based in multidimensional scaling, was developed in order to create better STN models that reveal more information about the search space regions each algorithm visited. The STN analysis created a visual representation of search dynamics, where it was possible to observe how DMAB+MA avoids areas where MA-20 is trapped. Such areas are likely to contain multiple local optima with large attraction basins. The STN provided a visual way to determine if best-known solutions/optimums are grouped together or disperse, finding this depends on the instance. The strength of the best nodes served as an indicator of instance difficulty, On average, the DMAB+MA had trajectories that ended more often in the best-known/optimal solutions and more paths conducting towards them. The proposal of the multidimensional scaling as a method to grouping several solution into locations is also a contribution to the development of the STN technique and its application for problems with discrete and combinatorial search spaces.

The next chapter presents an exact approach for solving the CBSP in relatively small instances, by modeling it as a constraint programming problem. The initial model was iteratively refined by employing information about the value of the optimum for the different instance topologies and the introduction of suitable priorities for the exploration. The final refined model is then compared with a branch and bound algorithm that was developed specially for the CBSP, as part of this work.

7

A constraint programming model for the CBSP

There are several factors that influence the election between exact and approximated methods as the chosen alternative to deal with an optimization problem. Additionally to the problem itself, these factors can include the instance size, the resources available, such as time and computational processing power, and even the purpose of a given application. The approximated methods, among which figure the heuristics, metaheuristics and hyperheuristics, represent a compromise between the amount of resources invested and the quality of the solution. While the solutions produced by approximated methods can not be guaranteed as optimal, they are often considered as *good enough* solutions, making the use of approximated methods a very practical approach, particularly for large instances of a problem and non critical applications. The exact methods, instead, can guarantee that the solution produced is optimal, but the resources they require to verify this are considerable more demanding than the ones of an approximated method, because they rely on the implicit enumeration of the search space. For this reason, in general, they are not practicable for problem instances of any size. In the adequate conditions, however, the exact methods can result in effective approaches to produce optimal results. In this work, the development of an exact method was explored with purposes that include obtaining the optimal solutions themselves, but also extend further beyond. An assessment of how the exact methods perform and which sizes and types of instances can be managed within a certain time budget is another step in the study of the CBSP difficulty in relation with solution approaches. The values of the optimal solutions also serve as a reference to compare with theoretical bounds and estimate how wide the gap between these values can be. This chapter presents a study on the use of constraint programming [94] on relatively small instances of the problem, with $n \leq 40$.

Section 7.1 presents an brief overview of the constraint programming approach. An initial model and its refinements are described in Section 7.2. The B&B implementation and the comparison of its performance with the final CP model are presented in Section 7.4. Finally, Section **??** summarizes the conclusions and further work.

7.1 Constraint programming

The constraint programming (CP) paradigm is an effective alternative to solve satisfiability and optimization problems, particularly combinatorial ones. It is based on a declarative approach, where the features of a problem and the conditions it solutions must met are described, as opposed to specifying the steps for creating the solutions. This descriptive stating of the problem is refereed as a CP model, consisting of the decision variables involved, their domains and the constraints among them. A CP solver program produces solutions that met the specifications of the model by selectively exploring the search space, i.e., all the different ways the decision variables can be assigned. Solvers are usually part of a larger software suite that provides also other tools for CP implementation, including modeling languages, compilers and profilers. Internally, a solver explores the search space by using search trees, constraint propagation and backtracking with aid from techniques of areas like artificial intelligence, mathematical programming, and operational research.

The constraint propagation is the distinctive key process in constraint programming. Through constraint propagation, the values whose assignation to a variable would break a constraint are eliminated from its domain and this information is employed to try eliminating values from the domains of the other variables involved in the same constraint [130]. The recursive process of constraint propagation has been abstractedly described as a network [94], where nodes represent decision variables. The nodes are labeled with the variable's domain values and variables involved in the same constraint have their nodes connected by an hyperedge. If the assignation of a value to a variable breaks a constraint, the solver's internal CP algorithm removes that value from the domain, which would modify the label of the node corresponding to that variable. The modification triggers a process where other hyperedges incident to the node are inspected, in order to determine how the change in a variable's domain may have affected the other constraints involving it. This can cause further modifications in the labels of other nodes, and it ends if all modifications have been processed or an empty label has been obtained.

7.2 Modeling the CBSP

A relevant aspect of the modeling process is to determine how the characteristic's of the problem in question are translated into constraints. There are different ways on which a constraint can be expressed, and these variations can affect how the model performs. For this reason, the modeling is often an iterative process, where additional constraints can be added or modified. Another way for trying to improve a model's performance through refinement is to vary the search strategy, which defines the order for assigning variables and exploring their domains. Along this section, an initial CP model for the CBSP is described and iteratively refined by adding new constraints based on the instance topologies, and the cost relationship between the CBS the bandwidth sum (BS).

7.2.1 An initial CBSP model

The initial model is quite simple. It states that the solutions must be bijective mapings between guest nodes and host nodes, how cyclic distances are measured and the problem's objective function. In the initial model and its subsequent refinements, only the CBSP traditional evaluation function was employed.

7.2.1.1 Data representation

The required input data consisted of finite, simple, undirected graphs. Each graph is represented in a standard format, with the .dzn extension for input files in Minizinc [86]. The input data specifies:

- the number of vertices n
- the number of edges e
- a 2-D array $E(1..e, 1..2)$ symbolizing the list of edges, where $E(i, 1)$ and $E(i, 2)$ are the endpoints of the i-th edge.

Further refined versions of the model require some additional data that is introduced in the corresponding sections.

7.2.1.2 Variables

The decision variables of the model are the labels assigned to each vertex, which represent an embedding as the series of associations between guest vertices V and host vertices V'.

- $g(1..n)$ is an array of the labels assigned to vertices, where $g(i)'$ is the label mapping guest vertex $i \in V$ to host vertex $g(i) \in V'$.

7.2.1.3 Constraints

In the CBSP, the embeddings are bijective mappings between guest and host vertex sets. In order to capture this relationship, it is necessary to express that there must be a unique host vertex for each guest vertex, and vice versa.

- To each unique guest vertex corresponds one unique host vertex as label.

$$\forall i, j \in [1..n]\,|i < j \qquad g(i) \neq g(j) \tag{7.1}$$

This constraint would be equal to a series of pairwise conjunctions stating that no pair of vertices can have the same label, in the form $g(1) \neq g(2) \wedge g(1) \neq g(3) \wedge \cdots \wedge g(1) \neq g(n) \wedge \cdots \wedge g(n-1) \neq g(n)$.

Large conjunctions like this can be costly to compute and the process of implementing them in an efficient way can be demanding. Moreover, the efficiency of a particular implementation can be tied to a specific solver. For these reasons, modern modeling languages and solvers usually implement instead global constraints as customized efficient algorithms based on inferences. Global constraints are concise ways to express relationships among several variables, such as constraint (7.1), thus improving performance and simplifying the modeling process. In terms of representation, despite the semantic redundancy they add to models (as they can be decomposed into simpler constraints), they provide a higher level of abstraction, facilitating modeling. In terms of reasoning, global constraints provide a better structure to the problem, allowing the filtering algorithms to be much more specialized and efficient [Beldiceanu].

In all the models, the well-known global constraint $alldifferent$, stating that all elements in an array must be pairwise distinct, is used instead of constraint (7.1).

$$alldifferent(g) \tag{7.2}$$

7.2.1.4 Objective function

The cost of a solution is given by the sum of cyclic distances.

$$cbs = \sum_{i=1}^{e} distance(i) \tag{7.3}$$

where $distance(1..e)$ is an array of cyclic distances; and $distance(i)$ represents the cyclic distance associated to edge i. Each cyclic distance can have a value between 1 and $d_{max} = \lfloor n/2 \rfloor$.

Cyclic distances for each edge of G are equal to the length of the shortest path between two adjacent vertices of the guest graph embedded in the host graph.

$$\forall i \in [1..e] \qquad distance(i) = \min\left[n - g(E(i,1)), |n - g(E(i,2))|\right] \tag{7.4}$$

The goal of the CBSP is to find the lowest cost embedding, therefore the objective function for the model is:

- minimize the sum of cyclic distances

$$minimize(cbs) \tag{7.5}$$

The first CBSP model, called M_0, is defined by the previously described variables, as well as the conjunction of the constraints and the objective function.

$$M_0 = (7.2) \wedge (7.5)$$

7.2.2 Breaking cyclic symmetries

Since the problem consists in embedding a graph into a cyclic topology, there are label assignations that result in isomorphic embeddings under rotation and mirror symmetries. In order to remove those solutions from the search space, two additional constraints were added to the basic model M_0. These new constraints ensure that only the first apparition in lexicographical order of the isomorphic embeddings is computed.

- The first vertex must be associated to the first label, thus eliminating the rotation symmetry.

$$g(1) = 1 \tag{7.6}$$

- The label of the second vertex must be lower than the label of the last, thus eliminating the mirror symmetry.

$$g(2) < g(n) \tag{7.7}$$

The first refined model, M_1, results from adding the symmetry breaking constraints to the initial model, thus: $M_1 = M_0 \wedge (7.6) \wedge (7.7)$.

7.2.3 Adding upper and lower bounds

When solving optimization problems using CP, the inclusion of cost bounds can improve the performance by helping to discard suboptimal solutions with cost outside the bounds. Previous results from the CBSP literature, presented in Section 2.3.1, include formulas for obtaining the exact value of the optimum or bounding it in function of the graph topology. These results were employed to implement new constraints for the cost of solutions. The data representation previously introduced was modified to include two new input variables:

- ub, the value of the CBS upper bound
- lb, the value of the CBS lower bound

In the case of graph topologies for which there are exact formulas to calculate the value of the optimum, both ub and lb got assigned that value. This includes the path, cycle, wheel and power of cycle topologies. If the value of the optimum can not be calculated in an exact way, the value of ub was calculated according to topology specific upper bound formulas, in the case where those exist, or the topology independent upper bound formula, otherwise. The first case corresponds to the Cartesian product topologies, while the later applies to the rest of the graphs: Harwell-Boeing, random graphs, Möbius ladder and triangulated meshes. Meanwhile, the value of lb was set as $e+1$ for all topologies.

The following are the constraints related to the lower and upper bounds. Notice that in the case where the exact value of the optimum is known, then $ub = lb$ and the constraints still hold.

$$cbs \geq lb \tag{7.8}$$

$$cbs \leq ub \tag{7.9}$$

Model M_2 results from adding the upper and lower bound constraints to model M_1, therefore $M_2 = M_1 \wedge (7.8) \wedge (7.9)$.

A constraint related to a lower bound for the CBS in relation to the BS was added in a further refined model. Recall the BSP [115] is a GEP similar to the CBSP, with the difference that the topology of the host graph is a path instead of a cycle. There is a bounded relationship between the value of the optimum of both problems [18], given by the expression:

$$\frac{BS}{2} \leq CBS \leq BS \tag{7.10}$$

Equation 7.11 is a topology independent lower bound for the BS [6].

$$BS \geq (a+1)e - \frac{a^2(2n-a-1)}{2} \tag{7.11}$$

where $a = 1/3(2n - 1 - \sqrt{(2n-1)^2 - 6e})$.

Based on equation 7.11, the following lower bound constraint was added to the model, resulting in model $M_3 = M_2 \wedge (7.12)$

$$cbs \geq \frac{\left((a+1)e - \frac{a^2(2n-a-1)}{2}\right)}{2} \tag{7.12}$$

7.3 Experimental results

The first assessment of the models compares the initial model M_0 and the incremental refinements to introduce symmetry breaking constraints in model M_1, lower and upper bounds in model M_2 and the additional lower bound related to the BSP in model M_3. For these experiments, the models were compiled with Minizinc 2.4.3 [86] and solved using integer search with the default built-in solver Gecode 6.1.1 [119]. For each execution, the time limit was 1 hour (3,600 seconds). The default strategy to assign values to decision variables was to choose the variable with the smallest domain size (*first_fail*), and then try to assign first the minimum value in the domain (*indomain-min*).

The instance set for the experiments in this chapter includes topology diverse graphs that have $n \leq 40$ vertices. The graphs are divided into the following subgroups, accordingly to the available knowledge about the value of the optimum.

- **topology-specific optimum.** It includes graphs for which the optimal value can be easily calculated (see Section 2.3.1.2). The topologies are paths, cycles, wheels, complete bipartites and k-th powers of cycle. For each topology, there are graphs of order $n \in \{10, 15, 20, 25, 30, 40\}$. In the case of the complete bipartite graphs, the vertices were distributed in two subsets of $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ vertices each. The graphs from the power of cycle topology were built with powers $k \in \{2, 3\}$.

- **topology-specific upper bound.** This group contains the Cartesian products of two graphs that belong to the path, cycle or complete topologies. While the value of the optimum is unknown, there are upper bounds reported for each of the resulting Cartesian products (see

Section 2.3.1.3). The experiments considered the Cartesian product of graphs $G_{n_1} \times G_{n2}$, with $3 \leq n_1, n_2 \leq 8$, $n_1 \geq n_2$ and $n_1(n_2) = n \leq 40$.

- **topology-independent upper bound.** This is the most heterogeneous of the subgroups. For these graphs, the CBS topology-independent upper bound formula was employed as reference. The topologies included in this subgroup are Möbius ladder, triangulated triangles, random graphs, and Harwell-Boeing. The Mobious ladder graphs had order $n \in \{12, 14, 16, 20, 24, 30, 40\}$. All the triangulated triangles with $3 \leq n \leq 40$ vertices were included, resulting in graphs of order $n \in \{10, 15, 21, 28, 36\}$. The random graphs had order $n \in \{10, 15, 20, 25, 30, 40\}$, with a probability of connection between vertices of $p \in \{0.5, 0.7, 0.9\}$. There are few graphs in the Harwell-Boeing collection that met the criterion of having $n \leq 40$ vertices. Because of this, the HB graphs in the experiments were only *jgl009*, *can24*, *pores1*, *ibm32* and *bcspwr01* with 9, 24, 30, 32 and 39 vertices, respectively.

7.3.1 Comparison for the initial model and the refined versions

This section presents a topology based analysis of the largest graphs that the models were able to solve within the time budget. This is an indicator of how challenging the different topologies can result for the models. In this analysis, the theoretical upper and lower bounds were used to evaluate the ability of the model to achieve solutions within these bounds and to finish the execution before the set time budget was spent. Otherwise, even if the reported solution was indeed optimal, it could not be recognized as such and the graph was not considered as properly solved by the model.

Tables 7.1 and 7.2 show the results for the initial model and the three incrementally refined versions of it. The instances in Table 7.1 belong to the first of the subgroups of instances previously introduced, the graphs for which the value of the optimum is calculated by topology-specific formulas. The other two subgroups, for which this is not the case, are presented in Table 7.2. For briefness, only the instances that were actually solved by at least one of the models are included in this comparison.

For each instance, their order n, size e, density (den), optimal value or lower and upper bounds are listed. The metrics for the performance of the models consider the best solution cost reported and their execution time.

Regarding the topologies with known optimal value, the initial model M_0 was able to solve all the graphs in the path and cycle typologies, but it failed to solve the wheel graphs over 10 vertices and the powers of cycle over 25 vertices and 50 edges. As for the rest of graphs, the initial model also failed to solve several of the Cartesian products, like those of more than 15 vertices and 25 edges, the Möbius ladder graphs with more than 16 vertices, triangulated triangles of more than 15 vertices, the random graphs of more than 10 vertices and it only solved the smallest Harwell-Boeing graph, with 11 nodes.

The incrementally refinements over the initial model M_0 helped the models M_1, M_2 and M_3 to solve increasingly larger problem sizes, while also reducing the execution time, specially for instances where the running time was closer to exceeding the predefined one hour budget. The removal of isomorphic solutions by the constraints to break mirror and rotation symmetries in model M_1 seems to have been effective in reducing the search scope. This helped model M_1 to solve all the graphs that model M_0 did, plus eighth larger ones, in the wheel, Cartesian products and Möbius ladder topologies. Model M_2, which added the upper and lower bounds constraints, solved nine more graphs when compared to model M_1, for a total of 57 solved graphs. These nine graphs were the largest graphs in the wheel and power of cycle topologies, previously unsolved by both models M_0 and M_1. In terms of solved instances, model M_3 only solved one additional larger instance with respect to model M_2, the Cartesian product graph *c5c3*. However, model M_3 reduced considerably the execution time, for example, by over 1,000 seconds for instance *p4c4*.

Table 7.3 summarizes the performance of the models in terms of the largest graphs that they were able to solve with respect to the largest graph in that topology. The path, cycle, wheel and power of cycle topologies were the only ones for which all the considered graphs were solved, by models M_2 and M_3. Regardless of model, only the smallest of the complete bipartite graphs, *bip5-5*, could be

Table 7.1: Performance comparison of the original CP model M_0 and the refined versions M_1, M_2, M_3 for instances with known optimum.

					M_0		M_1		M_2		M_3	
Graph	$\|V\|$	$\|E\|$	den	Op^*	*Best*	*T*	*Best*	*T*	*Best*	*T*	*Best*	*T*
path10	10	9	0.20	9	**9**	0.37	**9**	0.65	**9**	1.04	**9**	2.56
path15	15	14	0.13	14	**14**	0.39	**14**	0.49	**14**	0.38	**14**	0.27
path20	20	19	0.10	19	**19**	0.38	**19**	0.53	**19**	0.35	**19**	0.25
path25	25	24	0.08	24	**24**	0.38	**24**	0.5	**24**	0.41	**24**	0.25
path30	30	29	0.07	29	**29**	0.41	**29**	0.51	**29**	0.39	**29**	0.25
path40	40	39	0.05	39	**39**	0.45	**39**	0.55	**39**	0.37	**39**	0.27
cycle10	10	10	0.22	10	**10**	0.36	**10**	0.66	**10**	0.79	**10**	0.34
cycle15	15	15	0.14	15	**15**	0.3	**15**	0.62	**15**	0.46	**15**	0.26
cycle20	20	20	0.11	20	**20**	0.35	**20**	0.65	**20**	0.41	**20**	0.31
cycle25	25	25	0.08	25	**25**	0.35	**25**	0.61	**25**	0.43	**25**	0.32
cycle30	30	30	0.07	30	**30**	0.31	**30**	0.69	**30**	0.41	**30**	0.25
cycle40	40	40	0.05	40	**40**	0.41	**40**	1.81	**40**	0.45	**40**	0.26
wheel10	10	18	0.40	35	**35**	4.1	**35**	0.62	**35**	0.67	**35**	0.33
wheel15	15	28	0.27	71	71	1hr	**71**	332.61	**71**	0.4	**71**	0.23
wheel20	20	38	0.20	120	120	1hr	120	1hr	**120**	0.41	**120**	0.37
wheel25	25	48	0.16	181	181	1hr	181	1hr	**181**	0.39	**181**	0.28
wheel30	30	58	0.13	255	255	1hr	255	1hr	**255**	0.4	**255**	0.28
wheel40	40	78	0.10	440	440	1hr	440	1hr	**440**	0.44	**440**	0.25
bip5-5	10	25	0.56	65	**65**	47.58	**65**	2.39	**65**	1	**65**	0.85
cycleP10-2	10	20	0.44	30	**30**	0.39	**30**	1.44	**30**	0.5	**30**	0.49
cycleP10-3	10	30	0.67	60	**60**	9.75	**60**	0.8	**60**	0.34	**60**	0.33
cycleP15-2	15	30	0.29	45	**45**	2.91	**45**	0.69	**45**	0.43	**45**	0.39
cycleP15-3	15	45	0.43	90	**90**	1051.73	**90**	30.62	**90**	0.29	**90**	0.30
cycleP20-2	20	40	0.21	60	**60**	117.87	**60**	1.84	**60**	0.43	**60**	0.35
cycleP20-3	20	60	0.32	120	120	1hr	**120**	926.67	**120**	0.27	**120**	0.34
cycleP25-2	25	50	0.17	75	**75**	819.8	**75**	15.36	**75**	0.43	**75**	0.34
cycleP25-3	25	75	0.25	150	150	1hr	150	1hr	**150**	0.36	**150**	0.29
cycleP30-2	30	60	0.14	90	90	1hr	**90**	231.94	**90**	0.48	**90**	0.4
cycleP30-3	30	90	0.21	180	180	1hr	180	1hr	**180**	0.33	**180**	0.48
cycleP40-2	40	80	0.10	120	120	1hr	120	1hr	**120**	0.51	**120**	0.44
cycleP40-3	40	120	0.15	240	240	-	240	1hr	**240**	0.3	**240**	0.53

solved. Among the Cartesian products, the graphs with lower densities seem to be easier to solve, as demonstrated by the products $P \times P$ (two path graphs) and $P \times C$ (a path graph and a cycle graph) which were solved by model M_3 for larger order and size than the rest of the Cartesian products. The graphs in the Möbius ladder, triangulated meshes, random graphs, and Harwell-Boeing graphs were solved partially, for orders 24, 15, 10 and 9 respectively, by all the models. While the refined model M_3 did not solve more instances in these topologies than the initial model, it managed to achieve the optimum faster. For this reason, and the multiple cases among the wheel, power of cycle and

Table 7.2: Performance comparison of the original CP model M_0 and the refined versions M_1, M_2, M_3 for instances without formula for value of the optimum.

						M_0		M_1		M_2		M_3	
Graph	$\|V\|$	$\|E\|$	den	lb	ub	$Best$	T	$Best$	T	$Best$	T	$Best$	T
c3c3	9	18	0.50	19	39	**36**	2.01	**36**	0.84	**36**	1.07	**36**	0.57
c4c3	12	24	0.36	25	52	**52**	298.18	**52**	15.08	**52**	24.31	**52**	12.72
c5c3	15	30	0.29	31	65	65	1hr	65	1hr	65	1hr	65	2532.77
p3p3	9	12	0.33	13	24	**19**	0.91	**19**	0.86	**19**	0.46	**19**	0.38
p4p3	12	17	0.26	18	36	**29**	5.94	**29**	1.14	**29**	0.9	**29**	0.69
p4p4	16	24	0.20	25	60	**44**	876.02	**44**	70.92	**44**	51.85	**44**	52.38
p5p3	15	22	0.21	23	48	**42**	267	**42**	31.89	**42**	29.24	**42**	28.15
p6p3	18	27	0.18	28	60	55	1hr	**55**	3369.98	**55**	1 560.77	**55**	981.07
p6c3	18	33	0.22	34	123	69	1hr	**69**	2148.73	**69**	2 180.7	**69**	1 326.27
p3c3	9	15	0.42	16	33	**27**	0.83	**27**	0.82	**27**	2.07	**27**	0.42
p3c4	12	20	0.30	21	44	**40**	54.55	**40**	5.89	**40**	2.95	**40**	2.65
p3c5	15	25	0.24	26	55	55	1hr	**55**	936.39	**55**	577.85	**55**	487.35
p4c3	12	21	0.32	22	57	**43**	65.34	**43**	5	**43**	4.02	**43**	2.61
p4c4	16	28	0.23	29	76	64	1hr	**64**	2458.22	**64**	2 558.77	**64**	1 494.42
p5c3	15	27	0.26	28	87	**56**	1 900.63	**56**	156.42	**56**	101.53	**56**	76.52
c3k4	12	30	0.45	31	88	**72**	1 58.78	**72**	76.63	**72**	42.17	**72**	25.3
p3k4	12	26	0.39	27	80	**58**	243.11	**58**	18.12	**58**	16.18	**58**	9.14
mobLad12	12	18	0.27	19	58	**30**	13.75	**30**	1.16	**30**	0.97	**30**	0.63
mobLad14	14	21	0.23	22	79	**35**	159.19	**35**	5.1	**35**	4.96	**35**	2.54
mobLad16	16	24	0.20	25	102	**40**	1 347.07	**40**	35.56	**40**	29.46	**40**	17.9
mobLad20	20	30	0.16	31	157	50	1hr	**50**	1922.61	**50**	1 719.82	**50**	1 132.18
triTriangle10	10	18	0.40	19	50	**32**	2.18	**32**	0.68	**32**	0.39	**32**	0.32
triTriangle15	15	30	0.29	31	120	**62**	1 777.99	**62**	189.49	**62**	168.94	**62**	122.06
rand10-5	10	24	0.53	25	66	**51**	34.03	**51**	1.65	**51**	2.95	**51**	0.94
rand10-7	10	32	0.71	33	88	**77**	19.81	**77**	1.37	**77**	2.61	**77**	1.21
rand10-9	10	41	0.91	42	113	**106**	78.99	**106**	3.54	**106**	4.88	**106**	2.04
jgl009	9	32	0.89	33	80	**75**	10.08	**75**	0.91	**75**	1	**75**	0.49

Cartesian product topologies, where it solved bigger instances than models M_0, M_1 and M_2, model M_3 was considered the best.

7.3.2 Comparing search strategies

Search strategies determine how the solver proceeds in regard to the search space exploration, involving the type of search, the variable choice strategy that determines the order of assignation for the decision variables and constraints over the domain assignation that control the exploration of valid values. An example of a specific search strategy to solve a model is instantiating unassigned

Table 7.3: Largest solved instances per model, $*$ indicates the model solved the largest instance

	Largest graph ($*$)			M_0			M_1			M_2			M_3		
Topology	G	$\|V\|$	$\|E\|$	G	$\|V\|$	$\|E\|$	G	$\|V\|$	$\|E\|$	G	$\|V\|$	$\|E\|$	G	$\|V\|$	$\|E\|$
path	path40	40	39	$*$			$*$			$*$			$*$		
cycle	cycle40	40	40	$*$			$*$			$*$			$*$		
wheel	wheel40	40	78	wheel10	10	18	wheel15	15	28	$*$			$*$		
bip. complete	bip20-20	40	400	bip5-5	10	25	bip5-5			bip5-5			bip5-5		
power of cycle	cycleP40-3	40	120	cycleP25-2	25	50	cycleP30-2	30	60	$*$			$*$		
C. product $C \times C$	c8c5	40	80	c4c3	12	24	c4c3			c4c3			c5c3	15	36
C. product $P \times P$	p8p5	40	67	p5p3	15	22	p6p3	18	27	p6p3			p6p3		
C. product $P \times C$	p8c5	40	75	p5c3	15	27	p6c3	18	33	p6c3			p6c3		
C. product $C \times K$	c8k5	40	120	c3k4	12	30	c3k4			c3k4			c3k4		
C. product $P \times K$	p8k3	40	115	p3k4	12	26	p3k4			p3k4			p3k4		
Möbius ladder	mobLad40	40	60	mobLad16	16	24	mobLad20	20	30	mobLad20			mobLad20		
tri triangle	triTriangle40	36	84	triT15	15	30	triT15			triT15			triT15		
random	rand40-9	40	705	rand10-9	10	41	rand10-9			rand10-9			rand10-9		
HB	bcspwr01	39	42	jgl009	9	32	jgl009			jgl009			jgl009		
total solved						40			48			57			58

variables according to input order and then try first the lowest value in its domain.

Minizinc [86] allows to communicate search strategies to the solver within the model's code by using special instructions called search annotations. In a strict sense, the annotations specifying search strategies are not part of the model, but the modifications they introduce on how the solver searches can affect the performance. Therefore, testing them was considered relevant.

The tested strategies for variable choice were: *first-fail*, *smallest*, *dom-w-deg* and *input-order*. For each, the constraints over domain assignation *indomain-min* and *indomain-random* were also tested. Table 7.4 presents a description of the search priorities introduced by these search strategies.

Table 7.4: Search strategies for the choosing of variables and domain assignation that were tested.

Variable choice strategy	Priority
first-fail (default)	variable with the smallest domain size
input-order	order of apparition
smallest	variable with the smallest domain value
dom-w-deg	variable with the smallest domain size divided by weighted degree, i.e., the number of times it has been in a broken constraint.
Constraint over domain assignation	Priority
indomain-first (default)	domain values in ascending order
indomain-random	domain values in random order

In order to evaluate the use of the different search strategies introduced in Table 7.4, model M_3 was tested using each of them, under otherwise similar conditions than the previous experiments.

Table 7.5 shows that the number of solved instances varies considerably depending on the search strategy being utilized. The default pairing of *first-fail* and *indomain-min* resulted among the best ones, solving the third highest number of instances. Across all the variable choice strategies, the random domain exploration (*indomain-rand*) seems to produce the worst performance. It resulted in a smaller number of solved instances when compared to the domain exploration in ascending order (*indomain-min*), as few as just 41 solved graphs, which is no better than the initial model M_0. Meanwhile, model M_3 was able to solve the highest number instances by trying to assign first the variable with the smallest value in the domain (*smallest*) instead of the one with smallest domain size (*first-fail*). This results in the smallest host vertices being assigned to guest vertices first, which could be seen as visiting the host vertices in clockwise order to determine which guest vertex fits appropriately in each of them. This relatively simple change allowed to solve six instances more from the Cartesian and Möbius ladder topologies: *c6c3*, *p5p4*, *p3c6*, *p3k5*, *p4k4*, and *mobLad24*. The results for these instances are presented in detail in the following section, which compares the performance of the final model when employing the best search strategy to the model with the default search strategies as well as to an ad hoc branch and bound algorithm.

Table 7.5: Performance summary for M_3 testing alternative search strategies to the *first-fail* search strategy and *indomain-min* domain assignation.

Search strategy		optimal	feasible	avg. time (opt)	avg. time
first-fail (default)	*indomain-min* (default)	58	104	230.53	2584.81
	indomain-random	47	78	181.70	2750.69
input-order	*indomain-min*	59	103	147.21	3316.51
	indomain-random	48	74	139.04	2721.45
smallest	*indomain-min*	**64**	100	227.16	2458.26
	indomain-random	41	72	269.42	2877.94
dom-w-deg	*indomain-min*	54	95	111.16	2603.62
	indomain-random	41	80	82.80	2837.48

7.4 Comparing exact approaches: branch and bound

To the best of our knowledge, currently there is no report of other exact approaches to solve the CBSP. In order to provide a reference method for comparison with the refined CP model, this work proposed an ad hoc B&B algorithm for the CBSP. The main idea behind the B&B methodology is to use a tree structure to implicitly explore the whole search space of the problem by creating partitions of smaller subproblems. The nodes of the search tree contain partially defined solutions to such subproblems. The exploration begins from the root of the tree, and it is conducted by branching promising nodes into new nodes containing partial solutions of higher order and pruning branches that can not lead to the optimum. This process narrows the search, discarding regions of the search space where the optimal solution can not be located.

Algorithm 3 depicts the basic steps of the proposed B&B. The process begins by creating a solution which cost will be used as an initial upper bound during the search. The method for creating this solution is a depth-first-search visit order of the vertices of G, starting in a random vertex x. In the B&B algorithm, the root of the tree is a partially defined solution with the first label assigned to the first vertex. The order of this solution is one, since it has only one assigned vertex. The root solution is inserted into a priority queue that is used to keep track of the exploration. Next, an iterative process of exploring, branching and pruning begins and will stop when the queue is empty, meaning the implicit exploration of the search space has concluded.

When a partial solution b is extracted from the queue, the branching process creates new candidate nodes by assigning each one of the unused labels to the first unlabeled vertex in b. This produces $n - o(b)$ new partially defined solutions, where n is the number of vertices and $o(b)$ is the order of b.

In order to decide if each of these new solutions is going to be explored further or discarded, they have to be evaluated. Since the solutions are only defined partially, this evaluation is composed of a partial CBS calculation, given by the defined part of the solution, and a potential CBS cost, given by the undefined one. The partial CBS is the sum of cyclic distances calculated only for assigned edges, i.e., edges that have labels assigned to both endpoints. For the remaining edges, a simple best-case estimation $f_e(b')$ is calculated. The best-case estimation is equal to the number of unassigned edges, because, in the best case, each unassigned edge could have a cyclic distance equal to one.

Partial solution b is discarded if the sum of partial CBS $f_p(b')$ and the best-case estimation $f_e(b')$

Algorithm 3 Branch and Bound algorithm

```
1: $x \leftarrow$ a random vertex from $G$
2: $up \leftarrow$ dfs$(G, x)$
3: $Q \leftarrow$ empty priority queue
4: Create root solution $a$ by assigning $a(1) \leftarrow 1$
5: Insert $a$ into $Q$
6: while $Q$ is not empty do
7:    $b \leftarrow Q.pop()$
8:    $i \leftarrow$ first unassigned node in $b$
9:    for $j \in$ {unassigned labels in $b$} do
10:       $b' \leftarrow b$
11:       $b'(i) \leftarrow j$
12:       Evaluate partial cost $f_p(b')$ for assigned edges
13:       Estimate potential cost $f_e(b)$ for unassigned edges
14:       if $f_p(b') + f_e(b') < f(up)$ then
15:          if all vertices in $b'$ are assigned then
16:             $up \leftarrow b'$
17:          else
18:             Insert $b'$ into $Q$
19:          end if
20:       else
21:          Discard $b'$
22:       end if
23:    end for
24: end while
25: $g \leftarrow up$
26: return $g$
```

is greater than the cost of the current upper bound solution up. In case the sum is instead lower, and also the order of the solution is equal to the number of vertices of the host graph, b' is a completed solution actually better than the current upper bound solution up, therefore b' replaces up. Otherwise, b' is not a complete solution, but can not be discarded, so it enters the queue.

The priority criterion of the queue defines the exploration order by employing a combination of the order of partial solutions, their partial cost and a more elaborated estimation for the cost of the unassigned edges. Partial solutions of equal order are untied by the sum of their partial CBS and estimation $f_b(b')$. This second form of estimation is heuristic. It calculates the potential cost of assigning the most suitable available label to one of the endpoints of the first found edge that already has the other endpoint labeled.

The idea behind prioritizing order before potential cost is to reach higher order partial solutions and produce completed solutions as soon as possible. Since the CBS is a sum, it is difficult to discard solutions by considering only a few labeled edges, because such solutions are likely to have lower partial costs than the cost of up. If the number of edges with unlabeled endpoints is high, calculating an estimation of the cost for the unlabeled edges is also harder and more likely to be too optimistic. Therefore, by giving priority to higher order partial solutions, the algorithm can reach sections of the search tree that are easier to discard or produce a complete solution that lowers the current upper bound solution up, which would allow to further discard new partial solutions.

7.4.1 Constraint programming against branch and bound results

The B&B algorithm was executed under similar criteria than the CP models, with a time budget of 1 hour (3,600 seconds). Tables 7.6 and 7.7 present a comparison of the results for the B&B algorithm, model M_3 and M_4, which corresponds to the change in search strategy. As previously, the results list only the instances that were solved by at least one of the compared methods.

The B&B algorithm was able to solve most of the graphs with known value of the optimum, except by those in the wheel topology with over 15 vertices and the power of cycle graphs with more than 20 vertices and $k = 3$. For graphs with unknown optimal value, it solved fewer Cartesian product and Möbius ladder graphs. Notoriously, there are a few graphs that were not solved by model M_3, with the default search strategy, but that were solved by the B&B algorithm, such as *p5p4*, *p3k5* and *p4k4*. However, when using a more suitable search strategy, model M_4 was also able to solve these graphs, often with a smaller execution time than the B&B algorithm.

Table 7.8 presents a summary of the largest instances per topology solved by the B&B algorithm, model M_3 and the version of the model with the best search strategy.

Model M_4 added solved six more graphs than model M_3. These graphs are Cartesian products

Table 7.6: Performance comparison of model M_3, M_4 and B&B.

					B&B		M_3		M_4	
Graph	$\|V\|$	$\|E\|$	den	Op^*	*Best*	*T*	*Best*	*T*	*Best*	*T*
path10	10	9	0.20	9	**9**	0.01	**9**	2.56	**9**	0.78
path15	15	14	0.13	14	**14**	0.01	**14**	0.27	**14**	0.42
path20	20	19	0.10	19	**19**	0.01	**19**	0.25	**19**	0.44
path25	25	24	0.08	24	**24**	0.01	**24**	0.25	**24**	0.42
path30	30	29	0.07	29	**29**	0.01	**29**	0.25	**29**	0.44
path40	40	39	0.05	39	**39**	0.01	**39**	0.27	**39**	0.44
cycle10	10	10	0.22	10	**10**	0.01	**10**	0.34	**10**	0.58
cycle15	15	15	0.14	15	**15**	0.01	**15**	0.26	**15**	0.4
cycle20	20	20	0.11	20	**20**	0.01	**20**	0.31	**20**	0.42
cycle25	25	25	0.08	25	**25**	0.01	**25**	0.32	**25**	0.4
cycle30	30	30	0.07	30	**30**	0.01	**30**	0.25	**30**	0.4
cycle40	40	40	0.05	40	**40**	0.01	**40**	0.26	**40**	0.45
wheel10	10	18	0.40	35	**35**	0.09	**35**	0.33	**35**	0.75
wheel15	15	28	0.27	71	**71**	915.25	**71**	0.23	**71**	0.22
wheel20	20	38	0.20	120	-	1hr	**120**	0.37	**120**	0.23
wheel25	25	48	0.16	181	-	1hr	**181**	0.28	**181**	0.23
wheel30	30	58	0.13	255	-	1hr	**255**	0.28	**255**	0.24
wheel40	40	78	0.10	440	-	1hr	**440**	0.25	**440**	0.3
bip5-5	10	25	0.56	65	**65**	2.17	**65**	0.85	**65**	0.42
cycleP10-2	10	20	0.44	30	**30**	0	**30**	0.49	**30**	0.3
cycleP10-3	10	30	0.67	60	60	0.28	**60**	0.33	**60**	1.04
cycleP15-2	15	30	0.29	45	**45**	0.11	**45**	0.39	**45**	0.28
cycleP15-3	15	45	0.43	90	90	27.91	**90**	0.3	**90**	0.63
cycleP20-2	20	40	0.21	60	**60**	1.8	**60**	0.35	**60**	0.25
cycleP20-3	20	60	0.32	120	120	1012.22	**120**	0.34	**120**	0.58
cycleP25-2	25	50	0.17	75	**75**	27.63	**75**	0.34	**75**	0.25
cycleP25-3	25	75	0.25	150	150	1hr	**150**	0.29	**150**	0.57
cycleP30-2	30	60	0.14	90	**90**	423.72	**90**	0.4	**90**	0.26
cycleP30-3	30	90	0.21	180	180	1hr	**180**	0.48	**180**	0.58
cycleP40-2	40	80	0.10	120	1hr	1hr	**120**	0.44	**120**	0.25
cycleP40-3	40	120	0.15	240	240	1hr	**240**	0.53	**240**	0.6

of two cycles $C \times C$, two paths $P \times P$ and a path with a complete graph $P \times K$, as well as the Möbius ladder graph of order 24. The initial objective of the exact algorithm approach aspect in this research work was to solve graphs of order $n \leq 40$. While the best performing of the models is able to find optimal solutions within the time budget for graphs of that order, this is limited to specific topologies. These topologies include paths, cycles, wheels and power of cycles. The largest solved Cartesian products had 20 vertices, only half of the 40 that were set as objective, and the Möbius ladder, triangulated triangles, random and HB solved graphs have even smaller order. Because of

Table 7.7: Performance comparison of model M_3, M7 and B&B (unknown opts).

						B&B		M_3		M_4	
Graph	$\|V\|$	$\|E\|$	den	lb	ub	$Best$	T	$Best$	T	$Best$	T
c3c3	9	18	0.50	19	39	**36**	0.09	**36**	0.57	**36**	0.52
c4c3	12	24	0.36	25	52	**52**	9.33	**52**	12.72	**52**	4.22
c5c3	15	30	0.29	31	65	**65**	257.13	**65**	2532.77	**65**	106.08
c6c3	18	36	0.24	37	78	-	1hr	-	1hr	**78**	1854.78
p3p3	9	12	0.33	13	24	**19**	0	**19**	0.38	**19**	0.62
p4p3	12	17	0.26	18	36	**29**	0.13	**29**	0.69	**29**	0.65
p4p4	16	24	0.20	25	60	**44**	17.73	**44**	52.38	**44**	14.16
p5p3	15	22	0.21	23	48	**42**	9.88	**42**	28.15	**42**	9.87
p5p4	20	31	0.16	32	80	**63**	3133.41	63	1hr	**63**	2907.63
p6p3	18	27	0.18	28	60	**55**	426.58	**55**	981.07	**55**	478.88
p6c3	18	33	0.22	34	123	**69**	504.16	**69**	1326.27	**69**	808.89
p3c3	9	15	0.42	16	33	**27**	0.03	**27**	0.42	**27**	0.82
p3c4	12	20	0.30	21	44	**40**	1.67	**40**	2.65	**40**	0.96
p3c5	15	25	0.24	26	55	**55**	287.31	**55**	487.35	**55**	65.19
p3c6	18	30	0.20	31	66	-	1hr	66	1hr	**66**	2228.7
p4c3	12	21	0.32	22	57	**43**	2.08	**43**	2.61	**43**	1.55
p4c4	16	28	0.23	29	76	**64**	881.27	**64**	1494.423	**64**	504.98
p5c3	15	27	0.26	28	87	**56**	35.63	**56**	76.518	**56**	39.07
c3k4	12	30	0.45	31	88	**72**	29.69	**72**	25.296	**72**	11.28
p3k4	12	26	0.39	27	80	**58**	6.19	**58**	9.138	**58**	4.83
p3k5	15	40	0.38	41	145	**104**	2823.67	104	1hr	**104**	2077.73
p4k4	16	36	0.30	37	140	**88**	2671.72	88	1hr	**88**	2259.57
mobLad12	12	18	0.27	19	58	**30**	0.63	**30**	0.627	**30**	0.56
mobLad14	14	21	0.23	22	79	**35**	5.55	**35**	2.544	**35**	1.11
mobLad16	16	24	0.20	25	102	**40**	62.11	**40**	17.9	**40**	3.67
mobLad20	20	30	0.16	31	157	-	1hr	**50**	1132.18	**50**	63.14
mobLad24	24	36	0.13	37	225	-	1hr	76	1hr	**60**	1052.18
triTriangle10	10	18	0.40	19	50	**32**	0.03	**32**	0.32	**32**	0.31
triTriangle15	15	30	0.29	31	120	**62**	41.61	**62**	122.06	**62**	38.81
rand10-5	10	24	0.53	25	66	**51**	0.55	**51**	0.94	**51**	1.56
rand10-7	10	32	0.71	33	88	**77**	1.39	**77**	1.21	**77**	1.63
rand10-9	10	41	0.91	42	113	**106**	1.7	**106**	2.04	**106**	3.02
jgl009	9	32	0.89	33	80	**75**	0.17	**75**	0.49	**75**	0.9

this reason, the objective of solving the CBSP in an exact way for graphs of order $n \leq 40$ can be considered as partially achieved. As for why some of the topologies seem to be comparatively much harder for the CP models, this is probably related to how the edges are distributed. If the vertices have a larger number of edges, whenever a label is modified that could affect the cyclic distance of a larger number of edges. Therefore, in the constraint propagation phase there would be more variables involved in common constraints, potentially causing this phase to consume more time. The

Table 7.8: Largest solved instances per model, $*$ indicates the model solved the largest instance

	Largest graph ($*$)			B&B			M_3			M_4		
Topology	G	$\|V\|$	$\|E\|$	G	$\|V\|$	$\|E\|$	G	$\|V\|$	$\|E\|$	G	$\|V\|$	$\|E\|$
path	path40	40	39	$*$			$*$			$*$		
cycle	cycle40	40	40	$*$			$*$			$*$		
wheel	wheel40	40	78	wheel15	15	28	$*$			$*$		
bip. complete	bip20-20	40	400	bip5-5	10	25	bip5-5			bip5-5		
power of cycle	cycleP40-19	40	760	cycleP30-2	30	60	$*$			$*$		
C. product $C \times C$	c8c5	40	80	c5c3	15	30	c5c3			c6c3	18	36
C. product $P \times P$	p8p5	40	67	p5p4	20	31	p6p3	18	27	p5p4		
C. product $P \times C$	p8c5	40	75	p6c3	18	33	p6c3			p6c3		
C. product $C \times K$	c8k5	40	120	c3k4	12	30	c3k4			c3k4		
C. product $P \times K$	p8k3	40	115	p4k4	16	32	p3k4	12	26	p4k4		
Möbius ladder	mobLad40	40	60	mobLad16	16	24	mobLad20	20	30	mobLad24	24	36
tri triangle	triTriangle40	36	84	triT15	15	30	triT15			triT15		
random	rand40-9	40	705	rand10-9	10	41	rand10-9			rand10-9		
HB	bcspwr01	39	42	jgl009	9	32	jgl009			jgl009		

refinements for discarding solutions based on estimations of the upper and lower bounds of the CBS were proven helpful, reducing the execution time and allowing to solve some larger graphs. However, given the nature of the problem, it is likely that when there are few assigned labels, the cost of the solutions do not break any constraint. Furthermore, as previously discussed while introducing the design of the B&B algorithm, the potential cost for a partially defined solution can have a wide range. In order to improve the results of the CP models, it could be helpful to find a common characteristic for optimal solutions, perhaps based on individuals values of cyclic distances or their distributions.

7.5 Conjectures on optimal values for the Möbius ladder and triangulated triangle topologies

Several of the instances involved in this work belong to topologies for which there is not previous reports in the CBSP literature for formulas to estimate the value of the optimum. The optimal values reported by the CP method for instances of these type can be employed as a reference to at least conjecture the optimal value in function of the topology. This section presents topology specific conjectures about the CBS of the Möbius ladder graph.

Möbius ladder: The optimal value for all the solved Möbius ladder graphs was equal to the sum of the number of vertices and edges $n+e$. Figure 7.1(a) exemplifies this for the Möbius ladder graph of order twelve. Since the number of edges in the Möbius ladder graph is $e = n + n/2$, the conjecture is that the CBS of the Möbius ladder graph is likely to be the following.

$$\text{CBS}(L_n, \varphi^*) = 2n + \frac{n}{2} \tag{7.13}$$

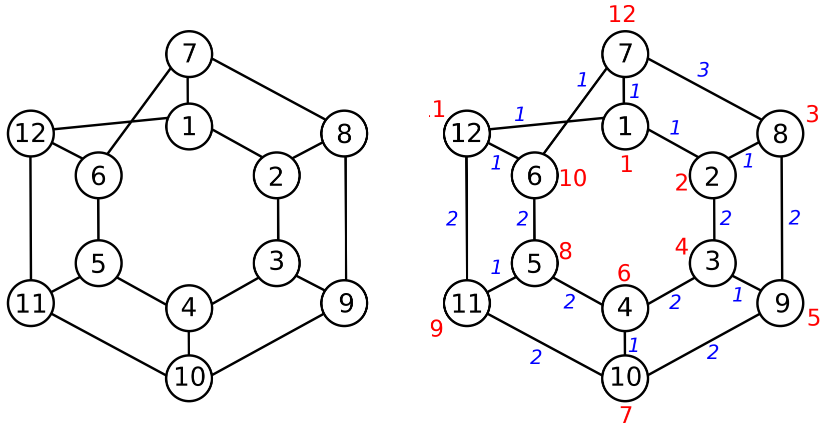
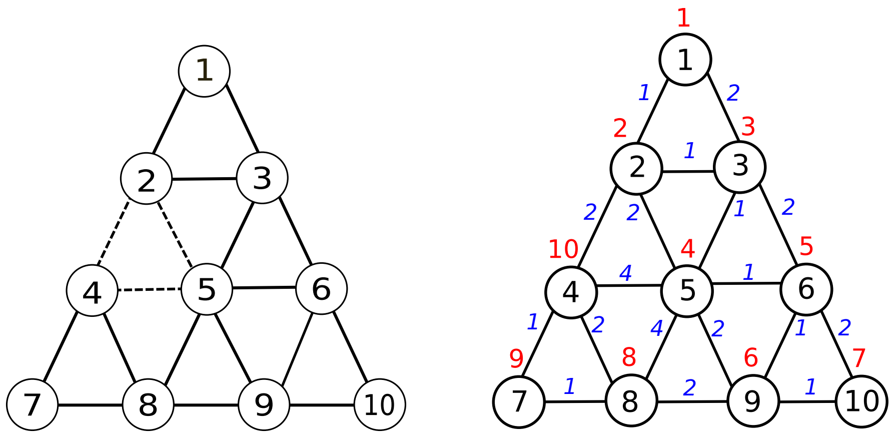

(a) Optimal labeling for L_{12}. (b) Optimal labeling for L_{12}.

Figure 7.1: Möbius ladder graph L_{10} and its optimal labeling. Labels and cyclic distances are shown in red and blue, respectively.

Triangulated triangles: Figure 7.2 presents an example of the triangulated triangle topology and its optimal solution. Every vertex in this type of graph is part of at least one cycle of length three along with two other vertices. The number of these groups of three unique vertices is referred here as $triangles$. Visually, the vertices can be grouped into horizontal levels connected by a row of triangles, with the graph T_{10} in the example having four levels. Each level contains one vertex more than the previous one and the row of triangles connecting them has two more triangles than the previous row, except for the first row.

Supposing the vertices in each individual triangle were to be embedded in such a way that the cyclic distance for its edges was maximal, these three cyclic distances could at most sum n. If each


(a) Graph T_{10}.

(b) Optimal labeling for T_{10}.

Figure 7.2: Triangulated triangle graph T_{10} and its optimal labeling. Labels and cyclic distances are shown in red and blue, respectively. The edges marked in dashed lines exemplify a triangle. In total, the graph has nine triangles and four levels.

triangle contributed with the worst cost of n to the CBS, then

$$\text{CBS}(T_n, \varphi^*) < n(triangles)\,, \tag{7.14}$$

where $triangles$ is the number of triangles within the graph.

This calculation does not take into account that many of the triangles in the T_n graph have edges in common, so while it holds, it is likely to be an overestimation that can be improved.

7.6 Conclusions

This chapter introduced the use of constraint programming as an exact method approach to solve the CBSP for relatively small graphs, of order $n \leq 40$. An initial CP model was proposed, consisting of constraints that ensure the feasibility of the solutions and the minimization of the CBS cost. In incrementally refined versions of the model new constraints were added. These constraints eliminated isomorphic solutions under mirrors and rotations (model M_1), included lower and upper bounds for the cost of the optimum based on the graph topology (model M_2) and the relationship between the CBSP and the BSP (model M_3). The models were tested over a set of topologically diverse graphs of order $n \leq 40$, founding model M_3 as the best performing one in terms of solved number of instances, their order and the execution time required.

Further tests with model M_3 included the use of search strategies, which help the solver in determining search priorities in the assignation of variables and the exploration of their domains. The default search strategy, employed in the previous tests, was to assign first the variable with the smallest domain size (*first-fail*) and explore its domain in ascending order (*indomain-min*). This default search strategy performed above average, ranking in third place in number of solved instances. However, the best performing search strategy was to prioritize the assignation of the variable with the smallest domain value (smallest), as before, exploring its domain in ascending order (*indomain-min*). This allowed model M_3 to solve six more instances than when the default search strategy was employed. The experiments with search strategies also showed that the random exploration of domains was unsuitable for the proposed model, since in all cases caused a worst performance than its ascending order counterpart, which was similar to that of the initial model M_0 before the incremental refinements.

In order to provide a comparison of the proposed CP approach with at least other exact method, an ad hoc B&B algorithm were proposed. This algorithm prioritizes the order of partial solutions and it employs an estimation of their potential CBS, based on the cyclic distance of assigned edges and the unassigned labels, to guide the exploration of the branches. While the results showed that the B&B algorithm was competitive with the initial CP model, the refined model and the use of search strategies were proven more reliable, solving a higher number of larger instances. Furthermore, when model M_3 employs the best search strategy its execution time is also smaller.

The objective of solving graphs of order $n \leq 40$ is considered partially archived, since it was not possible for all the included topologies. Particularly, the results for the complete bipartite, Cartesian products, Möbius ladder, triangulated triangles, random and Harwell-Boeing graphs variate in terms of the maximum order solved, reaching $n = 20$ in the best case. Further study of the topological influences is required, but density, regularity and the scope of the upper and lower bounds seem to play relevant roles. Finally, based on the observation of the optimal values produced, two conjectures for the optimal value of the Möbius ladder graph topology was proposed. Future research on the proposed CP model could explore the inclusion of constraints on the values of individual cyclic distances and studying the use of warm starts, which is a mechanism to provide the model with an initial known solution.

The next chapter closes this work by presenting a summary of the main findings and contributions that were produced, as well as a general outline of some possible way forward for future work and related research paths.

8 Conclusions

The research reported in this work focused in the study of the cyclic bandwidth sum problem from different complementary perspectives. These included the nature of the problem influenced by its objective function, instance features, factors contributing to the difficulty of its fitness landscape and the proposal of diverse solving approaches. This chapter closes the work by presenting a summary of the resulting conclusions and contributions, as well as a general outline for future work in the study of the CBSP.

8.1 Reducing neutrality through the fitness function

In Chapter 3, a study of three alternative fitness functions for the CBSP was presented. This had the objective of reducing the neutrality of the problem in order to help search algorithms to perform better. The design of the candidate alternative functions was based on the assignation of weights to cyclic distances of cost k and different definitions of which cyclic distances may contribute more to the total solution cost. The methodology developed by Garza-Fabre et al. [38] was employed to evaluate the candidate functions by considering three main aspects: improved discrimination capability, consistency with the problem objective, and demonstrable success guiding basic local search based algorithms to better results. It was found that the alternative fitness function f_3 met all these criteria, reducing the number of solutions that have equal cost, while maintaining consistency with the objective of the problem, and helping the steepest descent and iterated local search algorithms to significantly improve their performance.

Examining the occurrences of search cycles within the ILS algorithm under each of the evaluated functions showed that the they are linked to worse performance and more often present if the fitness function has poor discrimination ability. However, the absence of search cycles is not a guarantee of better results without being also paired with CBS-compatibility.

8.2 Metaheuristic and hyperheuristic solution approaches

The focus of Chapter 4 was the use of memetic algorithms to solve the CBSP, considering different configurations of genetic operators for selection, crossover, mutation and survival strategy. The main features of the proposed memetic algorithm (MA) framework were one offspring crossover, an intermittent approach to the local search and the use of the insertion operator as a secondary mutation. This study also considered the use of the alternative evaluation function f_3.

It was found that the selection pressure balance, managed by the selection and survival strategies, is a crucial factor for the performance of the MA on the CBSP, with the combination of binary-tournament selection and (μ, λ) survival strategy being the most successful one. The cyclic crossover, insertion and reduced 3-swap mutation operators were also present in the best performing MA, but the results suggest that they have a less determinant influence over performance in comparison to the other operators. Another relevant result is that the effectiveness of the operators can vary on different phases of the search and this could be taken advantage of by varying the operator configuration during execution. It was observed that fist-improvement strategy in the local search phase was likely to limit the helpfulness of function f_3 by accepting very small fitness improvements immediately when they are available.

The best performing MA was MA-20, which achieved significantly improved results when compared to the algorithms in the CBSP literature it was compared to, though it still exhibited some susceptibility to variations on instance topologies.

Chapter 5 presented the implementation of a dynamic multi-armed bandit as a hyperheuristic approach in order to automatize the selection of operators within the memetic algorithm, adapting between operator configurations based on a reward strategy that considered past good performance and usage frequency. The resulting hyperheuristic algorithm, DMAB+MA, significantly improved the results previously obtained by MA-20, producing new best-known results and reaching the objective of consistently produce the best-known results with more independence from the guest graph topology. Beyond the performance analysis of DMAB+MA, a study of how the operator and their combinations through the execution affect the fitness showed that operators of the same type can be very successful in complementing each other in different moments of the search. This was particularly observed in the analysis of mutation, selection and survival. Finally, in experiments removing the alternative evaluation scheme f_3 from the DMAB+MA a worsening of performance was observed, further demonstrating the importance of increasing discrimination in the fitness function to reduce neutrality and difficulty.

8.3 Fitness landscape and instance difficulty

In order to further understand the challenges of the problem for particular algorithms and the impact of the proposed alternative evaluation scheme, Chapter 6 analyzed the interplay of instance features, fitness landscape characteristics and algorithm's performance. To the best of the author knowledge, the first fitness landscape analysis of the CBSP problem was conducted. This analysis found that the alternative fitness function significantly reduces neutrality and the number of local optima while maintaining most of the studied structural features of the fitness landscape. The study comprehending the fitness landscape features with graph metrics and algorithms performance found relationships between the order of the graph, the autocorrelation coefficient, descent distance and difficulty for the DMAB+MA, likely associating the increase in graph order with larger, harder to escape attraction basins. High neutrality, the graph's eccentricity and its endpoint percentage were identified as factors affecting difficulty, specially for MA-20. DMAB+MA design, in particular the alternative fitness function f_3 allowed it to be less affected by these factors.

The inner mechanics of DMAB+MA and MA-20 were further studied by the analysis of their search trajectory networks, producing a visual representation and metrics for the search space exploration and its underlying structure. This provided additional insight on how they operate as well as the structure of the search space varying across graph topologies, and it demonstrated DMAB+MA ability to reach better areas of the landscape and to avoid regions where MA-20 can get trapped.

8.4 Exact solution approaches

Chapter 7 dealt with solving the CBSP in an exact way for relatively small graphs, of order $n \leq 40$. A constraint programming model for the CBSP was proposed and incrementally refined, improving its performance in terms of solved graphs and execution time. This was achieved by removing isomorphic solutions, using knowledge about lower and upper bounds under specific graph topologies, and cost relationships between the CBSP and BSP. Customized ways to conduct the search, defined by search strategies, were also explored, finding that the best performance was achieved by the prioritized assignation of variables with the smallest domain value and the exploration of domains in ascending order. Since, to the best of the author's knowledge, there are no other exact methods reported in the CBSP literature, an ad hoc B&B algorithm for the CBSP was developed in order to provide a comparison against the proposed CP models. The final refined model using the best search strategy outperformed the B&B algorithm and all the other version of the CP model. The results achieved by the proposed CP model allowed to draw conjectures on the value of the optimum for the Mobius ladder and triangulate triangles topologies, for which there are no topology specific upper or lower bounds reported. The objective of solving the CBSP in an exact way for instances of order $n \leq 40$ was partially achieved, since this was not possible for all the considered graph topologies.

8.5 Contributions summary

The following are the main contributions achieved by this research:

- A CBS-compatible alternative fitness function, with increased discrimination ability, that was able to significantly reduce the neutrality of the problem and the number of local optima, while maintaining the rest of the fitness landscape structure. The function was rigorously tested on several algorithms, finding it can significantly improve their performance.

- Several solution approaches that explored a variety of techniques and operators, and outperformed the ones previously reported in the literature, producing new optimal and best-found solutions. These included:

  - The steepest descent and iterated local search, used for the assessment of alternative fitness functions and sampling for local optima related measures in the fitness landscape analysis.

  - MA-20, a memetic algorithm that significantly improved the results from previously reported methods.

  - DMAB+MA, a hyperheuristic algorithm able to automatize the operator and the fitness function selection to perform consistently over different instance topologies. DMAB+MA further improved the results obtained by MA-20, achieving and updating the best-known solution costs. It was, to the best of this author knowledge, the first DMAB implementation that adapts several types of genetic operators and the fitness function simultaneously.

  - The first exact approaches to solve this problem, particularly the final CP model and the B&B algorithm, which both incorporated in their design relevant information about the problem characteristics.
  - The performance assessment of MACH and GVNS on numerous graphs beyond the ones reported originally in the literature can be considered a by-product contribution.

- Optimal and improved best-known solutions for a wide variety of graph topologies and increased instance sizes trated compared to the literature.

- A thoroughly study of genetic operators and their interaction with the alternative evaluation function in memetic algorithms and their automatized selection within the dynamic multi-armed bandit hyperheuristic.

- The first fitness landscape analysis of the CBSP, focused in the impact of the alternative evaluation scheme over neutrality, local and global optima, autocorrelation and negative slope coefficient.

- A study of the inner mechanics of DMAB+MA and MA-20 based on search trajectory networks, that allow to visualize, measure and compare how they conduct the search through specific areas of the search space and provides a new complementary perspective of the fitness landscape of the problem.

  - As by-product of this analysis, a contribution was made to the search trajectory network literature in the form of a novel method to group solutions into locations. While it was created with combinatorial problems in mind, it can be applied to any problem for which a notion of distance between solutions can be defined.

- A comprehensive study of the characteristics of the CBSP, its difficulty and how to solve it. This study linked together key aspects of the problem such as graph features, fitness landscape metrics and the performance of algorithms.

8.6 Future work

The following are possible interest areas for further research.

- Future research on the proposed CP model could explore the inclusion of constraints on the values of individual cyclic distances and studying the use of warm starts, which are a mechanism to provide the model with an initial known solution. Using the alternative fitness function in the CP model is also a relevant topic to be considered.
- The formal mathematical proof of the conjectures on the CBS calculation for the Mobius ladder and triangulated triangle topologies.
- The analysis of graph metrics, fitness landscape features and algorithms performance reported in Chapter 6 considered only MA-20 and DMAB+MA. It would be desirable to extend it to include more solution approaches, particularly the proposed exact algorithms.
- It was speculated that the results of the fitness landscape and STN based analysis suggested that more complex structures, such as attraction basins and fitness plateaus play a relevant role on the instance difficulty. Information about these structures was inferred from metrics such as descent distance and the neutrality ratio, but it may be worth assessing them more directly.
- The methodology reported in Chapter 6 can provide a very complete assessment of the characteristic of a problem, and could be possible to apply it to other graph embedding problems. Furthermore, the combined information about instance metrics, fitness landscape

features and algorithm performance has been employed for instance difficulty classification and algorithm selection based on machine learning predictions of success [124]. Exploring this type of powerful techniques would be an interesting natural extension of this work.

A

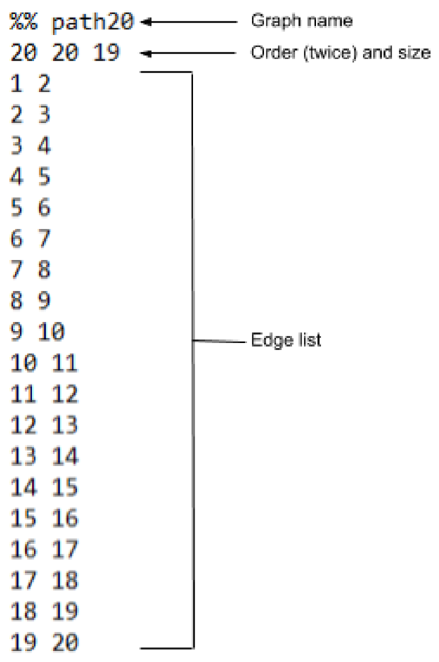
Experimental settings

This appendix describes the conditions for the experiments reported in Chapters 3, 4, 5, 6 and 7 of this work. It addresses the benchmark instances and criteria that were employed for comparing algorithms for the CBSP, such as the definition our the criterion for solution quality and the methodology for assessing statistical significance. It describes, as well, the experimental platform and the tools, libraries, languages and technologies employed for algorithm implementation and evaluation.

A.1 Instance benchmark set

This section provides details about the 642 instances involved in the diverse experiments reported along this work, their sizes, representation and graph topologies. In order to reduce the demand for computing power, the time employed in executing algorithms and processing their data, most of the experiments employed reduced instance subsets.

The instances were represented as plain text files with an standardized structure. The first line of an instance file is a comment that can include the name of the instance. The next line is the



```
%% path20
20 20 19
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 10
10 11
11 12
12 13
13 14
14 15
15 16
16 17
17 18
18 19
19 20
```

Graph name
Order (twice) and size
Edge list

Figure A.1: Graph P_{20} represented in the instance format.

number of vertices n (twice) and the number of edges e. The following e lines are space-separated pairs of vertices representing the edge set. Each edge is represented only once.

A.1.1 Standard topologies

The set of instances named standard includes 142 graphs with the following graph topologies: path, cycle, wheel, power of cycle, star, tree, complete bipartite, caterpillar, Möbius ladder, triangulated triangle and meshes. For most of these topologies there were exact formulas for the value of the optimum, shown in Section 2.3.1.2. For each topology (excluding tree, star, mesh and caterpillar) there were ten graphs with order $n = \{100, 120, 140, 160, 180, 200, 400, 600, 800, 1000\}$. The powers of cycles were generated for $k = \{2, 10\}$.

A.1.2 Cartesian products

This set includes 231 graphs resulting of the Cartesian product of paths P_n, cycles C_n and complete graphs K_n. The subsets of graphs are: $P_n \times P_m$, $P_n \times C_m$, $P_n \times K_m$, $C_n \times C_m$, $C_n \times K_m$ and $K_n \times K_m$. Subsets where the input graphs have the same topology ($P_n \times P_m$, $C_n \times C_m$ and $K_n \times K_m$) have 28 instances each, the rest of the subsets have 49 instances each.

All graphs in the Cartesian product set were generated in Python, employing the NetworkX library. The corresponding input graphs G_n and H_m had order $3 \leq n, m \leq 9$. The resulting graphs have order $9 \leq n \leq 81$ and their size is $81 \leq e \leq 405$. The upper bounds for graphs within this set were presented in Section 2.3.1.3.

A.1.3 Random graphs

The set of random graphs includes graphs created by the Erdös-Rényi method, which creates an edge between each pair of vertices according to a given probability. The probabilities were $p = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ with $n = \{10, 11, 12, 15, 20, 30, 40, 100\}$ vertices. There are a variable number of graphs for every combination of order and probability, ranging from one to ten, for 118 graphs in total.

There was also random graphs created with the Newman–Watts–Strogatz [87] and Barabási–Albert [7] models with $n = \{10, 11, 12, 100\}$ vertices. Newman–Watts–Strogatz graphs model are small-world networks. Vertices in the graph are adjacent to their k nearest neighbors with respect to a ring topology. Vertices are also adjacent to random vertex with probability p. The twelve graphs of this type were generated with $p = \{0.3, 0.5, 0.7\}$ and $k = \{2, 3, 5\}$. The Barabási–Albert method models the growth of networks by adding a new vertex and attaching it to a fixed number of other existing vertices ba. Twelve graphs were generated with $ba = \{2, 3, 5\}$.

A.1.4 Harwell-Boeing sparse matrices

The benchmark includes a set of 131 graphs derived from Harwell-Boeing sparse matrices,[1] related to real-world problems arising in several areas, from structural engineering to economics. The set contains 15 subsets from different domains, with graphs having order $9 \leq n \leq 5300$ and size $32 \leq e \leq 8271$.

A.1.5 Instances for fitness landscape analysis

Table A.1 lists the 90 instances involved in the fitness landscape analysis from Chapter 6.

A.2 Relative Root Mean Square Error

The relative root mean square error (RMSE) is a metric utilized for comparing algorithm's performance in terms of solution quality. For each instance being solved by a particular algorithm, the RMSE is computed for R independent executions. The RMSE metric for a certain test instance t is defined as:

$$\text{RMSE}(t) = 100\%\sqrt{\frac{\sum_{r=1}^{R}\left(\frac{\text{CBS}_r(t)-\text{CBS}^*(t)}{\text{CBS}^*(t)}\right)^2}{R}}, \tag{A.1}$$

where $\text{CBS}_r(t)$ is the best solution quality achieved by the algorithm at execution r, R is the total number of executions, and $\text{CBS}^*(t)$ is the best-know quality solution for instance t. An RMSE equal to 0% means the algorithm achieved the best-known solution quality in all R executions, and therefore it is the preferred value.

For comparing the algorithm performance among the complete instance set $\mathcal{T}$, the overall root mean square error (O-RMSE) is computed as the average RMSE value for the number of instances

[1]Harwell-Boeing sparce matrices collection can be found in `https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/`

Table A.1: Graphs employed in the fitness landscape analysis.

G	$\|V\|$	$\|E\|$	G	$\|V\|$	$\|E\|$	G	$\|V\|$	$\|E\|$
path10	10	9	c9k9	81	405	nos6	675	1290
path100	100	99	k9k9	81	648	will57	57	127
path11	11	10	p3c4	12	20	randba10_2	10	16
path12	12	11	p3k4	12	26	randba100_3	100	291
path200	200	199	p3p3	9	12	randba100_3	100	291
cycle10	10	10	p4p3	12	17	randba100_3	100	291
cycle100	100	100	p9c9	81	153	randba100_5	100	475
cycle11	11	11	p9k9	81	396	randba11_2	11	18
cycle12	12	12	p9p9	81	144	randba12_2	12	20
cycle200	200	200	ash85	85	219	randnws_3_100_3	100	132
cyclePow10-2	10	20	494_bus	494	586	randnws_3_100_5	100	264
cyclePow100-10	100	1000	662_bus	662	906	randnws_5_10_2	10	15
cyclePow100-2	100	200	685_bus	685	1282	randnws_5_100_3	100	147
cyclePow11-2	11	22	bcspwr01	39	46	randnws_5_100_5	100	297
cyclePow12-2	12	24	bcspwr02	49	59	randnws_5_11_2	11	20
cyclePow200-10	200	2000	bcspwr03	118	179	randnws_5_12_2	12	16
cyclePow200-2	200	400	bcsstk01	48	176	randnws_7_10_2	10	19
wheel10	10	18	bcsstk06	420	3720	randnws_7_100_3	100	170
wheel100	100	198	can_24	24	68	randnws_7_100_5	100	334
wheel11	11	20	can44	144	576	randnws_7_11_2	11	20
wheel12	12	22	can_292	292	1124	randnws_7_12_2	12	19
wheel200	200	398	can_445	445	1682	rand3_100	100	1422
caterpillar3	9	8	can_715	715	2975	rand5_10	10	20
mobiusLadder10	10	15	curtis54	54	124	rand5_100	100	2399
mobiusLadder12	12	18	dwt_503	503	2762	rand5_11	11	25
triTriangle10	10	18	dwt_592	592	2256	rand5_12	12	37
triTriangle6	6	9	ibm32	32	90	rand7_10	10	27
c3k4	12	30	impcol_b	59	281	rand7_100	100	3426
c4k3	12	24	impcol_d	425	1267	rand7_11	11	43
c9c9	81	162	nos4	100	247	rand7_12	12	43

$|\mathcal{T}|$.

$$\text{O-RMSE} = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \text{RMSE}(t) \ . \tag{A.2}$$

A.3 Statistical significance

The following methodology is employed to asses the statistical significance of algorithm's results. Normality of data distributions was evaluated by the *Shapiro-Wilk* test. *Bartlett's* test was implemented to determine whether the variances of the normally distributed data were homogeneous or not. *ANOVA* test was used in the cases where variance homogeneity was present and *Welch's t* parametric tests on the contrary. Meanwhile, *Kruskal-Wallis* test was implemented for non-normal data. In all cases the significance level considered was 0.05.

A.4 Experimental platform and coding tools

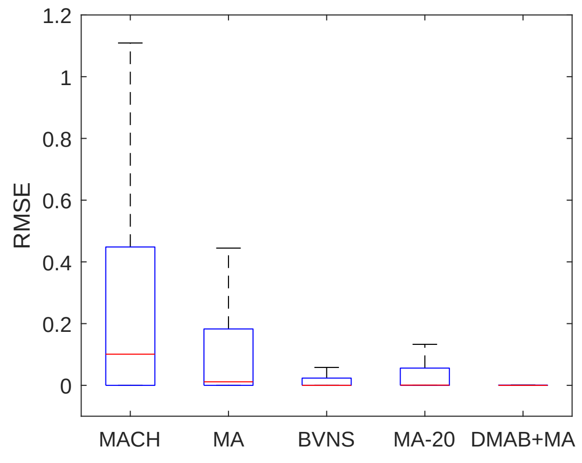
Python and the NetwokX [45] library were employed for instance generation and validation. Most of the algorithms included in this work were implemented in C/C++ and compiled with gcc/g++ using the optimization flag *-03*. MiniZinc [86] was used for coding and solving constraint programming and mixed-integer linear programming models of the CBSP. The parameters of algorithms were tuned employing R and the irace package [72] for automatized tuning. Statistical significance assessment, results processing and generation of charts and plots were ran in Matlab and R. Experiments were executed sequentially on the Neptuno cluster at Cinvestav Tamaulipas, which has 104 Intel(R) Xeon X5550 2.6 Hz processor nodes and 16 GB in RAM.

B

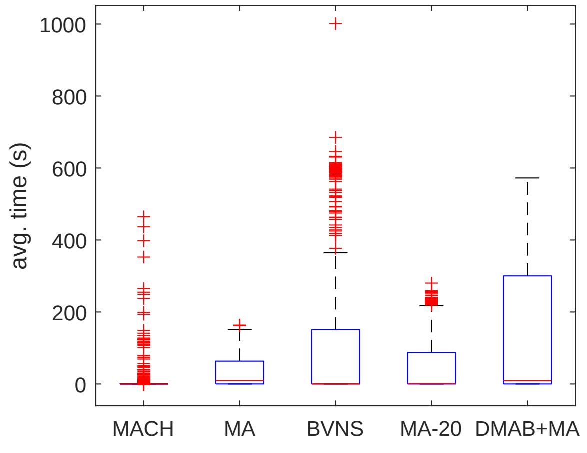
Detailed results

This appendix presents a comprehensive benchmark for MA-20 and DMAB+MA, as well as algorithms from the CBSP literature, including MACH [47], MA [85] and BVNS [85]. The BVNS and MA algorithms were executed with 50,000,000 iterations, while the MA was afforded 10,000,000 generations, employing the parameters reported in [85]. Figure B.1 shows the performance of the algorithms in terms of solution quality (RMSE) and the average time (in seconds) they required to produce the results. Figure B.2 summarizes the results of the pair-wise statistical significance analysis comparing the solution quality of MA-20 and DMAB+MA with that of the previously mentioned algorithms from the literature. DMAB+MA was the algorithm that most consistently achieved the best results in terms of solution quality, producing the smallest RMSE values and the largest amount of wins in the statistical significance comparison.

The following tables present detailed results for instances grouped into in four subsets: Cartesian products, Standard graphs, Harwell-Boeing graphs and Random graphs (see Appendix A.1). The results list the name, order and size of every graph, followed by the best cost ($Best$) produced by



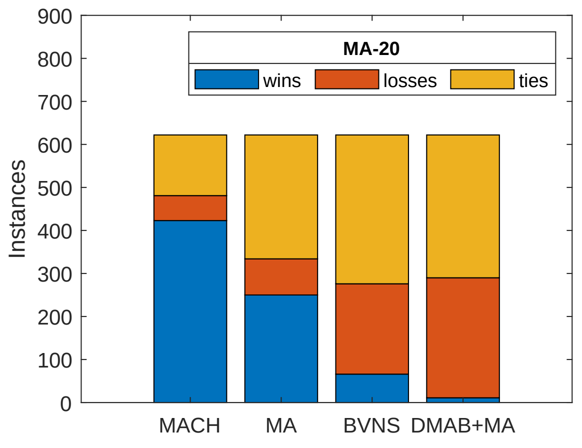
(a) RMSE values.



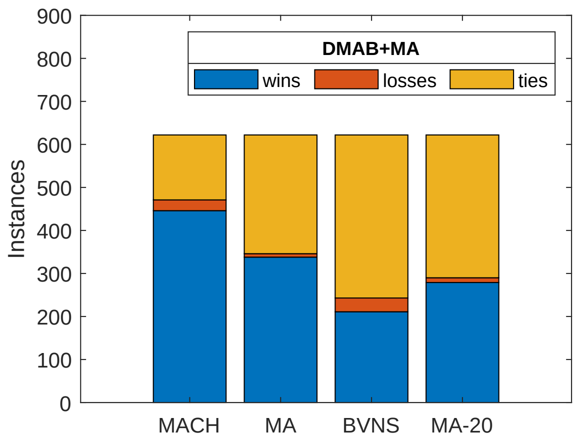
(b) Average execution time.

Figure B.1: RMSE values and average time to the best-found solution for MA-20, DMAB+MA and algorithms from the literature.

each algorithm, the average cost (Avg), its standard deviation (Std) and average time for producing the best cost solution. Blank cells in the columns corresponding to MACH indicate cases where this constructive heuristic did not report any solution after running for four hours.



(a) MA-20



(b) DMAB+MA

Figure B.2: Statistical significance summary for MA-20 and DMAB+MA compared to algorithms from the literature, in terms of wins, losses and ties. Wins favor either MA-20 in Figure B.2(a), or DMAB+MA in Figure B.2(b).

Table B.1: Detailed results of MACH, MA, BVNS, MA-20, and DMAB+MA for Cartesian products (part one)

			MACH				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
c3c3	9	18	36.00	37.08	0.75	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01
c4c3	12	24	52.00	52.00	0.00	0.01	52	52.00	0.00	0.01	52	52.00	0.00	0.01	52	52.00	0.00	0.01	52	52.00	0.00	0.01
c4c4	16	32	88.00	91.88	4.70	0.01	88	88.00	0.00	0.01	88	88.00	0.00	0.01	88	88.00	0.00	0.01	88	88.00	0.00	0.02
c5c3	15	30	65.00	65.00	0.00	0.01	65	65.00	0.00	0.01	65	65.00	0.00	0.01	65	65.00	0.00	0.01	65	65.00	0.00	0.01
c5c4	20	40	110.00	130.36	9.44	0.01	110	110.00	0.00	0.04	110	110.00	0.00	0.01	110	110.00	0.00	0.03	110	110.00	0.00	0.02
c5c5	25	50	165.00	194.78	10.05	0.01	165	166.10	2.64	6.68	165	165.00	0.00	0.01	165	165.72	2.47	2.49	165	165.00	0.00	0.41
c6c3	18	36	78.00	78.00	0.00	0.01	78	78.00	0.00	0.01	78	78.00	0.00	0.01	78	78.00	0.00	0.01	78	78.00	0.00	0.01
c6c4	24	48	132.00	159.88	16.46	0.01	132	132.00	0.00	0.01	132	132.00	0.00	0.01	132	132.00	0.00	0.01	132	132.00	0.00	0.03
c6c5	30	60	216.00	251.20	21.14	0.01	198	198.00	0.00	1.93	198	198.00	0.00	0.01	198	198.64	4.53	0.15	198	198.00	0.00	1.04
c6c6	36	72	276.00	340.00	26.07	0.01	276	277.12	5.54	8.08	276	276.00	0.00	0.02	276	292.80	13.86	0.38	276	276.00	0.00	5.96
c7c3	21	42	91.00	91.00	0.00	0.01	91	91.00	0.00	0.01	91	91.00	0.00	0.01	91	91.00	0.00	0.01	91	91.00	0.00	0.02
c7c4	28	56	166.00	195.00	21.64	0.01	154	154.00	0.00	0.01	154	154.00	0.00	0.01	154	154.00	0.00	0.02	154	154.00	0.00	0.08
c7c5	35	70	231.00	307.70	37.85	0.01	231	231.00	0.00	0.80	231	231.00	0.00	0.01	231	238.40	22.43	0.01	231	231.00	0.00	6.54
c7c6	42	84	342.00	434.60	40.43	0.01	322	324.08	10.42	6.42	322	322.00	0.00	0.01	322	325.84	13.30	3.43	322	322.00	0.00	33.78
c7c7	49	98	463.00	547.10	50.88	0.01	427	443.96	24.97	18.34	427	427.00	0.00	0.03	427	460.92	25.70	0.04	427	427.00	0.00	77.91
c8c3	24	48	104.00	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.02
c8c4	32	64	176.00	222.60	33.39	0.01	176	176.00	0.00	0.01	176	176.00	0.00	0.01	176	176.00	0.00	0.01	176	176.00	0.00	0.28
c8c5	40	80	278.00	360.04	53.26	0.01	264	264.00	0.00	0.29	264	264.00	0.00	0.01	264	266.52	17.82	0.02	264	264.00	0.00	38.98
c8c6	48	96	368.00	489.60	74.87	0.01	368	368.00	0.00	3.93	368	368.00	0.00	0.01	368	370.00	14.14	0.69	368	368.00	0.00	185.07
c8c7	56	112	516.00	683.40	85.69	0.01	488	494.32	22.47	18.20	488	488.00	0.00	0.05	488	519.48	45.30	6.91	488	488.00	0.00	156.42
c8c8	64	128	686.00	856.56	87.72	0.01	624	639.36	35.55	16.08	624	624.00	0.00	0.08	624	685.44	46.55	0.19	624	624.00	0.00	194.49
c9c3	27	54	117.00	117.00	0.00	0.01	117	117.00	0.00	0.02	117	117.00	0.00	0.01	117	117.00	0.00	0.01	117	117.00	0.00	0.01
c9c4	36	72	198.00	240.60	31.62	0.01	198	198.00	0.00	0.02	198	198.00	0.00	0.01	198	198.00	0.00	0.01	198	198.00	0.00	0.47
c9c5	45	90	297.00	397.20	74.01	0.01	297	297.00	0.00	0.35	297	297.00	0.00	0.01	297	297.00	0.00	0.01	297	297.00	0.00	170.78
c9c6	54	108	414.00	576.00	98.39	0.01	414	417.36	23.76	0.63	414	414.00	0.00	0.02	414	417.36	23.76	0.67	414	414.00	0.00	134.67
c9c7	63	126	577.00	786.94	105.57	0.01	549	555.04	29.89	4.81	549	549.00	0.00	0.05	549	551.60	18.38	6.05	549	549.00	0.00	106.83
c9c8	72	144	758.00	1047.04	116.18	0.02	702	717.28	43.83	21.07	702	702.00	0.00	0.16	702	797.00	68.94	0.45	702	702.00	0.00	200.15
c9c9	81	162	945.00	1210.02	137.45	0.02	873	904.90	60.68	16.84	873	873.00	0.00	0.26	873	965.80	70.31	0.58	873	873.00	0.00	138.15
c3k3	9	18	36.00	36.98	0.68	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01
c3k4	12	30	76.00	76.68	1.11	0.01	72	72.00	0.00	0.01	72	72.00	0.00	0.01	72	72.00	0.00	0.01	72	72.00	0.00	0.01
c3k5	15	45	134.00	135.38	1.58	0.01	126	126.00	0.00	0.01	126	126.00	0.00	0.01	126	126.00	0.00	0.01	126	126.00	0.00	0.01
c3k6	18	63	207.00	217.68	3.95	0.01	201	201.00	0.00	0.01	201	201.00	0.00	0.01	201	201.00	0.00	0.01	201	201.00	0.00	0.02
c3k7	21	84	304.00	309.02	2.38	0.01	297	297.00	0.00	0.01	297	297.00	0.00	0.01	297	297.00	0.00	0.01	297	297.00	0.00	0.01
c3k8	24	108	430.00	436.00	3.38	0.01	420	420.00	0.00	0.01	420	420.00	0.00	0.01	420	420.00	0.00	0.01	420	420.00	0.00	0.01
c3k9	27	135	582.00	592.16	4.06	0.01	573	573.00	0.00	0.01	573	573.00	0.00	0.01	573	573.00	0.00	0.01	573	573.00	0.00	0.01
c4k3	12	24	52.00	52.00	0.00	0.01	52	52.00	0.00	0.01	52	52.00	0.00	0.01	52	52.00	0.00	0.01	52	52.00	0.00	0.01
c4k4	16	40	104.00	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.01
c4k5	20	60	180.00	180.00	0.00	0.01	180	180.00	0.00	0.01	180	180.00	0.00	0.01	180	180.00	0.00	0.01	180	180.00	0.00	0.02

Table B.2: Detailed results of MACH, MA, BVNS, MA-20, and DMAB+MA for Cartesian products (part two)

			MACH				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	T	*Best*	*Avg*	*Std*	T	*Best*	*Avg*	*Std*	T	*Best*	*Avg*	*Std*	T	*Best*	*Avg*	*Std*	T
c4k6	24	84	284.00	284.00	0.00	0.01	284	284.00	0.00	0.01	284	284.00	0.00	0.01	284	284.00	0.00	0.01	284	284.00	0.00	0.01
c4k7	28	112	420.00	420.00	0.00	0.01	420	420.00	0.00	0.02	420	420.00	0.00	0.01	420	420.00	0.00	0.01	420	420.00	0.00	0.04
c4k8	32	144	592.00	592.00	0.00	0.01	592	592.00	0.00	0.26	592	592.00	0.00	0.01	592	592.00	0.00	0.01	592	592.00	0.00	0.12
c4k9	36	180	804.00	804.00	0.00	0.01	804	804.00	0.00	0.21	804	804.00	0.00	0.02	804	804.00	0.00	0.01	804	804.00	0.00	0.32
c5k3	15	30	65.00	65.00	0.00	0.01	65	65.00	0.00	0.01	65	65.00	0.00	0.01	65	65.00	0.00	0.01	65	65.00	0.00	0.01
c5k4	20	50	130.00	130.00	0.00	0.01	130	130.00	0.00	0.01	130	130.00	0.00	0.01	130	130.00	0.00	0.01	130	130.00	0.00	0.02
c5k5	25	75	225.00	225.00	0.00	0.01	225	225.00	0.00	0.01	225	225.00	0.00	0.01	225	225.00	0.00	0.01	225	225.00	0.00	0.01
c5k6	30	105	355.00	355.00	0.00	0.01	355	355.00	0.00	0.02	355	355.00	0.00	0.01	355	355.00	0.00	0.01	355	355.00	0.00	0.02
c5k7	35	140	525.00	525.00	0.00	0.01	525	525.00	0.00	0.05	525	525.00	0.00	0.01	525	525.00	0.00	0.01	525	525.00	0.00	0.13
c5k8	40	180	740.00	740.00	0.00	0.01	740	740.00	0.00	0.11	740	740.00	0.00	0.02	740	740.00	0.00	0.02	740	740.00	0.00	0.39
c5k9	45	225	1005.00	1005.00	0.00	0.01	1005	1005.00	0.00	0.16	1005	1005.00	0.00	0.03	1005	1005.00	0.00	0.03	1005	1005.00	0.00	0.88
c6k3	18	36	78.00	78.00	0.00	0.01	78	78.00	0.00	0.01	78	78.00	0.00	0.01	78	78.00	0.00	0.01	78	78.00	0.00	0.01
c6k4	24	60	156.00	156.00	0.00	0.01	156	156.00	0.00	0.01	156	156.00	0.00	0.01	156	156.00	0.00	0.01	156	156.00	0.00	0.01
c6k5	30	90	270.00	270.00	0.00	0.01	270	270.00	0.00	0.06	270	270.00	0.00	0.01	270	270.00	0.00	0.01	270	270.00	0.00	0.02
c6k6	36	126	426.00	426.00	0.00	0.01	426	426.00	0.00	0.11	426	426.00	0.00	0.01	426	426.00	0.00	0.02	426	426.00	0.00	0.06
c6k7	42	168	630.00	630.00	0.00	0.01	630	630.00	0.00	0.14	630	630.00	0.00	0.01	630	630.00	0.00	0.03	630	630.00	0.00	0.31
c6k8	48	216	888.00	888.00	0.00	0.01	888	888.00	0.00	0.28	888	888.00	0.00	0.03	888	888.00	0.00	0.07	888	888.00	0.00	0.73
c6k9	54	270	1206.00	1206.00	0.00	0.02	1206	1206.00	0.00	0.22	1206	1206.00	0.00	0.06	1206	1206.00	0.00	0.14	1206	1206.00	0.00	2.19
c7k3	21	42	91.00	91.00	0.00	0.01	91	91.00	0.00	0.01	91	91.00	0.00	0.01	91	91.00	0.00	0.01	91	91.00	0.00	0.02
c7k4	28	70	182.00	182.00	0.00	0.01	182	182.00	0.00	0.01	182	182.00	0.00	0.01	182	182.00	0.00	0.01	182	182.00	0.00	0.02
c7k5	35	105	315.00	315.00	0.00	0.01	315	315.00	0.00	0.03	315	315.00	0.00	0.01	315	315.00	0.00	0.02	315	315.00	0.00	0.10
c7k6	42	147	497.00	497.00	0.00	0.01	497	497.00	0.00	0.09	497	497.00	0.00	0.01	497	497.00	0.00	0.02	497	497.00	0.00	0.26
c7k7	49	196	735.00	735.00	0.00	0.01	735	735.00	0.00	0.21	735	735.00	0.00	0.04	735	735.00	0.00	0.09	735	735.00	0.00	1.12
c7k8	56	252	1036.00	1036.00	0.00	0.02	1036	1036.00	0.00	0.24	1036	1036.00	0.00	0.08	1036	1036.00	0.00	0.24	1036	1036.00	0.00	2.56
c7k9	63	315	1407.00	1407.00	0.00	0.02	1407	1407.00	0.00	0.44	1407	1407.00	0.00	0.16	1407	1407.00	0.00	0.61	1407	1407.00	0.00	6.10
c8k3	24	48	104.00	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.02
c8k4	32	80	208.00	208.00	0.00	0.01	208	208.00	0.00	0.07	208	208.00	0.00	0.01	208	208.00	0.00	0.01	208	208.00	0.00	0.06
c8k5	40	120	360.00	360.00	0.00	0.01	360	360.00	0.00	0.08	360	360.00	0.00	0.02	360	360.00	0.00	0.02	360	360.00	0.00	0.14
c8k6	48	168	568.00	568.00	0.00	0.01	568	568.00	0.00	0.24	568	568.00	0.00	0.02	568	568.00	0.00	0.11	568	568.00	0.00	0.51
c8k7	56	224	840.00	840.00	0.00	0.01	840	840.00	0.00	0.20	840	840.00	0.00	0.07	840	840.00	0.00	0.45	840	840.00	0.00	2.33
c8k8	64	288	1184.00	1184.00	0.00	0.02	1184	1184.00	0.00	0.49	1184	1184.00	0.00	0.15	1184	1184.00	0.00	1.21	1184	1184.00	0.00	8.87
c8k9	72	360	1608.00	1608.00	0.00	0.02	1608	1608.00	0.00	0.64	1608	1608.00	0.00	0.26	1608	1608.00	0.00	1.59	1608	1608.00	0.00	9.48
c9k3	27	54	117.00	117.00	0.00	0.01	117	117.00	0.00	0.01	117	117.00	0.00	0.01	117	117.00	0.00	0.01	117	117.00	0.00	0.01
c9k4	36	90	234.00	234.00	0.00	0.01	234	234.00	0.00	0.07	234	234.00	0.00	0.01	234	234.00	0.00	0.02	234	234.00	0.00	0.09
c9k5	45	135	405.00	405.00	0.00	0.01	405	405.00	0.00	0.17	405	405.00	0.00	0.02	405	405.00	0.00	0.09	405	405.00	0.00	0.62
c9k6	54	189	639.00	639.00	0.00	0.01	639	639.00	0.00	0.28	639	639.00	0.00	0.06	639	639.00	0.00	0.28	639	639.00	0.00	1.56
c9k7	63	252	945.00	945.00	0.00	0.02	945	945.00	0.00	0.67	945	945.00	0.00	0.15	945	945.00	0.00	0.96	945	945.00	0.00	5.06
c9k8	72	324	1332.00	1332.00	0.00	0.02	1332	1332.00	0.00	0.71	1332	1332.00	0.00	0.49	1332	1332.00	0.00	2.29	1332	1332.00	0.00	11.75

Table B.3: Detailed results of Mach, MA, BVNS, MA-20, and DMAB+MA for Cartesian products (part three)

			Mach				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
c9k9	81	405	1809.00	1809.00	0.00	0.02	1809	1809.00	0.00	1.26	1809	1809.00	0.00	0.49	1809	1809.00	0.00	6.48	1809	1809.00	0.00	24.66
k3k3	9	18	36.00	37.04	0.73	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01
k4k3	12	30	76.00	76.84	1.35	0.01	72	72.00	0.00	0.01	72	72.00	0.00	0.01	72	72.00	0.00	0.01	72	72.00	0.00	0.01
k4k4	16	48	160.00	165.08	3.39	0.01	152	152.00	0.00	0.01	152	152.00	0.00	0.01	152	152.00	0.00	0.01	152	152.00	0.00	0.02
k5k3	15	45	134.00	136.00	1.37	0.01	126	126.00	0.00	0.01	126	126.00	0.00	0.01	126	126.00	0.00	0.01	126	126.00	0.00	0.01
k5k4	20	70	278.00	282.56	2.59	0.01	256	256.00	0.00	0.02	256	256.00	0.00	0.01	256	256.00	0.00	0.01	256	256.00	0.00	0.01
k5k5	25	100	488.00	504.36	7.96	0.01	460	460.00	0.00	1.39	460	460.00	0.00	0.01	460	460.00	0.00	0.88	460	460.00	0.00	0.08
k6k3	18	63	211.00	217.88	2.62	0.01	201	201.00	0.00	0.01	201	201.00	0.00	0.01	201	201.00	0.00	0.01	201	201.00	0.00	0.02
k6k4	24	96	434.00	442.92	3.71	0.01	392	392.00	0.00	0.01	392	392.00	0.00	0.01	392	392.00	0.00	0.01	392	392.00	0.00	0.01
k6k5	30	135	765.00	786.16	8.97	0.01	695	695.00	0.00	0.02	695	695.00	0.00	0.01	695	695.00	0.00	0.40	695	695.00	0.00	0.07
k6k6	36	180	1232.00	1265.36	14.93	0.01	1128	1128.00	0.00	1.56	1128	1128.00	0.00	0.01	1128	1141.68	17.65	2.20	1128	1128.00	0.00	0.76
k7k3	21	84	303.00	308.50	2.19	0.01	297	297.00	0.00	0.01	297	297.00	0.00	0.01	297	297.00	0.00	0.01	297	297.00	0.00	0.01
k7k4	28	126	636.00	648.20	5.29	0.01	568	568.00	0.00	0.01	568	568.00	0.00	0.01	568	568.00	0.00	0.01	568	568.00	0.00	0.02
k7k5	35	175	1122.00	1144.26	11.72	0.01	985	985.00	0.00	0.01	985	985.00	0.00	0.01	985	985.00	0.00	0.01	985	985.00	0.00	0.12
k7k6	42	231	1795.00	1831.08	15.71	0.02	1587	1587.00	0.00	0.14	1587	1587.00	0.00	0.02	1587	1587.00	0.00	0.83	1587	1587.00	0.00	0.12
k7k7	49	294	2668.00	2724.88	23.34	0.02	2408	2408.00	0.00	1.01	2408	2408.00	0.00	0.03	2408	2451.20	53.45	3.62	2408	2408.00	0.00	4.57
k8k3	24	108	430.00	436.64	3.67	0.01	420	420.00	0.00	0.01	420	420.00	0.00	0.01	420	420.00	0.00	0.01	420	420.00	0.00	0.01
k8k4	32	160	836.00	849.64	8.23	0.01	784	784.00	0.00	0.01	784	784.00	0.00	0.01	784	784.00	0.00	0.01	784	784.00	0.00	0.03
k8k5	40	220	1554.00	1584.32	14.80	0.02	1340	1340.00	0.00	0.01	1340	1340.00	0.00	0.01	1340	1340.00	0.00	0.01	1340	1340.00	0.00	0.41
k8k6	48	288	2478.00	2527.44	20.34	0.02	2136	2136.00	0.00	0.01	2136	2136.00	0.00	0.01	2136	2143.52	53.17	0.02	2136	2136.00	0.00	0.22
k8k7	56	364	3714.00	3764.88	23.19	0.02	3220	3220.00	0.00	0.10	3220	3220.00	0.00	0.03	3220	3225.36	37.90	16.81	3220	3220.00	0.00	3.20
k8k8	64	448	5192.00	5300.68	41.59	0.01	4640	4640.00	0.00	1.37	4640	4640.00	0.00	0.11	4640	4747.52	127.63	0.43	4640	4640.00	0.00	13.89
k9k3	27	135	585.00	592.54	3.46	0.01	573	573.00	0.00	0.01	573	573.00	0.00	0.01	573	573.00	0.00	0.01	573	573.00	0.00	0.01
k9k4	36	198	1110.00	1131.44	10.53	0.01	1048	1048.00	0.00	0.01	1048	1048.00	0.00	0.01	1048	1048.00	0.00	0.01	1048	1048.00	0.00	0.03
k9k5	45	270	1953.00	1995.68	15.05	0.02	1765	1765.00	0.00	0.02	1765	1765.00	0.00	0.01	1765	1765.00	0.00	0.01	1765	1765.00	0.00	2.42
k9k6	54	351	3305.00	3356.48	24.47	0.02	2787	2787.00	0.00	0.02	2787	2787.00	0.00	0.01	2787	2787.00	0.00	0.01	2787	2787.00	0.00	1.21
k9k7	63	441	4954.00	5002.08	29.75	0.01	4172	4172.00	0.00	0.03	4172	4172.00	0.00	0.03	4172	4172.00	0.00	0.01	4172	4172.00	0.00	45.67
k9k8	72	540	6972.00	7056.40	40.65	0.01	5984	5984.00	0.00	0.34	5984	5984.00	0.00	0.14	5984	6002.40	91.06	6.89	5984	5984.00	0.00	173.87
k9k9	81	648	9436.00	9556.00	61.09	0.02	8280	8280.00	0.00	0.81	8280	8280.00	0.00	0.34	8280	8444.48	242.20	3.13	8280	8280.00	0.00	292.58
p3c3	9	15	28.00	29.08	0.67	0.01	27	27.00	0.00	0.01	27	27.00	0.00	0.01	27	27.00	0.00	0.01	27	27.00	0.00	0.01
p3c4	12	20	44.00	45.16	1.00	0.01	40	40.00	0.00	0.01	40	40.00	0.00	0.01	40	40.00	0.00	0.01	40	40.00	0.00	0.01
p3c5	15	25	61.00	64.24	3.02	0.01	55	55.00	0.00	0.82	55	55.00	0.00	0.01	55	55.00	0.00	0.03	55	55.00	0.00	0.01
p3c6	18	30	78.00	83.76	6.06	0.01	66	66.00	0.00	0.04	66	66.00	0.00	0.01	66	66.00	0.00	0.01	66	66.00	0.00	0.01
p3c7	21	35	99.00	106.04	8.02	0.01	77	77.00	0.00	0.01	77	77.00	0.00	0.01	77	77.00	0.00	0.01	77	77.00	0.00	0.01
p3c8	24	40	118.00	129.00	11.11	0.01	88	88.00	0.00	0.01	88	88.00	0.00	0.01	88	88.00	0.00	0.01	88	88.00	0.00	0.01
p3c9	27	45	142.00	157.60	15.14	0.01	99	99.00	0.00	0.02	99	99.00	0.00	0.01	99	99.00	0.00	0.01	99	99.00	0.00	0.02
p4c3	12	21	43.00	43.00	0.00	0.01	43	43.00	0.00	0.01	43	43.00	0.00	0.01	43	43.00	0.00	0.01	43	43.00	0.00	0.01
p4c4	16	28	74.00	76.16	2.01	0.01	64	64.00	0.00	0.01	64	64.00	0.00	0.01	64	64.00	0.00	0.01	64	64.00	0.00	0.01

Table B.4: Detailed results of Mach, MA, BVNS, MA-20, and DMAB+MA for Cartesian products (part four)

			Mach				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
p4c5	20	35	105.00	108.36	3.99	0.01	91	91.24	0.96	0.23	91	91.00	0.00	0.01	91	91.00	0.00	0.01	91	91.00	0.00	0.02
p4c6	24	42	144.00	149.00	5.05	0.01	114	116.00	3.73	26.68	114	114.00	0.00	0.01	114	116.96	1.77	0.25	114	114.00	0.00	0.71
p4c7	28	49	183.00	190.68	8.07	0.01	133	137.82	10.94	13.03	133	133.00	0.00	0.01	133	133.32	2.26	2.08	133	133.00	0.00	0.18
p4c8	32	56	230.00	241.60	9.97	0.01	152	154.40	9.83	3.91	152	152.00	0.00	0.01	152	152.00	0.00	0.63	152	152.00	0.00	0.18
p4c9	36	63	279.00	296.16	12.44	0.01	171	183.44	23.96	7.16	171	171.00	0.00	0.01	171	171.00	0.00	0.32	171	171.00	0.00	0.21
p5c3	15	27	56.00	56.00	0.00	0.01	56	56.00	0.00	0.01	56	56.00	0.00	0.01	56	56.00	0.00	0.01	56	56.00	0.00	0.01
p5c4	20	36	100.00	101.12	1.00	0.01	92	92.76	0.98	21.79	92	92.00	0.00	0.01	92	93.20	0.99	0.50	92	92.00	0.00	0.63
p5c5	25	45	148.00	149.12	1.00	0.01	129	129.32	2.26	0.06	129	129.00	0.00	0.01	129	129.88	3.01	0.93	129	129.00	0.00	0.07
p5c6	30	54	198.00	200.24	2.01	0.01	166	167.66	5.48	3.01	166	166.00	0.00	0.01	166	166.16	1.13	1.01	166	166.00	0.00	0.07
p5c7	35	63	258.00	261.48	2.99	0.01	203	208.72	9.15	8.52	203	203.00	0.00	0.03	203	207.20	2.78	0.02	203	203.00	0.00	2.51
p5c8	40	72	324.00	326.52	2.99	0.01	232	244.76	15.49	3.06	232	232.00	0.00	0.02	232	240.40	9.97	0.06	232	232.00	0.00	1.55
p5c9	45	81	396.00	402.20	4.90	0.01	261	285.86	29.12	7.68	261	261.00	0.00	0.02	261	276.20	19.61	1.56	261	261.00	0.00	0.88
p6c3	18	33	69.00	69.00	0.00	0.01	69	69.00	0.00	0.01	69	69.00	0.00	0.01	69	69.00	0.00	0.01	69	69.00	0.00	0.01
p6c4	24	44	124.00	124.00	0.00	0.01	116	116.32	1.58	1.16	116	116.00	0.00	0.01	116	116.00	0.00	0.07	116	116.00	0.00	0.10
p6c5	30	55	183.00	183.00	0.00	0.01	173	173.00	0.00	0.06	173	173.00	0.00	0.01	173	173.00	0.00	0.01	173	173.00	0.00	4.42
p6c6	36	66	248.00	248.00	0.00	0.01	222	226.72	15.55	4.28	222	222.00	0.00	0.02	222	225.80	7.31	1.46	222	222.00	0.00	0.46
p6c7	42	77	325.00	325.00	0.00	0.01	279	283.94	14.75	1.42	279	279.00	0.00	0.04	279	282.66	9.48	0.53	279	279.00	0.00	0.88
p6c8	48	88	408.00	408.00	0.00	0.01	328	343.66	21.21	10.15	328	328.00	0.00	0.45	328	334.56	10.18	0.19	328	328.00	0.00	12.12
p6c9	54	99	503.00	503.00	0.00	0.01	369	398.30	26.17	5.95	369	369.00	0.00	0.85	369	386.28	16.11	0.17	369	369.00	0.00	6.65
p7c3	21	39	82.00	82.00	0.00	0.01	82	82.00	0.00	0.02	82	82.00	0.00	0.01	82	82.00	0.00	0.01	82	82.00	0.00	0.02
p7c4	28	52	146.00	146.00	0.00	0.01	138	138.44	3.11	0.06	138	138.00	0.00	0.01	138	138.00	0.00	0.01	138	138.00	0.00	0.11
p7c5	35	65	216.00	216.00	0.00	0.01	206	208.40	5.55	8.50	206	206.00	0.00	0.01	206	206.00	0.00	2.03	206	206.00	0.00	18.14
p7c6	42	78	294.00	294.00	0.00	0.01	282	286.56	13.71	22.55	282	282.00	0.00	0.05	282	283.76	2.01	0.60	282	282.00	0.00	3.49
p7c7	49	91	386.00	386.00	0.00	0.01	353	359.78	15.27	6.36	353	353.00	0.00	0.15	353	360.28	11.86	0.12	353	353.00	0.00	1.14
p7c8	56	104	486.00	486.00	0.00	0.01	424	441.02	28.23	4.28	424	424.00	0.00	2.63	424	430.56	15.19	0.50	424	424.00	0.00	1.94
p7c9	63	117	600.00	600.00	0.00	0.01	495	522.06	39.36	5.77	495	495.00	0.00	33.54	495	501.00	4.95	0.55	495	495.00	0.00	30.83
p8c3	24	45	95.00	95.00	0.00	0.01	95	95.00	0.00	0.03	95	95.00	0.00	0.01	95	95.00	0.00	0.01	95	95.00	0.00	0.01
p8c4	32	60	168.00	168.00	0.00	0.01	160	160.00	0.00	0.71	160	160.00	0.00	0.01	160	160.00	0.00	0.01	160	160.00	0.00	0.60
p8c5	40	75	249.00	249.00	0.00	0.01	239	239.66	4.67	1.97	239	239.00	0.00	0.01	239	239.00	0.00	0.31	239	239.00	0.00	79.00
p8c6	48	90	340.00	340.00	0.00	0.01	332	333.80	5.45	5.68	332	332.00	0.00	0.03	332	335.60	7.27	2.83	332	332.00	0.00	192.25
p8c7	56	105	447.00	447.00	0.00	0.01	437	439.92	13.03	22.68	437	437.00	0.00	0.85	437	440.20	11.75	0.27	437	437.00	0.00	6.62
p8c8	64	120	564.00	564.00	0.00	0.01	524	534.18	29.48	10.71	524	524.00	0.00	6.81	524	532.24	15.93	1.97	524	524.00	0.00	17.30
p8c9	72	135	697.00	697.00	0.00	0.01	623	643.08	41.52	10.40	623	623.00	0.00	18.72	623	630.60	18.29	0.84	623	623.00	0.00	187.24
p9c3	27	51	108.00	108.00	0.00	0.01	108	108.00	0.00	0.16	108	108.00	0.00	0.01	108	108.00	0.00	0.01	108	108.00	0.00	0.03
p9c4	36	68	190.00	190.00	0.00	0.01	182	182.00	0.00	0.13	182	182.00	0.00	0.01	182	182.00	0.00	0.02	182	182.00	0.00	0.96
p9c5	45	85	282.00	282.00	0.00	0.01	272	273.22	8.63	1.31	272	272.00	0.00	0.01	272	272.00	0.00	0.07	272	272.00	0.00	201.56
p9c6	54	102	386.00	386.00	0.00	0.01	378	380.78	11.15	3.58	378	378.00	0.00	0.03	378	378.00	0.00	2.79	378	378.00	0.00	231.38
p9c7	63	119	508.00	508.00	0.00	0.01	500	513.26	26.87	8.97	500	500.00	0.00	0.41	500	508.46	19.72	8.52	500	500.00	0.00	243.85

Table B.5: Detailed results of MACH, MA, BVNS, MA-20, and DMAB+MA for Cartesian products (part five)

			MACH				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
p9c8	72	136	642.00	642.00	0.00	0.01	628	644.82	55.86	19.92	628	628.00	0.00	30.21	628	637.40	22.61	7.26	628	628.00	0.00	55.80
p9c9	81	153	794.00	794.00	0.00	0.01	745	764.84	42.29	19.22	745	745.00	0.00	83.29	745	751.02	14.21	1.43	745	745.00	0.00	299.71
p3k3	9	15	28.00	28.94	0.68	0.01	27	27.00	0.00	0.01	27	27.00	0.00	0.01	27	27.00	0.00	0.01	27	27.00	0.00	0.01
p3k4	12	26	58.00	60.76	1.27	0.01	58	58.00	0.00	0.01	58	58.00	0.00	0.01	58	58.00	0.00	0.01	58	58.00	0.00	0.01
p3k5	15	40	104.00	107.56	1.40	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.01	104	104.00	0.00	0.01
p3k6	18	57	169.00	174.16	1.98	0.01	169	169.00	0.00	0.01	169	169.00	0.00	0.01	169	169.00	0.00	0.01	169	169.00	0.00	0.02
p3k7	21	77	254.00	261.12	2.99	0.01	254	254.00	0.00	0.01	254	254.00	0.00	0.01	254	254.00	0.00	0.01	254	254.00	0.00	0.01
p3k8	24	100	364.00	374.40	3.59	0.01	364	364.00	0.00	0.01	364	364.00	0.00	0.01	364	364.00	0.00	0.01	364	364.00	0.00	0.01
p3k9	27	126	504.00	513.28	3.95	0.01	502	502.00	0.00	0.01	502	502.00	0.00	0.01	502	502.00	0.00	0.01	502	502.00	0.00	0.01
p4k3	12	21	43.00	43.00	0.00	0.01	43	43.00	0.00	0.01	43	43.00	0.00	0.01	43	43.00	0.00	0.01	43	43.00	0.00	0.01
p4k4	16	36	88.00	88.00	0.00	0.01	88	88.00	0.00	0.01	88	88.00	0.00	0.01	88	88.00	0.00	0.01	88	88.00	0.00	0.01
p4k5	20	55	155.00	155.00	0.00	0.01	155	155.00	0.00	0.02	155	155.00	0.00	0.01	155	155.00	0.00	0.01	155	155.00	0.00	0.02
p4k6	24	78	248.00	248.00	0.00	0.01	248	248.00	0.00	0.05	248	248.00	0.00	0.01	248	248.00	0.00	0.01	248	248.00	0.00	0.04
p4k7	28	105	371.00	371.00	0.00	0.01	371	371.00	0.00	0.06	371	371.00	0.00	0.01	371	371.00	0.00	0.01	371	371.00	0.00	0.15
p4k8	32	136	528.00	528.00	0.00	0.01	528	528.00	0.00	0.16	528	528.00	0.00	0.01	528	528.00	0.00	0.02	528	528.00	0.00	0.33
p4k9	36	171	723.00	723.00	0.00	0.01	723	723.00	0.00	0.36	723	723.00	0.00	0.01	723	723.00	0.00	0.02	723	723.00	0.00	0.55
p5k3	15	27	56.00	56.00	0.00	0.01	56	56.00	0.00	0.01	56	56.00	0.00	0.01	56	56.00	0.00	0.01	56	56.00	0.00	0.01
p5k4	20	46	114.00	114.00	0.00	0.01	114	114.00	0.00	0.01	114	114.00	0.00	0.01	114	114.00	0.00	0.01	114	114.00	0.00	0.02
p5k5	25	70	200.00	200.00	0.00	0.01	200	200.00	0.00	0.01	200	200.00	0.00	0.01	200	200.00	0.00	0.01	200	200.00	0.00	0.03
p5k6	30	99	319.00	319.00	0.00	0.01	319	319.00	0.00	0.07	319	319.00	0.00	0.01	319	319.00	0.00	0.01	319	319.00	0.00	0.08
p5k7	35	133	476.00	476.00	0.00	0.01	476	476.00	0.00	0.19	476	476.00	0.00	0.02	476	476.00	0.00	0.02	476	476.00	0.00	0.42
p5k8	40	172	676.00	676.00	0.00	0.01	676	676.00	0.00	0.23	676	676.00	0.00	0.01	676	676.00	0.00	0.03	676	676.00	0.00	1.81
p5k9	45	216	924.00	924.00	0.00	0.01	924	924.00	0.00	0.45	924	924.00	0.00	0.03	924	924.00	0.00	0.11	924	924.00	0.00	1.80
p6k3	18	33	69.00	69.00	0.00	0.01	69	69.00	0.00	0.01	69	69.00	0.00	0.01	69	69.00	0.00	0.01	69	69.00	0.00	0.01
p6k4	24	56	140.00	140.00	0.00	0.01	140	140.00	0.00	0.03	140	140.00	0.00	0.01	140	140.00	0.00	0.01	140	140.00	0.00	0.02
p6k5	30	85	245.00	245.00	0.00	0.01	245	245.00	0.00	0.07	245	245.00	0.00	0.01	245	245.00	0.00	0.01	245	245.00	0.00	0.09
p6k6	36	120	390.00	390.00	0.00	0.01	390	390.00	0.00	0.51	390	390.00	0.00	0.01	390	390.00	0.00	0.03	390	390.00	0.00	0.45
p6k7	42	161	581.00	581.00	0.00	0.01	581	581.00	0.00	0.38	581	581.00	0.00	0.01	581	581.00	0.00	0.06	581	581.00	0.00	1.17
p6k8	48	208	824.00	824.00	0.00	0.01	824	824.00	0.00	0.31	824	824.00	0.00	0.03	824	824.00	0.00	0.20	824	824.00	0.00	2.74
p6k9	54	261	1125.00	1125.00	0.00	0.02	1125	1125.00	0.00	0.81	1125	1125.00	0.00	0.07	1125	1125.00	0.00	0.25	1125	1125.00	0.00	6.67
p7k3	21	39	82.00	82.00	0.00	0.01	82	82.00	0.00	0.01	82	82.00	0.00	0.01	82	82.00	0.00	0.01	82	82.00	0.00	0.02
p7k4	28	66	166.00	166.00	0.00	0.01	166	166.00	0.00	0.05	166	166.00	0.00	0.01	166	166.00	0.00	0.01	166	166.00	0.00	0.04
p7k5	35	100	290.00	290.00	0.00	0.01	290	290.00	0.00	0.15	290	290.00	0.00	0.01	290	290.00	0.00	0.02	290	290.00	0.00	0.27
p7k6	42	141	461.00	461.00	0.00	0.01	461	461.00	0.00	0.24	461	461.00	0.00	0.01	461	461.00	0.00	0.08	461	461.00	0.00	1.34
p7k7	49	189	686.00	686.00	0.00	0.01	686	686.00	0.00	0.45	686	686.00	0.00	0.04	686	686.00	0.00	0.24	686	686.00	0.00	2.73
p7k8	56	244	972.00	972.00	0.00	0.02	972	972.00	0.00	0.63	972	972.00	0.00	0.08	972	972.00	0.00	0.46	972	972.00	0.00	9.22
p7k9	63	306	1326.00	1326.00	0.00	0.02	1326	1326.00	0.00	0.77	1326	1326.00	0.00	0.13	1326	1326.00	0.00	1.40	1326	1326.00	0.00	21.03
p8k3	24	45	95.00	95.00	0.00	0.01	95	95.00	0.00	0.01	95	95.00	0.00	0.01	95	95.00	0.00	0.01	95	95.00	0.00	0.02

Table B.6: Detailed results of MACH, MA, BVNS, MA-20, and DMAB+MA for Cartesian products (part six)

			MACH				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
p8k4	32	76	192.00	192.00	0.00	0.01	192	192.00	0.00	0.21	192	192.00	0.00	0.01	192	192.00	0.00	0.02	192	192.00	0.00	0.14
p8k5	40	115	335.00	335.00	0.00	0.01	335	335.00	0.00	0.17	335	335.00	0.00	0.02	335	335.00	0.00	0.04	335	335.00	0.00	0.69
p8k6	48	162	532.00	532.00	0.00	0.01	532	532.00	0.00	0.35	532	532.00	0.00	0.03	532	532.00	0.00	0.15	532	532.00	0.00	2.00
p8k7	56	217	791.00	791.00	0.00	0.01	791	791.00	0.00	0.51	791	791.00	0.00	0.08	791	791.00	0.00	0.64	791	791.00	0.00	6.39
p8k8	64	280	1120.00	1120.00	0.00	0.02	1120	1120.00	0.00	0.79	1120	1120.00	0.00	0.13	1120	1120.00	0.00	1.34	1120	1120.00	0.00	151.16
p8k9	72	351	1527.00	1527.00	0.00	0.02	1527	1527.00	0.00	1.45	1527	1527.00	0.00	0.29	1527	1527.00	0.00	2.52	1527	1527.00	0.00	25.50
p9k3	27	51	108.00	108.00	0.00	0.01	108	108.00	0.00	0.15	108	108.00	0.00	0.01	108	108.00	0.00	0.01	108	108.00	0.00	0.03
p9k4	36	86	218.00	218.00	0.00	0.01	218	218.00	0.00	0.09	218	218.00	0.00	0.01	218	218.00	0.00	0.02	218	218.00	0.00	0.26
p9k5	45	130	380.00	380.00	0.00	0.01	380	380.00	0.00	0.29	380	380.00	0.00	0.02	380	380.00	0.00	0.18	380	380.00	0.00	1.09
p9k6	54	183	603.00	603.00	0.00	0.01	603	603.00	0.00	0.60	603	603.00	0.00	0.06	603	603.00	0.00	0.60	603	603.00	0.00	4.48
p9k7	63	245	896.00	896.00	0.00	0.01	896	896.00	0.00	0.78	896	896.00	0.00	0.15	896	896.00	0.00	1.19	896	896.00	0.00	21.05
p9k8	72	316	1268.00	1268.00	0.00	0.02	1268	1268.00	0.00	0.93	1268	1268.00	0.00	0.21	1268	1268.00	0.00	3.08	1268	1268.00	0.00	28.87
p9k9	81	396	1728.00	1728.00	0.00	0.02	1728	1728.00	0.00	2.41	1728	1728.00	0.00	0.54	1728	1728.00	0.00	5.78	1728	1728.00	0.00	52.67
p3p3	9	12	19.00	19.00	0.00	0.01	19	19.00	0.00	0.01	19	19.00	0.00	0.01	19	19.00	0.00	0.01	19	19.00	0.00	0.01
p4p3	12	17	31.00	33.40	1.98	0.01	29	29.00	0.00	0.01	29	29.00	0.00	0.01	29	29.00	0.00	0.01	29	29.00	0.00	0.01
p4p4	16	24	58.00	61.20	3.96	0.01	44	44.00	0.00	0.01	44	44.00	0.00	0.01	44	44.00	0.00	0.01	44	44.00	0.00	0.01
p5p3	15	22	43.00	43.32	0.47	0.01	42	42.00	0.00	0.34	42	42.00	0.00	0.01	42	42.00	0.00	0.01	42	42.00	0.00	0.01
p5p4	20	31	87.00	93.68	8.07	0.01	63	63.32	2.26	0.27	63	63.00	0.00	0.01	63	63.00	0.00	0.01	63	63.00	0.00	0.01
p5p5	25	40	133.00	153.08	18.33	0.01	90	90.82	4.49	1.20	90	90.00	0.00	0.01	90	90.00	0.00	0.01	90	90.00	0.00	0.02
p6p3	18	27	57.00	58.96	1.48	0.01	55	55.92	0.99	14.88	55	55.00	0.00	0.01	55	55.00	0.00	0.16	55	55.00	0.00	0.05
p6p4	24	38	122.00	137.32	8.23	0.01	82	87.00	7.09	2.34	82	82.00	0.00	0.01	82	82.00	0.00	0.10	82	82.00	0.00	0.03
p6p5	30	49	187.00	217.60	28.89	0.01	117	122.50	12.19	7.30	117	117.00	0.00	0.01	117	117.00	0.00	0.01	117	117.00	0.00	0.02
p6p6	36	60	254.00	301.00	36.34	0.01	152	172.20	24.62	23.37	152	152.00	0.00	0.01	152	152.00	0.00	0.02	152	152.00	0.00	0.05
p7p3	21	32	73.00	73.48	0.50	0.01	68	68.42	1.05	0.14	68	68.00	0.00	0.01	68	68.00	0.00	0.02	68	68.00	0.00	0.02
p7p4	28	45	161.00	181.16	14.83	0.01	105	112.44	8.19	2.42	105	105.00	0.00	0.01	105	106.68	4.15	0.77	105	105.00	0.00	0.45
p7p5	35	58	242.00	288.00	39.09	0.01	149	164.02	15.87	17.77	149	149.00	0.00	0.02	149	149.00	0.00	0.53	149	149.00	0.00	0.17
p7p6	42	71	333.00	417.88	54.34	0.01	193	220.12	27.19	1.55	193	193.00	0.00	0.14	193	193.00	0.00	0.17	193	193.00	0.00	0.16
p7p7	49	84	423.00	529.56	67.13	0.01	245	287.50	42.17	7.66	245	245.00	0.00	0.46	245	245.00	0.00	0.15	245	245.00	0.00	0.21
p8p3	24	37	91.00	92.92	1.56	0.01	79	79.80	2.42	0.90	79	79.00	0.00	0.01	79	79.00	0.00	0.02	79	79.00	0.00	0.01
p8p4	32	52	206.00	224.28	16.38	0.01	128	137.66	8.71	20.94	128	128.00	0.00	0.06	128	132.16	4.04	0.16	128	128.00	0.00	1.86
p8p5	40	67	303.00	372.40	57.13	0.01	181	204.66	19.56	5.80	181	181.00	0.00	0.20	181	189.42	10.75	0.27	181	181.00	0.00	0.79
p8p6	48	82	408.00	502.92	66.05	0.01	234	279.38	37.51	13.38	234	234.00	0.00	0.91	234	234.00	0.00	2.66	234	234.00	0.00	0.38
p8p7	56	97	521.00	677.08	89.76	0.01	297	367.34	54.10	16.63	297	297.00	0.00	3.80	297	297.00	0.00	0.62	297	297.00	0.00	0.44
p8p8	64	112	650.00	824.60	113.40	0.01	361	452.54	64.73	9.56	360	360.00	0.00	10.67	360	360.00	0.00	0.60	360	360.00	0.00	2.46
p9p3	27	42	109.00	109.44	0.50	0.01	90	91.42	4.94	2.19	90	90.00	0.00	0.01	90	90.00	0.00	0.01	90	90.00	0.00	0.01
p9p4	36	59	257.00	280.64	20.51	0.01	155	162.34	12.16	12.09	155	155.00	0.00	0.02	155	155.00	0.00	0.02	155	155.00	0.00	0.60
p9p5	45	76	382.00	465.64	74.71	0.01	218	244.60	21.39	13.45	218	218.00	0.00	0.85	218	222.48	6.60	0.13	218	218.00	0.00	7.09
p9p6	54	93	495.00	618.48	88.24	0.01	282	339.56	40.24	10.63	281	281.00	0.00	9.18	281	290.24	17.57	1.48	281	281.00	0.00	2.02

Table B.7: Detailed results of Mach, MA, BVNS, MA-20, and DMAB+MA for Cartesian products (part seven)

			Mach				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
p9p7	63	110	628.00	823.40	117.41	0.01	356	431.08	67.44	8.88	356	356.00	0.00	35.58	356	360.32	17.27	4.22	356	356.00	0.00	1.42
p9p8	72	127	779.00	1044.16	135.47	0.01	433	549.96	76.40	28.15	431	431.24	0.43	178.65	431	433.24	15.84	3.29	431	431.00	0.00	6.11
p9p9	81	144	944.00	1244.88	152.51	0.01	519	666.34	99.12	42.15	516	516.98	0.55	168.71	516	516.02	0.14	12.20	516	516.00	0.00	40.95

Table B.8: Detailed results of Mach, MA, BVNS, MA-20, and DMAB+MA for Harwell-Boeing graphs (part one)

			Mach				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
young1c	841	1624	54633.00	59331.40	1252.32	0.09	24337	30641.02	3140.20	127.20	18764	21856.96	2632.64	587.52	24914	28818.00	2492.86	232.56	21905	22592.52	369.77	503.13
young2c	841	1624	51347.00	58991.74	2150.85	0.09	21427	30164.88	3886.54	126.18	18448	22221.44	2397.89	578.41	24914	28818.00	2492.86	232.63	21905	22592.52	369.77	502.90
young3c	841	1671	41479.00	60437.96	7493.14	46.47	24954	33046.14	4352.24	120.10	19352	23347.48	2919.67	575.00	26359	30413.24	2626.97	219.17	22869	23898.62	413.18	489.85
young4c	841	1624	52063.00	59225.06	1576.54	0.09	22983	30806.76	3317.10	127.98	18679	21836.42	2245.74	581.59	24914	28818.00	2492.86	232.63	21905	22592.48	369.78	503.04
bcspwr01	39	46	101.00	114.60	7.91	0.01	98	111.48	8.72	26.84	98	98.00	0.00	1.43	98	98.00	0.00	0.66	98	98.00	0.00	2.63
bcspwr02	49	59	157.00	180.90	21.24	0.02	150	179.26	19.88	44.35	148	148.02	0.14	146.74	148	148.00	0.00	0.95	148	148.00	0.00	3.50
bcspwr03	118	179	734.00	951.58	81.79	0.24	746	954.24	104.60	39.89	669	680.82	5.73	112.31	663	668.66	9.78	25.04	662	663.26	0.85	298.04
bcspwr04	274	669	5422.00	6686.28	655.18	3.01	4191	4637.76	369.54	74.06	4090	4334.66	163.31	266.75	4085	4430.34	263.61	84.81	4080	4247.56	126.32	354.55
bcspwr05	443	590	5359.00	6473.10	720.59	7.72	7508	9393.24	779.25	60.03	4442	5000.20	341.86	178.29	4794	5269.72	257.73	115.47	4085	4503.06	167.16	344.57
bcspwr06	1454	1923	28146.00	40827.76	5820.56	254.99	23500	32269.62	7005.98	132.23	29022	32251.06	2258.35	591.18	52941	61872.86	3064.88	233.47	26758	34439.36	2949.44	334.15
bcspwr07	1612	2106					29105	42193.74	13078.93	137.24	32855	37124.82	2396.10	604.74	68849	76775.10	3752.52	254.15	33317	42714.66	4607.67	347.92
bcspwr08	1624	2213					30770	49115.82	14818.43	133.06	38652	44496.46	3212.08	605.10	77958	84503.08	3530.63	250.15	38542	47475.10	4371.12	349.38
bcspwr09	1723	2394					147155	164380.56	11447.42	97.27	46453	59326.02	8820.82	606.96	87183	97766.88	4583.09	237.28	43296	63542.36	9245.17	322.61
bcspwr10	5300	8271					1821769	2197831.46	152125.22	163.63	416877	503657.08	61079.57	1000.98	1968924	2161344.08	78452.62	280.33	800571	1115294.44	145487.63	381.28
bcsstk01	48	176	1158.00	1329.44	113.42	0.01	936	955.72	25.31	29.37	936	936.00	0.00	11.86	936	937.60	3.23	16.80	936	936.00	0.00	5.53
bcsstk02	66	2145	35937.00	35937.00	0.00	0.13	35937	35937.00	0.00	0.01	35937	35937.00	0.00	0.01	35937	35937.00	0.00	0.01	35937	35937.00	0.00	0.11
bcsstk04	132	1758	35091.00	37724.08	1553.58	0.98	29812	30126.94	347.92	36.76	29812	29812.04	0.28	113.75	29812	30064.72	308.12	63.91	29812	29912.48	109.97	339.41
bcsstk05	153	1135	13904.00	16378.94	2153.00	0.82	11059	11333.56	407.25	12.14	11059	11059.04	0.28	56.76	11059	11059.00	0.00	8.07	11059	11059.00	0.00	300.03
bcsstk06	420	3720	65461.00	83423.66	6709.56	28.47	55484	67213.28	6547.29	19.73	55340	63102.14	3193.90	536.73	53489	59554.20	5223.61	213.36	51849	57484.16	4083.81	353.15
bcsstk07	420	3720	66229.00	83213.26	7563.80	28.59	53466	68658.02	7815.22	19.16	55244	62602.78	3060.75	532.29	53489	59540.32	5219.81	214.67	51849	57484.38	4083.98	353.17
bcsstm07	420	3416	64120.00	76820.90	6705.07	26.06	49665	63439.64	7279.80	15.93	49406	58944.18	4465.45	506.29	48609	53072.22	5136.76	199.22	47492	53341.18	3908.72	334.84
bcsstk19	817	3018	50190.00	72248.48	12317.71	69.16	23106	36483.20	8874.21	90.04	30665	48995.12	6293.72	588.64	38519	46721.02	4442.51	240.26	22591	29827.10	3323.54	351.72
can_24	24	68	212.00	248.40	14.46	0.02	182	187.60	11.31	0.38	182	182.00	0.00	0.01	182	182.00	0.00	0.11	182	182.00	0.00	0.03
can_61	61	248	1285.00	1556.24	181.69	0.03	1137	1167.36	46.18	25.33	1137	1137.00	0.00	0.31	1137	1137.00	0.00	5.98	1137	1137.00	0.00	3.06
can_62	62	78	228.00	259.84	18.23	0.03	193	251.16	26.23	29.13	192	193.74	0.88	181.00	192	193.20	4.20	7.40	192	192.00	0.00	10.54
can_73	73	152	956.00	1062.12	67.75	0.10	825	840.92	47.51	10.87	825	825.96	0.64	239.40	825	825.00	0.00	3.59	825	825.00	0.00	25.05
can_96	96	336	2071.00	2637.64	287.27	0.20	1616	1766.02	292.45	3.90	1616	1616.00	0.00	18.41	1616	1646.88	152.82	1.09	1616	1616.00	0.00	4.10
can_144	144	576	2250.00	2258.58	6.88	0.02	1776	3293.80	563.72	75.17	1776	1776.00	0.00	2.87	1776	2516.32	748.99	7.64	1776	1776.00	0.00	44.13
can_161	161	608	6657.00	7841.26	739.00	0.97	5010	5302.92	470.41	40.88	5002	5008.08	2.35	172.19	4999	5062.32	109.11	38.80	4998	4999.80	0.64	306.41
can_187	187	652	3809.00	5836.16	1078.78	1.33	3059	4488.76	1226.15	9.59	3059	3068.20	3.64	219.58	3059	3290.66	696.89	52.01	3059	3059.00	0.00	94.51
can_229	229	774	8320.00	11843.46	2289.70	2.05	6269	7473.28	1287.56	37.46	6273	6293.72	8.29	97.93	6262	6278.06	5.40	58.60	6246	6250.60	1.92	294.20
can_256	256	1330	25856.00	33708.14	4223.68	2.42	19220	20332.60	821.67	66.40	19122	19255.70	105.62	204.16	19132	19531.24	484.65	119.22	19102	19126.62	35.76	292.38
can_268	268	1407	29800.00	37033.62	4689.48	3.44	19880	21320.32	1158.65	55.57	19503	19682.56	281.84	244.82	19521	20022.30	719.02	119.25	19503	19685.26	226.75	189.77
can_292	292	1124	20766.00	25929.08	2141.71	6.53	15133	17285.68	1955.14	91.16	15139	15256.82	423.53	216.84	15127	15781.26	1082.28	108.48	15109	15125.90	7.04	282.09
can_445	445	1682	40448.00	50717.72	3812.98	17.12	27249	30728.36	2084.31	62.06	26733	27369.88	574.59	350.30	27865	28903.22	498.09	134.94	26634	26767.32	53.67	386.96

Table B.9: Detailed results of MACH, MA, BVNS, MA-20, and DMAB+MA for Harwell-Boeing graphs (part two)

			MACH				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
can_634	634	3297	103540.00	129700.74	12673.74	56.31	73165	82995.66	3989.76	85.69	76908	91065.08	5904.69	568.94	73056	76571.98	2687.48	205.31	68527	72524.42	1829.20	335.20
can_715	715	2975	86257.00	113971.48	11437.74	72.65	63156	75799.94	8528.26	79.25	62404	72781.02	8967.44	580.27	69103	71054.40	1926.73	169.94	60349	64689.94	1392.98	309.31
can_838	838	4586	216608.00	346290.46	60038.66	122.41	124637	143583.44	9133.71	76.34	106252	129454.90	12091.13	599.30	109602	113604.50	6589.60	181.68	94520	97764.96	3549.58	318.99
can_1054	1054	5571					211112	246809.36	15159.96	63.22	173438	200473.06	17780.45	606.09	172864	187501.56	14659.66	224.34	143632	183913.32	23277.15	160.35
can_1072	1072	5686					223036	258367.88	16703.11	70.99	195409	221738.52	14554.74	613.49	181053	192547.34	13427.67	210.70	150715	191977.18	32628.30	156.86
impcol_b	59	281	2421.00	2819.72	197.30	0.06	1822	1870.66	73.77	12.03	1822	1822.00	0.00	0.30	1822	1822.00	0.00	0.20	1822	1822.00	0.00	0.56
impcol_c	137	352	4742.00	5544.64	336.59	0.41	3410	3795.30	307.26	30.99	3354	3366.00	3.78	173.50	3351	3412.60	132.12	42.46	3350	3417.72	88.69	371.19
impcol_d	425	1267	22858.00	32632.66	5855.78	12.08	13712	20690.52	2662.58	36.57	13341	16409.54	1536.03	427.99	12821	16372.84	2652.91	122.12	12162	15423.72	2519.64	403.15
impcol_e	225	1187	28081.00	34173.28	3188.15	3.42	15652	17242.18	1549.69	32.07	15523	15904.56	562.16	364.47	15511	15950.80	649.48	87.01	15491	15967.84	640.42	322.61
west0067	67	287	3337.00	3473.70	164.26	0.04	2409	2494.74	64.38	19.90	2407	2407.10	0.30	134.44	2407	2429.96	36.68	21.75	2407	2407.00	0.00	11.38
west0132	132	404	6205.00	6911.84	277.38	0.33	4814	5225.42	163.45	54.85	4780	4800.44	7.70	150.48	4770	4899.52	123.70	46.05	4768	4770.26	2.17	313.58
west0156	156	371	7120.00	7754.18	311.16	0.20	4638	4933.64	236.33	103.12	4626	4645.90	9.10	163.01	4611	4698.14	113.01	37.17	4604	4608.16	2.09	281.93
west0167	167	489	8203.00	10799.10	1119.05	0.49	5547	6055.60	585.95	24.72	5534	5550.24	5.96	139.59	5523	5534.46	15.65	49.19	5516	5519.00	1.12	325.56
west0381	381	2150	155096.00	164648.20	4385.82	6.46	100253	103016.14	1762.89	89.63	100219	102080.98	1502.74	413.06	100225	102093.04	1687.81	178.48	100209	100943.06	735.66	486.20
west0479	479	1889	124487.00	132887.40	4214.77	17.26	69966	71983.00	1473.18	104.72	70382	72322.62	1641.20	441.60	70740	72262.30	1023.44	157.25	69875	71406.26	589.10	267.00
west0497	497	1715	78320.00	91703.86	5504.76	21.49	46797	52658.56	3326.23	60.11	46387	49440.00	2544.32	492.99	47796	50782.20	2263.08	135.27	46702	47822.00	766.96	301.64
west0655	655	2841	266643.00	283637.26	9641.30	48.27	143668	147081.36	3699.50	109.00	143763	153218.60	6028.91	577.74	147148	150303.34	3539.90	192.78	143808	147321.98	1184.26	281.10
west0989	989	3500	464396.00	503147.48	14242.60	134.17	252128	259890.02	3615.15	88.46	250264	263139.04	8880.93	609.26	257352	265162.86	4418.41	222.34	246154	255459.14	3069.51	384.37
west1505	1505	5437					566391	583878.54	8751.31	82.66	540081	584276.78	27432.88	645.61	561648	588878.52	18289.88	232.03	542714	561671.64	7584.02	286.71
jgl009	9	32	76.00	76.00	0.00	0.01	75	75.00	0.00	0.01	75	75.00	0.00	0.01	75	75.00	0.00	0.01	75	75.00	0.00	0.01
jgl011	11	49	142.00	142.00	0.00	0.01	141	141.00	0.00	0.01	141	141.00	0.00	0.01	141	141.00	0.00	0.01	141	141.00	0.00	0.01
dwt_59	59	104	295.00	362.22	37.66	0.05	267	334.70	36.07	20.30	235	235.00	0.00	76.06	235	237.94	11.78	8.40	235	235.00	0.00	26.65
dwt_66	66	127	192.00	192.00	0.00	0.02	192	196.76	16.50	9.25	192	192.00	0.00	0.27	192	192.00	0.00	2.10	192	192.00	0.00	0.34
dwt_72	72	75	200.00	224.96	15.45	0.05	175	214.18	22.24	35.24	174	183.24	4.01	186.96	167	172.46	4.67	15.90	167	167.00	0.00	26.14
dwt_87	87	227	1228.00	1428.14	153.66	0.11	934	1089.12	138.86	57.02	932	934.32	1.49	230.09	932	932.78	2.81	15.57	932	932.00	0.00	115.42
dwt_162	162	510	2484.00	3729.20	457.36	0.82	1851	3213.16	726.55	7.51	1837	1840.34	2.52	184.29	1837	2021.72	379.20	32.33	1837	1888.94	210.91	353.75
dwt_193	193	1650	29040.00	34932.64	3776.18	2.39	22954	24360.06	1237.76	32.38	22944	22956.68	7.94	195.98	22934	23527.60	834.77	117.96	22935	23004.68	315.65	331.06
dwt_209	209	767	8093.00	10183.06	1015.40	1.83	6989	8162.32	653.46	39.80	6372	6436.26	28.13	167.51	6365	6660.92	339.59	67.21	6369	6390.92	20.02	310.68
dwt_221	221	704	5566.00	7196.58	860.60	2.00	4007	6012.72	865.11	46.32	3789	3803.12	8.38	56.55	3782	3979.28	451.88	70.19	3775	3779.18	2.50	381.86
dwt_245	245	608	5613.00	6666.00	599.85	2.20	4704	5931.46	752.52	37.43	3895	4046.10	133.61	167.19	3873	4225.10	343.70	71.99	3859	4000.24	137.08	361.67
dwt_307	307	1108	15341.00	18482.76	1931.10	5.70	10841	12826.94	1492.39	46.95	10704	10764.56	28.98	192.74	10770	10971.96	220.65	97.21	10661	10992.34	293.87	164.16
dwt_310	310	1069	9019.00	12583.92	2173.24	5.68	6575	11008.68	1844.62	28.17	6482	6510.36	11.96	120.63	6527	6767.42	497.57	102.84	6454	6802.30	700.28	392.42
dwt_361	361	1296	16456.00	20995.80	2889.29	9.15	12243	15754.06	1993.90	40.85	12153	12234.72	40.22	78.04	12356	12667.96	131.34	102.30	12061	12069.86	10.71	362.54
dwt_419	419	1572	21652.00	26142.92	2364.15	14.13	15686	17090.32	1480.08	110.41	14609	16421.66	1328.63	424.96	15596	17304.74	1402.09	157.44	14820	15577.26	230.21	280.55
dwt_503	503	2762	54075.00	68529.42	7833.72	29.92	37775	46268.12	6849.64	50.47	37615	43103.36	4501.87	519.30	40617	43240.72	3061.27	158.32	37448	37703.96	149.62	377.62
dwt_592	592	2256	35978.00	45235.26	5132.62	40.39	30480	38511.56	5168.36	86.81	26186	29539.58	2745.75	522.48	30272	32162.52	1798.42	179.38	25076	27677.52	2255.33	292.92
dwt_758	758	2618	28018.00	44070.40	7573.91	77.90	29679	41077.08	5319.71	76.79	15979	25561.32	7763.38	576.48	31052	37748.16	3773.36	237.82	15444	17980.52	1533.73	380.66
dwt_869	869	3208	49501.00	68702.68	8539.18	122.95	42792	64780.40	6784.37	122.00	33723	42910.32	6117.69	594.56	52383	58019.70	4417.83	226.94	31395	34167.66	2194.98	470.06

Table B.10: Detailed results of Mach, MA, BVNS, MA-20, and DMAB+MA for Harwell-Boeing graphs (part three)

			Mach				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
dwt_878	878	3285	62429.00	86857.44	10698.00	127.62	58814	72896.22	8789.64	131.39	48816	59601.20	9719.17	603.12	61065	65613.48	2558.83	217.27	45856	47927.32	2278.56	535.69
dwt_918	918	3233	84089.00	110641.74	13674.40	141.05	46994	58875.80	5471.66	94.66	39397	49768.22	7914.90	597.59	58792	71284.84	6711.59	236.30	38290	40057.52	797.50	389.48
dwt_1005	1005	3808					66979	80054.74	7826.33	119.66	68480	80082.66	7778.23	599.11	83937	95838.52	7392.76	240.76	64108	71889.56	2722.49	330.99
dwt_1007	1007	3784					57237	91398.90	14147.92	136.36	53474	74429.78	16821.02	605.10	76748	85346.44	6793.74	222.49	51228	54573.56	4099.79	536.61
dwt_1242	1242	4592	140302.00	173521.72	17294.16	352.63	97238	115759.00	12067.31	132.89	105536	127545.68	16962.39	614.97	135391	151390.66	8323.75	245.23	93740	102532.32	5729.87	362.19
jagmesh1	936	2664	37773.00	61974.34	9619.89	134.18	23464	39159.92	7529.96	132.24	23477	37426.80	5821.20	587.17	39438	49389.14	6247.76	228.98	23459	23531.94	74.42	392.54
jagmesh2	1009	2928					43845	51874.94	6000.33	136.75	44714	55480.66	6283.22	596.75	60534	67575.26	4162.29	229.65	42043	45445.02	2185.00	389.20
jagmesh3	1089	3136					43843	57661.88	7396.96	129.85	49295	57674.76	7211.66	596.77	69843	78141.16	4804.70	232.77	43547	47009.30	2866.53	376.61
jagmesh4	1440	4032					56007	72555.64	9377.58	142.22	47447	57384.56	5122.39	611.20	117688	133381.22	9318.80	239.97	28948	37820.80	8962.84	545.40
jagmesh5	1180	3285	58311.00	76269.06	11055.94	264.80	37623	48936.68	5992.95	141.25	37329	46275.72	6592.99	589.84	70276	82695.40	5097.72	233.60	34970	40839.94	2768.25	409.03
jagmesh6	1377	3808					41179	60499.82	9774.96	143.39	35511	48084.30	9035.13	603.75	101758	113588.40	6064.43	235.62	30177	36429.60	4573.56	520.05
jagmesh7	1138	3156					39714	48745.66	5790.34	138.75	37783	44296.26	3837.53	605.71	68707	78878.24	5220.98	235.18	29129	33160.34	3382.75	383.10
jagmesh8	1141	3162	70601.00	86014.30	7857.62	237.68	41668	55717.64	7058.97	139.96	44366	50581.50	5195.28	592.06	74023	84160.68	5237.84	209.24	42728	45101.10	1845.00	448.64
jagmesh9	1349	3876	82048.00	125564.52	19030.08	397.69	48391	78428.22	12525.27	142.87	54154	73762.40	12097.89	604.54	105374	126301.20	9083.93	256.13	47831	54359.82	7677.26	457.53
nos4	100	247	1220.00	1564.42	299.45	0.07	1033	1294.90	226.02	7.71	1031	1031.02	0.14	105.60	1031	1031.00	0.00	2.79	1031	1031.00	0.00	42.84
nos5	468	2352	96626.00	113888.38	10036.91	15.03	56803	64252.76	5957.96	22.99	55772	58991.16	4245.80	521.75	57633	61449.18	3802.01	149.75	55691	57910.52	3331.97	373.87
nos6	675	1290	33323.00	47184.62	5272.95	1.77	16838	20671.88	2334.70	111.59	12334	14290.28	956.52	457.29	16108	17224.66	1077.43	177.89	13205	15343.46	741.52	453.61
nos7	729	1944	94564.00	119962.14	8861.21	10.54	42487	53579.30	6401.00	42.28	42195	46792.38	4031.65	589.25	48878	50442.14	2301.37	160.44	41176	45610.92	5402.92	335.08
494_bus	494	586	4910.00	5703.78	490.91	9.01	10984	12677.10	735.37	79.33	4113	4996.68	448.35	169.95	5160	5600.60	220.86	118.43	4496	4891.28	197.83	347.85
662_bus	662	906	11992.00	15955.78	1617.86	28.72	24416	28407.14	1839.66	82.28	9296	11322.64	965.55	481.23	11176	12589.84	719.90	182.30	9238	10485.30	613.80	294.59
685_bus	685	1282	13594.00	17574.32	2242.16	34.02	28364	33641.70	2540.00	87.67	11157	14861.12	1963.67	541.32	15541	16849.06	658.83	165.72	10254	11814.68	625.54	292.73
ash85	85	219	1190.00	1431.06	148.65	0.13	913	1020.78	79.89	70.43	913	913.02	0.14	159.87	913	931.92	21.56	11.68	913	913.00	0.00	44.71
curtis54	54	124	470.00	608.06	88.56	0.03	411	456.96	34.15	47.72	411	411.00	0.00	53.44	411	411.00	0.00	8.44	411	411.00	0.00	51.62
ibm32	32	90	493.00	541.12	22.73	0.01	405	411.96	6.55	23.36	405	405.00	0.00	0.47	405	406.68	3.20	3.13	405	405.00	0.00	0.20
pores_1	30	68	207.00	207.00	0.00	0.01	185	192.82	6.59	11.65	185	185.00	0.00	0.02	185	187.52	3.72	4.90	185	185.00	0.00	1.16
will57	57	127	408.00	451.20	54.07	0.03	336	396.56	35.08	9.41	335	335.00	0.00	16.35	335	335.26	1.84	2.19	335	335.00	0.00	10.53
pores_2	1224	5822	262697.00	337697.96	29535.77	249.24	163369	207296.54	25794.59	58.64	201405	256167.38	36342.13	630.39	214843	234017.60	12997.06	227.96	165251	190735.82	13357.54	326.35
1138_bus	1138	1458	21615.00	26378.40	2514.16	125.17	35173	43929.80	3509.11	91.40	20076	24374.16	2229.53	586.84	32420	35406.70	1395.63	225.07	18629	23365.36	1992.09	357.35
abb313	313	1553	51478.00	56958.54	2654.01	11.69	37973	40729.62	2686.62	110.17	37998	38596.78	1158.98	223.95	38002	39034.52	1442.54	120.27	37972	38022.88	30.35	273.36
arc130	130	715	14742.00	16200.00	601.27	2.04	14398	14411.22	9.25	64.84	14397	14397.00	0.00	10.19	14397	14620.08	277.90	25.07	14397	14397.00	0.00	60.52
ash219	219	431	8254.00	8888.10	338.02	2.42	6232	6433.40	191.54	79.90	6264	6299.56	16.66	89.78	6254	6451.24	198.47	38.04	6237	6247.74	4.28	299.92
ash292	292	958	9757.00	11833.86	1320.16	4.58	7452	10509.68	1459.55	24.00	6452	6563.28	60.88	133.42	6492	6629.04	76.09	96.65	6402	6571.70	255.20	379.94
ash331	331	660	16444.00	18542.44	956.30	8.69	13662	14160.94	454.47	88.30	13788	13899.90	73.94	185.52	13749	14148.64	315.44	51.75	13664	13752.12	50.05	302.49
ash608	608	1212	54640.00	60047.66	3205.86	50.60	42983	43710.18	676.57	70.29	39581	42810.32	1718.43	518.75	39910	41211.70	1383.92	153.14	41450	43422.44	660.26	276.58
ash958	958	1912	127263.00	142242.36	7228.99	193.75	75428	77244.56	2123.78	90.23	76352	90569.38	6852.74	596.95	81279	86752.72	4841.72	207.21	77359	79729.38	1073.80	287.40
bp_0	822	3260	420057.00	440298.30	9522.66	106.62	223931	231994.86	3344.34	86.10	223073	232182.70	5587.33	586.87	222470	226663.78	3554.60	216.20	216660	222938.86	1591.31	336.51
bp_200	822	3788	535516.00	555385.82	9815.36	109.67	294102	301105.82	2752.04	75.10	290600	305269.76	5914.66	591.98	289981	295653.56	3803.98	229.39	287475	290985.24	1670.82	260.11
bp_400	822	4015	572167.00	596704.38	9497.85	114.25	324060	329346.28	2766.97	90.17	318247	332603.40	8265.38	596.02	318009	324172.94	5490.57	221.25	316164	319131.72	1395.22	266.14

Table B.11: Detailed results of Mach, MA, BVNS, MA-20, and DMAB+MA for Harwell-Boeing graphs (part four)

			Mach				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
bp_600	822	4157	602503.00	629506.40	12134.44	100.85	351209	356786.48	2899.07	84.74	342239	358065.46	6642.59	604.68	340582	346804.32	4354.77	216.61	337960	344438.24	2071.26	328.96
bp_800	822	4518	642744.00	682406.42	12845.75	124.12	385821	394573.04	3876.95	85.26	375938	389350.96	8475.83	581.14	377262	384494.30	4799.71	211.43	374300	381337.30	2388.42	245.51
bp_1000	822	4635	679857.00	704191.44	13126.95	115.83	402496	409493.08	3721.98	112.94	387928	403023.90	8210.95	599.26	391698	398680.60	4719.19	217.26	386326	394067.24	2978.83	328.38
bp_1200	822	4698	684878.00	722660.52	12759.37	110.82	407185	416958.12	3965.48	84.00	395464	408103.36	7820.06	600.68	398286	403586.50	4832.61	224.66	393936	400254.84	3086.96	292.68
bp_1400	822	4760	711741.00	732518.48	12835.04	114.31	418067	425577.46	3931.56	72.15	399548	413866.98	8808.66	591.90	405614	412433.44	4757.19	196.70	404171	411182.86	2486.09	313.49
bp_1600	822	4809	699646.00	738986.70	17753.96	118.46	413006	419124.52	3041.27	99.20	400758	415325.08	8837.94	598.25	402934	409152.34	5404.32	215.44	400206	405558.82	2029.27	350.76
fs_541_1	541	2466	109111.00	120966.20	5202.30	24.91	91002	97085.22	2241.08	73.59	101251	109599.56	4423.86	590.82	92556	96543.46	2673.44	191.94	88449	90027.60	586.81	282.09
fs_541_2	541	2466	109280.00	120169.20	6346.99	25.37	92132	97180.78	2018.10	78.06	99100	109587.18	5074.71	588.48	92556	96543.46	2673.44	191.97	88449	90027.60	586.81	282.12
fs_541_3	541	2466	108525.00	120257.80	6022.25	25.77	92132	97138.14	1821.39	85.46	100882	109938.84	5102.33	593.93	92556	96543.46	2673.44	191.93	88449	90027.60	586.81	282.12
fs_541_4	541	2466	106898.00	121138.06	5902.71	25.89	93227	97275.70	1908.90	76.06	100192	109637.50	5400.01	591.54	92556	96543.46	2673.44	191.91	88449	90027.60	586.81	282.09
lund_a	147	1151	12160.00	14750.48	1455.05	1.13	10171	11068.34	772.58	8.27	10165	10165.68	0.51	224.80	10165	10444.64	502.68	58.71	10165	10165.00	0.00	315.13
lund_b	147	1147	11758.00	15092.28	1347.61	1.10	10192	11217.82	940.98	8.19	10160	10162.30	2.35	218.12	10160	10528.50	523.47	51.20	10160	10160.08	0.34	325.65
shl_0	663	1682	146233.00	150985.36	2605.65	38.58	123072	126022.76	1425.56	79.38	113423	117301.26	1582.05	562.39	114748	116095.34	715.94	182.55	114028	117780.04	1066.26	312.34
shl_200	663	1720	150394.00	158115.94	2935.22	39.36	129804	132504.30	1491.63	63.05	119137	123209.10	1697.45	578.76	120698	121877.52	780.98	203.42	119986	123034.72	1013.09	303.05
shl_400	663	1709	150164.00	156132.60	2767.23	40.15	131422	134568.34	1479.94	76.40	121819	124156.54	1502.49	572.03	121741	123089.72	748.37	198.76	121936	124890.52	1095.37	293.01
str_0	363	2446	153765.00	167767.26	4943.70	18.36	63958	65505.86	1437.31	99.78	63977	66867.34	2552.32	463.48	64105	65841.08	1228.68	192.03	63928	64287.58	138.91	305.66
str_200	363	3049	203298.00	220102.54	7125.11	15.74	94943	96531.06	1794.87	101.74	94488	99327.02	4041.34	418.51	94421	97195.84	2937.03	140.96	94737	95746.90	404.16	255.99
str_400	363	3124	211966.00	227214.26	8001.62	13.53	99960	100952.78	770.73	90.98	99645	102630.96	2821.32	328.92	100007	102149.74	2360.46	160.59	99891	100353.54	207.02	308.96
str_600	363	3244	207363.00	234430.02	10192.94	12.56	105396	107034.54	1692.03	100.24	105175	110888.84	3880.90	412.58	105543	109188.28	3311.26	185.87	105184	105917.52	265.14	306.54
will199	199	660	19667.00	21096.92	704.46	0.27	13721	13910.02	133.83	82.57	13727	13772.94	23.09	158.31	13708	13818.04	145.37	61.22	13703	13716.36	6.29	310.43

Table B.12: Detailed results of Mach, MA, BVNS, MA-20, and DMAB+MA for random graphs (part one)

			Mach				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
rand1N100_1	100	498	9651.00	9984.66	180.96	0.03	7672	7861.06	113.73	77.98	7625	7635.92	6.50	116.92	7614	7683.94	54.18	60.97	7614	7616.82	5.52	251.78
rand1N100_2	100	510	9779.00	10146.46	278.00	0.11	7778	7909.44	103.49	96.11	7749	7764.60	8.83	169.46	7742	7795.24	46.13	72.88	7742	7744.40	1.76	281.19
rand1N100_3	100	513	9906.00	9910.02	2.47	0.21	7904	8092.10	135.42	85.36	7898	7912.38	6.58	133.87	7892	7952.50	69.23	64.45	7892	7894.60	1.54	315.89
rand1N100_4	100	494	10008.00	10120.44	77.50	0.10	7584	7750.46	109.75	95.05	7546	7560.38	7.52	121.83	7540	7599.60	41.65	56.10	7540	7541.80	5.54	282.68
rand1N100_5	100	450	8512.00	9103.80	400.82	0.07	6635	6809.66	115.14	73.10	6637	6648.30	6.66	146.41	6627	6683.38	61.99	61.35	6627	6630.08	2.24	300.53
rand1N100_6	100	461	8400.00	8881.80	217.67	0.12	6817	6990.28	107.61	94.30	6811	6824.50	7.11	158.72	6802	6867.52	66.04	60.19	6802	6803.10	1.05	299.68
rand1N100_7	100	510	9779.00	10056.90	218.15	0.10	7764	7930.48	101.72	75.03	7748	7764.60	7.64	171.62	7742	7795.24	46.13	72.87	7742	7744.40	1.76	281.20
rand1N100_8	100	513	9906.00	9910.20	2.35	0.21	7899	8103.48	146.20	84.19	7900	7913.68	7.23	166.02	7892	7952.50	69.23	64.47	7892	7894.60	1.54	315.91
rand1N100_9	100	494	10008.00	10101.82	81.09	0.10	7561	7720.62	99.98	100.55	7548	7565.12	9.37	117.87	7540	7599.60	41.65	56.10	7540	7541.80	5.54	282.66
rand1N100_10	100	450	8514.00	9083.84	414.73	0.08	6635	6817.24	111.23	95.19	6636	6646.58	6.53	144.98	6627	6683.38	61.99	61.36	6627	6630.08	2.24	300.56
rand3N100_1	100	1422	33128.00	33128.00	0.00	0.25	28298	28578.08	173.92	59.59	28234	28258.92	12.43	195.77	28241	28335.74	83.41	128.69	28231	28241.32	5.46	305.87
rand3N100_2	100	1482	36283.00	36775.40	425.73	0.20	29694	30014.98	201.51	64.94	29614	29628.08	7.56	157.36	29610	29715.66	86.42	126.88	29610	29612.02	1.76	300.29
rand3N100_3	100	1528	36030.00	36030.00	0.00	0.53	30975	31408.88	199.99	74.66	30951	30970.04	12.80	145.25	30945	31070.06	89.92	134.39	30945	30947.88	1.98	319.70
rand3N100_4	100	1469	34845.00	34871.66	21.08	0.34	29422	29740.42	167.46	60.41	29319	29345.70	20.72	163.03	29319	29436.34	74.22	125.96	29319	29323.36	10.78	286.96
rand3N100_5	100	1455	34508.00	34599.20	85.03	0.17	29108	29384.98	189.84	83.23	28984	29002.90	12.01	150.67	28979	29099.60	92.67	152.73	28979	28981.86	3.56	313.35
rand3N100_6	100	1422	33128.00	33128.00	0.00	0.25	28349	28586.62	150.29	69.27	28238	28261.78	12.95	134.44	28241	28335.74	83.41	128.71	28231	28241.32	5.46	305.85
rand3N100_7	100	1482	36283.00	36749.88	437.64	0.21	29686	30024.80	222.84	64.78	29613	29629.38	8.75	128.01	29610	29715.66	86.42	126.89	29610	29612.02	1.76	300.29
rand3N100_8	100	1528	36030.00	36030.00	0.00	0.53	31062	31391.40	152.16	71.61	30952	30967.74	11.54	139.96	30945	31070.06	89.92	134.43	30945	30947.88	1.98	319.66

Table B.13: Detailed results of Mach, MA, BVNS, MA-20, and DMAB+MA for random graphs (part two)

			Mach				MA				BVNS				MA-20				DMAB+MA			
Graph	$\lvert V\rvert$	$\lvert E\rvert$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
rand3N100_9	100	1469	34845.00	34868.22	21.65	0.34	29460	29714.34	170.61	57.83	29320	29342.94	17.71	172.46	29319	29436.34	74.22	125.90	29319	29323.36	10.78	286.97
rand3N100_10	100	1455	34508.00	34582.24	84.61	0.17	29043	29331.74	177.53	83.73	28981	29005.10	10.30	158.68	28979	29099.60	92.67	152.77	28979	28981.86	3.56	313.34
rand5N10_1	10	20	43.00	43.00	0.00	0.01	41	41.00	0.00	0.01	41	41.00	0.00	0.01	41	41.00	0.00	0.01	41	41.00	0.00	0.01
rand5N100_1	100	2399	61177.00	61177.00	0.00	0.17	52042	52399.76	254.12	32.70	51947	51968.56	13.04	113.89	51944	52049.72	116.14	184.15	51944	51948.30	3.63	341.25
rand5N100_10	100	2464	62245.00	62245.00	0.00	0.28	53792	54202.04	198.22	57.10	53714	53736.92	10.04	137.67	53712	53858.30	142.69	170.53	53712	53715.08	3.01	304.72
rand5N100_2	100	2505	63861.00	63861.00	0.00	0.33	54758	55119.84	246.93	50.91	54657	54671.88	9.31	117.24	54654	54777.30	106.01	153.38	54654	54656.68	1.57	329.50
rand5N100_3	100	2526	62874.00	62875.26	1.50	0.58	55416	55885.10	236.57	44.02	55321	55337.66	11.08	120.13	55317	55475.66	136.85	192.72	55317	55321.08	1.68	339.12
rand5N100_4	100	2453	61052.00	62094.30	933.00	0.32	53525	53942.50	199.21	42.67	53486	53512.98	20.97	183.62	53480	53616.90	103.00	167.85	53480	53487.16	11.26	274.83
rand5N100_5	100	2464	62245.00	62245.00	0.00	0.28	53833	54210.74	198.86	46.46	53717	53737.82	10.97	148.74	53712	53858.30	142.69	170.54	53712	53715.10	3.00	302.41
rand5N100_6	100	2399	61177.00	61177.00	0.00	0.17	52084	52367.00	203.84	63.89	51949	51968.54	9.19	153.94	51944	52049.72	116.14	184.14	51944	51948.30	3.63	341.28
rand5N100_7	100	2505	63861.00	63861.00	0.00	0.33	54761	55122.76	215.36	56.54	54657	54671.64	9.03	159.89	54654	54777.30	106.01	153.36	54654	54656.68	1.57	329.52
rand5N100_8	100	2526	62874.00	62875.32	1.50	0.58	55431	55863.48	239.78	71.35	55325	55336.66	9.49	130.95	55317	55475.66	136.85	192.70	55317	55321.08	1.68	339.14
rand5N100_9	100	2453	61052.00	62019.94	938.96	0.33	53645	53953.46	191.16	36.51	53483	53508.16	15.59	218.30	53480	53616.94	103.04	167.84	53480	53487.16	11.26	274.86
rand5N11_1	11	25	57.00	58.06	0.68	0.01	55	55.00	0.00	0.01	55	55.00	0.00	0.01	55	55.00	0.00	0.01	55	55.00	0.00	0.01
rand5N11_2	11	28	76.00	76.00	0.00	0.01	63	63.00	0.00	0.01	63	63.00	0.00	0.01	63	63.00	0.00	0.01	63	63.00	0.00	0.01
rand5N12_1	12	37	105.00	110.20	2.56	0.01	93	93.00	0.00	0.02	93	93.00	0.00	0.01	93	93.00	0.00	0.01	93	93.00	0.00	0.01
rand5N12_2	12	25	59.00	61.96	1.77	0.01	56	56.44	0.50	1.24	56	56.00	0.00	0.01	56	56.00	0.00	0.01	56	56.00	0.00	0.02
rand5N15_1	15	58	192.00	196.94	4.35	0.01	178	178.26	0.75	0.12	178	178.00	0.00	0.01	178	178.00	0.00	0.01	178	178.00	0.00	0.02
rand5N20_1	20	98	460.00	466.52	2.70	0.02	398	398.98	1.76	27.13	398	398.00	0.00	0.02	398	398.00	0.00	0.37	398	398.00	0.00	0.11
rand5N30_1	30	210	1487.00	1488.00	1.01	0.02	1269	1273.52	10.51	29.74	1269	1269.00	0.00	0.44	1269	1269.00	0.00	0.23	1269	1269.00	0.00	0.25
rand5N40_1	40	399	3706.00	3740.72	58.69	0.04	3308	3337.40	17.36	23.73	3308	3308.00	0.00	10.53	3308	3313.00	8.50	21.37	3308	3308.00	0.00	4.52
rand7N10_1	10	27	59.00	62.00	3.03	0.01	56	56.00	0.00	0.01	56	56.00	0.00	0.01	56	56.00	0.00	0.01	56	56.00	0.00	0.01
rand7N100_1	100	3426	87406.00	88033.06	327.67	0.51	78729	79059.78	153.47	56.73	78648	78670.62	12.35	111.00	78636	78786.98	119.79	194.80	78636	78646.24	5.02	350.01
rand7N100_2	100	3499	89233.00	89888.08	460.71	0.52	80949	81215.18	131.78	58.96	80846	80876.86	20.11	152.56	80833	80952.24	89.85	192.95	80836	80848.18	7.18	336.31
rand7N100_3	100	3464	88499.00	88781.14	276.60	0.75	79890	80182.04	179.35	44.71	79796	79824.64	15.23	174.62	79792	79913.48	89.14	193.27	79796	79803.90	3.71	335.62
rand7N100_4	100	3397	85915.00	85915.00	0.00	0.55	78083	78413.46	156.78	47.50	77977	78010.60	21.42	196.44	77972	78088.30	88.19	198.62	77974	77987.36	7.80	340.90
rand7N100_5	100	3484	89646.00	89674.42	24.43	0.32	80414	80710.22	167.58	50.24	80283	80300.42	10.96	117.47	80276	80374.26	85.34	184.91	80277	80281.82	2.90	363.91
rand7N100_6	100	3426	87406.00	88052.18	278.36	0.50	78756	79112.30	209.15	56.44	78640	78668.90	12.24	127.59	78636	78786.98	119.79	194.78	78636	78646.24	5.02	350.04
rand7N100_7	100	3499	89233.00	89871.88	480.64	0.51	80917	81210.52	148.23	27.65	80842	80885.50	17.98	131.24	80833	80952.24	89.85	193.00	80836	80848.18	7.18	336.34
rand7N100_8	100	3464	88499.00	88874.16	307.43	0.71	79884	80203.80	182.40	47.11	79798	79825.04	14.34	167.94	79792	79913.14	89.32	193.26	79796	79803.90	3.71	335.71
rand7N100_9	100	3397	85915.00	85915.00	0.00	0.54	78047	78407.08	201.35	41.97	77976	78014.04	20.11	140.15	77972	78088.40	88.32	198.62	77974	77987.30	7.70	347.46
rand7N100_10	100	3484	89646.00	89671.48	24.73	0.32	80300	80680.36	184.66	87.25	80282	80300.26	12.01	116.91	80276	80374.42	85.31	184.89	80277	80281.82	2.90	363.96
rand7N11_1	11	43	125.00	127.36	1.65	0.01	108	108.00	0.00	0.01	108	108.00	0.00	0.01	108	108.00	0.00	0.01	108	108.00	0.00	0.01
rand7N11_2	11	34	82.00	88.96	3.61	0.01	81	81.00	0.00	0.01	81	81.00	0.00	0.01	81	81.00	0.00	0.01	81	81.00	0.00	0.01
rand7N12_1	12	43	124.00	131.12	4.54	0.01	116	116.06	0.24	0.11	116	116.00	0.00	0.01	116	116.00	0.00	0.01	116	116.00	0.00	0.02
rand7N12_2	12	38	114.00	114.00	0.00	0.01	98	98.00	0.00	0.09	98	98.00	0.00	0.01	98	98.00	0.00	0.01	98	98.00	0.00	0.02
rand9N10_1	10	38	100.00	100.00	0.00	0.01	95	95.00	0.00	0.01	95	95.00	0.00	0.01	95	95.00	0.00	0.01	95	95.00	0.00	0.01
rand9N10_2	10	37	91.00	91.00	0.00	0.01	89	89.00	0.00	0.01	89	89.00	0.00	0.01	89	89.00	0.00	0.01	89	89.00	0.00	0.01

Table B.14: Detailed results of Mach, MA, BVNS, MA-20, and DMAB+MA for random graphs (part three)

			Mach				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
rand9N10_3	10	42	116.00	116.00	0.00	0.01	112	112.00	0.00	0.01	112	112.00	0.00	0.01	112	112.00	0.00	0.01	112	112.00	0.00	0.01
rand9N10_4	10	42	115.00	115.00	0.00	0.01	111	111.00	0.00	0.01	111	111.00	0.00	0.01	111	111.00	0.00	0.01	111	111.00	0.00	0.01
rand9N10_5	10	43	120.00	120.00	0.00	0.01	116	116.00	0.00	0.01	116	116.00	0.00	0.01	116	116.00	0.00	0.01	116	116.00	0.00	0.01
rand9N10_6	10	37	102.00	102.56	0.91	0.01	90	90.00	0.00	0.01	90	90.00	0.00	0.01	90	90.00	0.00	0.01	90	90.00	0.00	0.01
rand9N10_7	10	41	114.00	114.00	0.00	0.01	107	107.00	0.00	0.01	107	107.00	0.00	0.01	107	107.00	0.00	0.01	107	107.00	0.00	0.01
rand9N10_8	10	39	106.00	106.32	0.47	0.01	102	102.00	0.00	0.01	102	102.00	0.00	0.01	102	102.00	0.00	0.01	102	102.00	0.00	0.01
rand9N10_9	10	43	118.00	118.00	0.00	0.01	115	115.00	0.00	0.01	115	115.00	0.00	0.01	115	115.00	0.00	0.01	115	115.00	0.00	0.01
rand9N10_10	10	35	94.00	95.04	1.01	0.01	86	86.00	0.00	0.01	86	86.00	0.00	0.01	86	86.00	0.00	0.01	86	86.00	0.00	0.01
rand9N100_1	100	4425	113535.00	114247.74	192.85	0.59	106633	106845.14	102.30	67.17	106569	106593.86	10.49	119.03	106557	106626.92	48.86	193.92	106561	106573.56	5.27	350.92
rand9N100_2	100	4452	113148.00	114133.36	368.78	0.71	107413	107685.04	119.97	73.09	107401	107433.60	16.73	145.34	107388	107483.36	77.28	214.67	107391	107408.12	6.32	344.40
rand9N100_3	100	4449	113479.00	114273.66	270.10	0.74	107345	107531.00	122.05	63.48	107243	107260.20	9.16	120.97	107230	107315.78	79.80	215.56	107237	107246.84	4.02	354.70
rand9N100_4	100	4415	112764.00	113410.20	380.85	0.72	106211	106472.56	160.42	85.78	106154	106173.26	9.68	94.95	106148	106259.28	93.66	184.70	106152	106158.74	2.78	383.29
rand9N100_5	100	4467	113931.00	114562.58	374.70	0.68	107731	107950.70	137.87	84.45	107699	107715.42	7.70	122.39	107690	107753.72	70.34	197.79	107696	107702.46	3.40	347.62
rand9N100_6	100	4425	113894.00	114244.96	175.67	0.62	106627	106818.70	116.08	74.33	106571	106595.20	13.08	149.49	106557	106626.92	48.86	193.93	106561	106573.56	5.27	350.98
rand9N100_7	100	4452	113614.00	114354.08	297.15	0.64	107467	107684.34	110.33	74.05	107397	107440.48	17.02	123.02	107388	107483.36	77.28	214.62	107391	107408.12	6.32	344.26
rand9N100_8	100	4449	113456.00	114374.24	351.38	0.72	107260	107461.32	124.83	97.39	107242	107258.04	8.68	117.38	107230	107315.78	79.87	215.69	107237	107246.84	4.02	354.76
rand9N100_9	100	4415	112738.00	113401.84	381.97	0.73	106211	106484.06	145.88	72.88	106156	106173.02	7.67	105.64	106148	106259.18	93.61	184.66	106152	106158.74	2.78	383.29
rand9N100_10	100	4467	113931.00	114623.96	361.32	0.66	107744	107959.60	127.99	60.70	107699	107715.68	7.91	127.44	107690	107753.72	70.34	197.71	107696	107702.46	3.40	347.81
rand9N11_1	11	50	145.00	145.00	0.00	0.01	141	141.00	0.00	0.01	141	141.00	0.00	0.01	141	141.00	0.00	0.01	141	141.00	0.00	0.01
rand9N11_2	11	51	156.00	156.00	0.00	0.01	145	145.00	0.00	0.01	145	145.00	0.00	0.01	145	145.00	0.00	0.01	145	145.00	0.00	0.01
rand9N12_1	12	56	177.00	177.00	0.00	0.01	166	166.00	0.00	0.01	166	166.00	0.00	0.01	166	166.00	0.00	0.01	166	166.00	0.00	0.01
rand9N12_2	12	56	178.00	181.78	3.52	0.01	166	166.00	0.00	0.01	166	166.00	0.00	0.01	166	166.00	0.00	0.01	166	166.00	0.00	0.02
rand9N15_1	15	93	363.00	370.00	10.60	0.01	347	347.00	0.00	0.01	347	347.00	0.00	0.01	347	347.00	0.00	0.01	347	347.00	0.00	0.02
rand9N15_2	15	94	374.00	374.66	1.53	0.01	351	351.00	0.00	0.02	351	351.00	0.00	0.01	351	351.00	0.00	0.01	351	351.00	0.00	0.01
rand9N15_3	15	93	370.00	372.96	2.94	0.01	344	344.00	0.00	0.48	344	344.00	0.00	0.01	344	344.00	0.00	0.02	344	344.00	0.00	0.02
rand9N15_4	15	93	368.00	371.60	3.59	0.01	349	349.00	0.00	0.01	349	349.00	0.00	0.01	349	349.00	0.00	0.01	349	349.00	0.00	0.01
rand9N15_5	15	89	343.00	350.10	6.22	0.01	320	320.00	0.00	0.01	320	320.00	0.00	0.01	320	320.00	0.00	0.01	320	320.00	0.00	0.01
rand9N15_6	15	96	394.00	395.24	0.62	0.01	357	357.00	0.00	0.01	357	357.00	0.00	0.01	357	357.00	0.00	0.01	357	357.00	0.00	0.01
rand9N15_7	15	93	370.00	373.20	3.04	0.01	344	344.00	0.00	0.54	344	344.00	0.00	0.01	344	344.00	0.00	0.02	344	344.00	0.00	0.02
rand9N15_8	15	93	365.00	365.00	0.00	0.01	344	344.00	0.00	0.04	344	344.00	0.00	0.01	344	344.00	0.00	0.01	344	344.00	0.00	0.03
rand9N15_9	15	98	401.00	401.36	0.48	0.01	372	372.00	0.00	0.01	372	372.00	0.00	0.01	372	372.00	0.00	0.01	372	372.00	0.00	0.01
rand9N15_10	15	95	376.00	379.28	5.63	0.01	352	352.00	0.00	0.01	352	352.00	0.00	0.01	352	352.00	0.00	0.01	352	352.00	0.00	0.01
rand9N20_1	20	172	905.00	918.04	8.11	0.02	836	836.36	0.66	7.69	836	836.00	0.00	0.03	836	836.00	0.00	0.06	836	836.00	0.00	0.10
rand9N20_2	20	171	900.00	922.44	10.53	0.01	835	835.02	0.14	5.67	835	835.00	0.00	0.02	835	835.00	0.00	0.02	835	835.00	0.00	2.11
rand9N20_3	20	175	926.00	933.86	6.34	0.01	867	867.00	0.00	0.09	867	867.00	0.00	0.02	867	867.00	0.00	0.02	867	867.00	0.00	0.08
rand9N20_4	20	175	949.00	949.00	0.00	0.01	861	861.00	0.00	2.80	861	861.00	0.00	0.01	861	861.00	0.00	0.03	861	861.00	0.00	0.05
rand9N20_5	20	164	856.00	870.70	13.14	0.02	788	788.00	0.00	11.55	788	788.00	0.00	0.03	788	788.00	0.00	0.03	788	788.00	0.00	0.10
rand9N20_6	20	173	908.00	933.86	16.71	0.02	840	840.18	0.44	17.66	840	840.00	0.00	0.11	840	840.00	0.00	0.05	840	840.00	0.00	0.14

Table B.15: Detailed results of MACH, MA, BVNS, MA-20, and DMAB+MA for random graphs (part four)

			MACH				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
rand9N20_7	20	166	897.00	903.30	6.69	0.01	788	788.40	1.05	13.18	788	788.00	0.00	0.05	788	788.00	0.00	0.02	788	788.00	0.00	0.52
rand9N20_8	20	169	897.00	898.74	1.50	0.02	827	827.00	0.00	2.09	827	827.00	0.00	0.09	827	827.00	0.00	0.04	827	827.00	0.00	0.43
rand9N20_9	20	170	900.00	900.00	0.00	0.01	831	831.00	0.00	1.20	831	831.00	0.00	0.04	831	831.00	0.00	0.02	831	831.00	0.00	0.23
rand9N20_10	20	166	897.00	905.92	6.40	0.01	788	788.38	0.73	22.67	788	788.00	0.00	0.04	788	788.00	0.00	0.02	788	788.00	0.00	0.52
rand9N30_1	30	394	3038.00	3151.78	37.71	0.01	2858	2862.74	4.03	53.91	2858	2858.08	0.27	200.27	2858	2858.00	0.00	1.35	2858	2858.00	0.00	11.62
rand9N30_2	30	387	3054.00	3068.44	13.18	0.02	2777	2782.32	3.94	26.31	2777	2777.00	0.00	32.25	2777	2777.00	0.00	1.87	2777	2777.00	0.00	4.80
rand9N30_3	30	391	3034.00	3078.90	26.16	0.02	2834	2838.60	3.76	52.51	2834	2834.00	0.00	37.85	2834	2834.18	0.72	18.18	2834	2834.00	0.00	10.17
rand9N30_4	30	398	3099.00	3148.58	22.13	0.02	2912	2913.74	2.37	62.04	2912	2912.00	0.00	85.79	2912	2912.00	0.00	3.62	2912	2912.00	0.00	249.63
rand9N30_5	30	387	3037.00	3089.70	26.58	0.02	2770	2771.98	3.22	24.09	2770	2770.00	0.00	3.33	2770	2770.00	0.00	0.85	2770	2770.00	0.00	3.40
rand9N30_6	30	388	3029.00	3070.74	30.77	0.02	2793	2799.72	4.50	41.73	2793	2793.00	0.00	21.84	2793	2793.00	0.00	6.03	2793	2793.00	0.00	43.38
rand9N30_7	30	389	3017.00	3058.58	40.02	0.02	2806	2810.06	3.57	32.35	2806	2806.00	0.00	57.78	2806	2806.00	0.00	8.58	2806	2806.00	0.00	4.16
rand9N30_8	30	391	3080.00	3118.56	26.29	0.02	2837	2841.36	4.22	50.27	2837	2837.00	0.00	6.01	2837	2837.00	0.00	0.62	2837	2837.00	0.00	6.85
rand9N30_9	30	392	3053.00	3138.82	34.74	0.01	2824	2827.66	5.58	40.18	2824	2824.00	0.00	2.00	2824	2824.00	0.00	0.94	2824	2824.00	0.00	3.70
rand9N30_10	30	396	3098.00	3129.38	22.03	0.02	2863	2867.22	3.59	51.46	2863	2863.00	0.00	28.55	2863	2863.24	0.82	8.92	2863	2863.00	0.00	7.83
rand9N40_1	40	701	7159.00	7296.66	52.31	0.03	6710	6722.20	11.16	41.41	6710	6710.34	0.48	169.93	6710	6710.00	0.00	10.79	6710	6710.00	0.00	48.30
rand9N40_2	40	696	7088.00	7245.20	64.55	0.04	6673	6687.48	13.03	76.74	6673	6673.00	0.00	84.39	6673	6673.00	0.00	7.64	6673	6673.00	0.00	37.90
rand9N40_3	40	701	7218.00	7325.68	54.49	0.05	6734	6742.30	6.84	49.55	6733	6733.96	0.73	221.63	6733	6733.00	0.00	24.16	6733	6733.00	0.00	146.03
rand9N40_4	40	699	7091.00	7314.64	58.49	0.05	6706	6719.38	8.41	43.20	6703	6704.12	0.66	172.84	6703	6703.04	0.20	40.88	6703	6703.00	0.00	290.52
rand9N40_5	40	700	7132.00	7297.84	58.82	0.04	6711	6730.44	11.38	67.48	6711	6711.00	0.00	70.54	6711	6711.62	2.16	23.26	6711	6711.00	0.00	105.13
rand9N40_6	40	696	7114.00	7282.42	55.26	0.04	6673	6688.20	13.13	75.63	6673	6673.02	0.14	89.55	6673	6673.00	0.00	7.65	6673	6673.00	0.00	37.89
rand9N40_7	40	721	7431.00	7539.68	59.20	0.04	6994	7008.70	8.44	62.33	6994	6995.44	0.64	216.98	6994	6994.00	0.00	25.29	6994	6994.00	0.00	319.70
rand9N40_8	40	687	7149.00	7178.42	24.35	0.05	6536	6554.14	13.45	47.25	6535	6535.44	0.50	125.04	6535	6535.00	0.00	4.21	6535	6535.00	0.00	64.64
rand9N40_9	40	701	7159.00	7289.48	47.86	0.03	6710	6722.98	11.79	44.24	6710	6710.38	0.49	138.89	6710	6710.00	0.00	10.78	6710	6710.00	0.00	48.30
rand9N40_10	40	701	7290.00	7352.76	46.81	0.03	6714	6728.90	10.40	57.76	6713	6714.80	0.95	242.86	6713	6714.66	3.23	42.19	6713	6713.00	0.00	199.73

Table B.16: Detailed results of MACH, MA, BVNS, MA-20, and DMAB+MA for Standard graphs (part one)

			MACH				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
bipartiteComplete50-50	100	2500	62500.00	62500.00	0.00	0.17	62500	62500.00	0.00	0.03	62500	62500.00	0.00	0.02	62500	62500.00	0.00	0.01	62500	62500.00	0.00	305.07
bipartiteComplete60-60	120	3600	108000.00	108000.00	0.00	0.32	108000	108000.00	0.00	0.05	108000	108000.00	0.00	0.05	108000	108000.00	0.00	0.02	108000	108000.00	0.00	347.00
bipartiteComplete70-70	140	4900	171500.00	171500.00	0.00	0.55	171500	171500.00	0.00	0.09	171500	171500.00	0.00	0.08	171500	171500.00	0.00	0.05	171500	171500.00	0.00	306.09
bipartiteComplete80-80	160	6400	256000.00	256000.00	0.00	0.89	256000	256000.00	0.00	0.17	256000	256000.00	0.00	0.13	256000	256000.00	0.00	0.08	256000	256000.40	0.81	197.96
bipartiteComplete90-90	180	8100	364500.00	364500.00	0.00	1.40	364500	364500.00	0.00	0.24	364500	364500.00	0.00	0.19	364500	364500.00	0.00	0.13	364500	364502.56	1.34	119.47
bipartiteComplete200-200	400	40000	4000000.00	4000000.00	0.00	30.15	4000000	4000000.00	0.00	3.62	4000000	4000000.00	0.00	2.58	4000000	4000000.00	0.00	2.09	4000050	4000060.04	4.19	245.09
bipartiteComplete300-300	600	90000	13500000.00	13500000.00	0.00	148.75	13500000	13500000.00	0.00	18.13	13500000	13500000.00	0.00	9.67	13500000	13500000.00	0.00	7.76	13500118	13500139.92	7.28	337.12
bipartiteComplete400-400	800	160000	32000000.00	32000000.00	0.00	464.36	32000000	32000000.00	0.00	51.38	32000000	32000000.00	0.00	23.28	32000000	32000000.00	0.00	20.89	32000218	32000241.08	10.87	267.46
caterpillar3	9	8	13.00	13.66	0.48	0.01	12	12.00	0.00	0.01	12	12.00	0.00	0.01	12	12.00	0.00	0.01	12	12.00	0.00	0.01
caterpillar4	14	13	26.00	27.02	0.87	0.01	23	23.00	0.00	0.01	23	23.00	0.00	0.01	23	23.00	0.00	0.01	23	23.00	0.00	0.01
caterpillar5	20	19	44.00	47.16	1.40	0.01	37	37.00	0.00	0.55	37	37.00	0.00	0.01	37	37.00	0.00	0.01	37	37.00	0.00	0.02
caterpillar6	27	26	71.00	73.44	2.54	0.01	56	56.00	0.00	0.65	56	56.00	0.00	0.01	56	56.00	0.00	0.02	56	56.00	0.00	0.10
caterpillar7	35	34	102.00	109.10	3.83	0.01	79	79.00	0.00	0.93	79	79.00	0.00	0.02	79	79.00	0.00	0.03	79	79.00	0.00	0.30
caterpillar13	104	103	518.00	550.62	23.88	0.03	348	364.44	11.36	74.50	347	347.00	0.00	41.52	347	348.62	4.99	11.90	347	347.46	2.30	147.00

Table B.17: Detailed results of MACH, MA, BVNS, MA-20, and DMAB+MA for Standard graphs (part two)

			MACH				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
caterpillar14	119	118	639.00	671.16	27.29	0.05	428	465.86	21.26	67.50	418	418.04	0.20	115.90	418	428.88	20.89	15.42	418	420.34	4.88	258.41
caterpillar16	152	151	917.00	966.78	43.08	0.10	669	747.64	39.11	78.46	586	602.72	14.61	206.26	586	639.74	35.94	19.63	586	616.48	17.70	304.72
caterpillar17	170	169	1090.00	1140.74	44.28	0.13	817	916.70	52.69	72.59	684	725.40	20.64	149.33	684	782.42	44.77	19.00	701	761.54	25.35	318.23
caterpillar19	209	208	1465.00	1535.90	55.68	0.22	1249	1414.82	70.93	64.76	912	1013.62	38.50	171.76	959	1096.44	53.60	21.47	962	1049.60	33.14	310.49
caterpillar23	299	298	2520.00	2610.84	88.26	0.66	2556	2919.36	193.60	75.41	1543	1799.46	81.79	151.50	1803	2080.46	124.65	20.95	1675	1870.46	60.03	320.33
caterpillar29	464	463	4854.00	4964.64	110.15	2.26	6387	7206.64	452.09	71.96	3041	3474.14	171.21	173.91	3738	4331.48	246.92	20.72	3489	3782.10	124.25	276.62
caterpillar35	665	664	8264.00	8434.68	167.55	6.78	12428	14744.42	969.84	78.61	5607	6080.50	289.20	358.37	6855	7571.28	357.44	31.12	6564	7155.64	203.81	298.73
caterpillar39	819	818	11320.00	11468.90	160.71	11.28	19061	22287.66	1469.85	74.80	7562	8141.98	282.82	266.03	9043	10230.44	538.07	52.94	9159	9726.04	282.02	282.75
cycle10	10	10	10.00	10.00	0.00	0.01	10	10.00	0.00	0.01	10	10.00	0.00	0.01	10	10.00	0.00	0.01	10	10.00	0.00	0.01
cycle11	11	11	11.00	11.00	0.00	0.01	11	11.00	0.00	0.01	11	11.00	0.00	0.01	11	11.00	0.00	0.01	11	11.00	0.00	0.01
cycle12	12	12	12.00	12.00	0.00	0.01	12	12.00	0.00	0.01	12	12.00	0.00	0.01	12	12.00	0.00	0.01	12	12.00	0.00	0.01
cycle20	20	20	20.00	20.00	0.00	0.01	20	20.00	0.00	0.04	20	20.00	0.00	0.01	20	20.00	0.00	0.01	20	20.00	0.00	0.06
cycle25	25	25	25.00	25.00	0.00	0.01	25	25.00	0.00	0.34	25	25.00	0.00	0.01	25	25.00	0.00	0.04	25	25.00	0.00	0.19
cycle30	30	30	30.00	30.00	0.00	0.01	30	30.00	0.00	1.89	30	30.00	0.00	0.01	30	30.00	0.00	0.30	30	30.00	0.00	0.19
cycle35	35	35	35.00	35.00	0.00	0.01	35	35.70	4.95	5.52	35	35.00	0.00	0.02	35	35.00	0.00	1.14	35	35.00	0.00	0.34
cycle40	40	40	40.00	40.00	0.00	0.01	40	48.48	16.13	7.04	40	40.00	0.00	0.08	40	40.00	0.00	3.41	40	40.00	0.00	0.65
cycle100	100	100	100.00	100.00	0.00	0.01	100	100.00	0.00	2.33	122	169.24	21.90	219.65	100	161.80	57.65	5.15	100	100.00	0.00	14.74
cycle120	120	120	120.00	120.00	0.00	0.01	120	120.00	0.00	4.45	186	240.04	18.65	74.42	120	208.60	54.43	13.11	120	120.00	0.00	32.53
cycle125	125	125	125.00	125.00	0.00	0.01	125	173.16	60.16	17.99	187	256.34	22.90	95.24	125	231.04	66.06	13.51	125	125.00	0.00	47.54
cycle140	140	140	140.00	140.00	0.00	0.01	140	142.80	19.80	9.05	248	301.20	21.97	14.39	160	276.28	57.31	15.09	140	140.00	0.00	33.71
cycle150	150	150	150.00	150.00	0.00	0.01	150	320.12	134.90	61.33	298	346.04	36.66	38.91	220	306.56	45.20	17.91	150	151.44	7.83	137.85
cycle160	160	160	160.00	160.00	0.00	0.01	160	172.20	36.09	13.40	288	346.52	26.47	0.04	250	365.12	79.89	18.46	160	160.00	0.00	33.77
cycle180	180	180	180.00	180.00	0.00	0.01	180	234.40	80.16	15.14	336	397.08	27.23	0.05	328	437.64	80.21	22.97	180	180.00	0.00	60.50
cycle200	200	200	200.00	200.00	0.00	0.02	200	307.32	108.81	20.65	374	438.64	34.37	0.06	376	523.32	107.79	27.05	200	200.00	0.00	54.27
cycle300	300	300	300.00	300.00	0.00	0.02	3098	3646.92	260.21	75.10	666	898.00	130.27	3.96	866	1093.64	180.87	44.06	598	749.12	71.39	316.41
cycle400	400	400	400.00	400.00	0.00	0.01	400	1146.64	416.05	97.27	798	884.84	68.44	0.17	1554	2008.88	283.48	81.98	400	407.96	56.29	307.18
cycle475	475	475	475.00	475.00	0.00	0.02	8964	10299.26	606.84	72.04	1160	1579.86	235.81	23.26	2496	2965.34	360.87	115.15	2014	2463.14	221.86	310.67
cycle600	600	600	600.00	600.00	0.00	0.02	842	2464.00	895.21	126.77	1198	1344.16	97.02	0.28	4452	5236.52	296.16	148.22	600	717.92	134.36	523.31
cycle800	800	800	800.00	800.00	0.00	0.03	1638	4349.80	1213.59	129.64	1598	1788.60	140.16	0.45	8658	10541.68	581.70	192.66	860	1171.04	144.10	572.38
cycle1000	1000	1000	1000.00	1000.00	0.00	0.05	3180	6498.20	1858.53	132.89	1998	2261.04	149.74	0.68	15874	18048.84	806.18	228.02	1142	1713.96	285.19	570.80
cyclePow10-2	10	20	30.00	32.72	2.31	0.01	30	30.00	0.00	0.01	30	30.00	0.00	0.01	30	30.00	0.00	0.01	30	30.00	0.00	0.01
cyclePow11-2	11	22	33.00	35.72	2.31	0.01	33	33.00	0.00	0.01	33	33.00	0.00	0.01	33	33.00	0.00	0.01	33	33.00	0.00	0.01
cyclePow12-2	12	24	36.00	38.56	2.10	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01	36	36.00	0.00	0.01
cyclePow100-10	100	1000	5562.00	5710.44	71.76	0.03	5500	5500.00	0.00	0.20	5500	5500.00	0.00	0.42	5500	5783.80	1134.72	11.10	5500	5500.00	0.00	0.38
cyclePow100-2	100	200	300.00	303.12	2.29	0.01	300	300.00	0.00	1.88	300	300.00	0.00	3.02	300	458.00	159.32	2.49	300	300.00	0.00	24.84
cyclePow120-10	120	1200	6668.00	6781.04	66.49	0.04	6600	6600.00	0.00	0.96	6600	6600.00	0.00	2.29	6600	8146.52	2639.44	4.35	6600	6600.00	0.00	0.78
cyclePow120-2	120	240	360.00	362.00	1.85	0.02	360	360.00	0.00	3.85	360	360.00	0.00	17.10	360	536.80	178.63	6.35	360	360.00	0.00	38.53
cyclePow140-10	140	1400	7762.00	7897.80	58.46	0.06	7700	7700.00	0.00	2.80	7700	7700.00	0.00	5.06	7700	11226.60	3568.11	0.39	7700	7700.00	0.00	1.49

Table B.18: Detailed results of MACH, MA, BVNS, MA-20, and DMAB+MA for Standard graphs (part three)

			MACH				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
cyclePow140-2	140	280	420.00	422.96	2.43	0.02	420	420.00	0.00	9.33	420	427.68	31.63	61.45	420	635.64	255.78	11.62	420	420.00	0.00	60.22
cyclePow160-10	160	1600	8866.00	9004.28	62.20	0.07	8800	8800.00	0.00	6.34	8800	8800.00	0.00	9.60	8800	11882.20	3979.69	0.77	8800	8800.00	0.00	1.67
cyclePow160-2	160	320	480.00	482.12	2.15	0.02	480	495.28	78.17	11.95	480	567.56	143.86	137.50	480	817.12	306.13	18.20	480	480.00	0.00	61.26
cyclePow180-10	180	1800	9938.00	10121.64	73.38	0.08	9900	10082.60	1291.18	9.16	9900	9900.00	0.00	20.32	9900	13994.20	4670.78	2.12	9900	9900.00	0.00	3.06
cyclePow180-2	180	360	540.00	542.32	2.15	0.01	540	652.64	207.56	17.77	540	751.28	203.88	106.25	540	979.76	326.73	27.16	540	540.00	0.00	39.31
cyclePow200-10	200	2000	11052.00	11194.92	64.71	0.10	11000	11143.00	1011.16	15.34	11000	11000.00	0.00	43.57	11000	15767.40	5221.62	3.28	11000	11000.00	0.00	6.01
cyclePow200-2	200	400	600.00	603.04	2.15	0.01	600	820.56	277.47	22.60	600	1014.20	195.27	56.92	600	1132.80	308.70	27.59	600	600.00	0.00	63.37
cyclePow400-10	400	4000	22062.00	22193.32	69.95	0.31	22000	43369.32	11873.01	89.08	22000	35743.40	9125.26	222.49	23296	34858.20	13154.39	169.12	22000	22000.00	0.00	80.41
cyclePow400-2	400	800	1200.00	1202.64	2.15	0.03	1200	3516.20	1369.06	103.82	2138	2522.84	190.82	10.85	3622	4279.84	817.06	122.02	1200	1200.00	0.00	135.92
cyclePow600-10	600	6000	33082.00	33211.52	67.78	0.64	50050	88245.88	17049.09	151.77	53130	63653.04	5494.55	434.19	51274	81616.76	20049.86	229.13	33000	34427.84	5184.53	289.44
cyclePow600-2	600	1200	1800.00	1802.56	2.21	0.06	2978	7222.32	2121.76	127.95	3418	3924.96	311.63	41.07	10432	11594.64	775.18	164.50	1800	1948.00	321.84	373.25
cyclePow800-10	800	8000	44072.00	44187.68	60.78	1.09	100980	157741.88	28272.27	161.96	78210	92600.28	7308.85	462.91	95976	140205.40	18333.29	236.38	44000	54896.24	12812.27	446.20
cyclePow800-2	800	1600	2400.00	2402.56	2.43	0.09	7286	12790.36	3161.43	134.35	4618	5206.28	396.26	43.88	20198	23015.12	1802.37	233.61	2400	3134.76	712.36	539.48
cyclePow1000-2	1000	2000	3000.00	3002.56	2.32	0.13	9832	18412.12	3906.75	137.31	5762	6600.28	462.18	127.77	34592	39649.08	2333.06	233.97	3000	4095.28	770.83	557.57
mesh2D5x4	20	31	87.00	96.52	8.39	0.01	63	63.64	3.17	0.07	63	63.00	0.00	0.01	63	63.00	0.00	0.01	63	63.00	0.00	0.01
mesh2D5x5	25	40	133.00	152.64	18.08	0.01	90	90.88	4.35	0.43	90	90.00	0.00	0.01	90	90.00	0.00	0.01	90	90.00	0.00	0.02
mesh2D5x6	30	49	187.00	222.20	27.57	0.01	117	120.22	9.76	6.92	117	117.00	0.00	0.01	117	117.00	0.00	0.02	117	117.00	0.00	0.03
mesh2D5x7	35	58	242.00	297.78	40.51	0.01	149	158.44	13.92	7.18	149	149.00	0.00	0.03	149	149.00	0.00	0.27	149	149.00	0.00	0.16
mesh2D5x8	40	67	303.00	373.60	57.24	0.01	181	198.56	19.13	15.35	181	181.00	0.00	0.15	181	188.48	10.53	0.96	181	181.00	0.00	1.36
mesh3D4x4x4	64	144	1022.00	1236.12	104.45	0.01	752	779.14	34.26	34.42	752	752.00	0.00	0.87	752	752.00	0.00	0.24	752	752.00	0.00	295.64
mesh2D10x10	100	180	1322.00	1846.00	202.29	0.01	724	946.70	116.13	21.64	702	704.14	1.14	202.31	700	705.68	30.79	24.69	700	700.00	0.00	292.77
mesh2D5x25	125	220	2625.00	3301.80	562.63	0.01	700	1108.72	224.69	38.22	696	696.96	1.05	167.96	696	696.00	0.00	8.69	696	696.00	0.00	193.02
mesh3D5x5x5	125	300	3989.00	4521.74	374.22	0.04	2278	2392.80	141.63	67.68	2262	2267.02	2.93	210.76	2262	2267.06	20.15	15.17	2262	2262.00	0.00	276.24
mesh2D10x15	150	275	2811.00	3671.24	405.50	0.02	1340	1631.20	195.66	24.05	1303	1311.22	4.48	164.30	1294	1367.78	86.68	27.89	1289	1292.06	1.08	314.67
mesh2D7x25	175	318	4035.00	5357.28	935.75	0.02	1312	1727.82	339.96	14.17	1320	1332.10	5.39	152.44	1310	1322.40	48.82	32.68	1308	1308.00	0.00	236.57
mesh2D8x25	200	367	4825.00	6687.92	1002.67	0.01	1681	2076.66	312.34	15.42	1703	1720.48	8.31	80.40	1689	1741.60	145.94	33.75	1679	1679.00	0.00	312.45
mesh3D6x6x6	216	540	10334.00	12570.18	868.04	0.11	5536	5719.28	171.54	46.65	5505	5541.26	15.11	107.45	5500	5583.20	128.96	53.79	5492	5492.56	1.21	307.35
mesh2D15x20	300	565	9231.00	11777.04	675.23	0.02	4785	5952.72	566.86	66.68	3688	3727.28	16.03	141.54	3692	3907.92	318.61	70.75	3630	3646.66	5.82	359.20
mesh3D7x7x7	343	882	23981.00	29389.80	2247.46	0.70	12772	14474.92	598.65	77.39	11956	12033.80	39.74	149.00	12167	12424.42	84.77	83.57	11893	11934.38	16.60	307.77
mesh2D19x25	475	906	21014.00	24710.16	888.98	0.04	15322	18639.18	1513.37	58.52	7368	8659.38	781.03	377.02	8120	8787.50	674.22	113.80	7260	7330.62	51.43	336.97
mesh3D8x8x8	512	1344	54672.00	62886.90	4180.46	2.48	30485	33523.00	1468.60	64.31	23019	24359.60	1353.08	474.86	24831	25772.88	358.20	129.36	22967	23100.26	98.90	272.20
mesh2D25x26	650	1249	37183.00	39980.28	829.16	0.06	31549	37348.14	2915.46	70.15	11847	15137.20	2380.37	478.24	14892	16736.58	1608.96	175.81	11659	12439.40	631.52	324.78
mesh3D9x9x9	729	1944	103291.00	120571.80	7408.10	8.89	62594	69234.34	3273.33	85.48	43501	51506.52	3973.66	585.10	48390	50003.56	703.87	182.80	41389	43033.48	701.29	338.62
mesh2D28x30	840	1622	58112.00	59516.08	460.91	0.09	55386	66676.44	5705.90	76.42	21315	25847.70	1638.99	577.29	24115	28728.26	2523.78	226.83	17814	22099.08	2483.86	329.27
mesh2D20x50	1000	1930	60698.00	92680.68	5911.45	0.12	76549	94027.82	6467.35	96.35	21635	30258.36	4965.33	597.73	35958	40996.12	3238.38	237.62	22617	29792.30	3307.89	360.34
mesh3D10x10x10	1000	2700	168085.00	216353.50	14673.77	22.09	119167	132531.82	7518.42	78.56	84969	102090.38	7745.08	605.30	88595	91528.26	2890.92	200.62	70940	76059.02	1753.78	325.72
mesh3D11x11x11	1331	3630	317459.00	362225.38	20953.53	79.57	211115	240252.88	13476.41	91.87	157673	179189.80	13487.00	603.51	153138	161363.38	5645.58	228.48	117888	129196.12	4927.68	301.71
mesh3D12x12x12	1728	4752	516960.00	591890.22	34160.33	199.09	359879	409358.34	24669.89	99.10	251473	299958.18	19521.73	632.38	256217	283325.88	18172.57	253.22	193910	213781.22	8752.56	381.02

Table B.19: Detailed results of Mach, MA, BVNS, MA-20, and DMAB+MA for Standard graphs (part four)

			Mach				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
mesh3D13x13x13	2197	6084	815289.00	921799.02	53100.78	436.53	558891	658646.16	46605.64	106.07	425435	471642.18	23238.03	685.05	420354	488684.00	31874.50	259.23	285791	337104.00	21362.38	394.28
path10	10	9	9.00	9.00	0.00	0.01	9	9.00	0.00	0.01	9	9.00	0.00	0.01	9	9.00	0.00	0.01	9	9.00	0.00	0.01
path11	11	10	10.00	10.00	0.00	0.01	10	10.00	0.00	0.01	10	10.00	0.00	0.01	10	10.00	0.00	0.01	10	10.00	0.00	0.01
path12	12	11	11.00	11.00	0.00	0.01	11	11.00	0.00	0.01	11	11.00	0.00	0.01	11	11.00	0.00	0.01	11	11.00	0.00	0.01
path15	15	14	14.00	14.00	0.00	0.01	14	14.00	0.00	0.01	14	14.00	0.00	0.01	14	14.00	0.00	0.01	14	14.00	0.00	0.01
path20	20	19	19.00	19.00	0.00	0.01	19	19.00	0.00	0.01	19	19.00	0.00	0.01	19	19.00	0.00	0.01	19	19.00	0.00	0.02
path25	25	24	24.00	24.00	0.00	0.01	24	24.00	0.00	0.16	24	24.00	0.00	0.01	24	24.00	0.00	0.02	24	24.00	0.00	0.04
path30	30	29	29.00	29.00	0.00	0.01	29	29.00	0.00	0.29	29	29.00	0.00	0.04	29	29.00	0.00	0.07	29	29.00	0.00	0.09
path35	35	34	34.00	34.00	0.00	0.01	34	34.00	0.00	0.70	34	34.00	0.00	0.12	34	34.00	0.00	0.17	34	34.00	0.00	0.23
path40	40	39	39.00	39.00	0.00	0.01	39	39.00	0.00	1.25	39	39.00	0.00	0.48	39	39.00	0.00	0.32	39	39.00	0.00	0.37
path100	100	99	99.00	99.00	0.00	0.01	99	104.10	12.48	8.92	99	152.62	26.07	91.82	99	128.48	23.95	17.80	99	99.00	0.00	10.03
path120	120	119	119.00	119.00	0.00	0.01	119	131.92	18.30	18.81	119	192.76	40.10	0.04	131	179.98	29.77	18.75	119	119.00	0.00	17.36
path125	125	124	124.00	124.00	0.00	0.01	124	170.48	43.70	55.81	153	237.82	28.68	77.28	140	202.12	27.79	17.91	124	124.46	3.25	134.57
path140	140	139	139.00	139.00	0.00	0.01	139	164.08	28.42	27.30	139	215.98	38.25	0.01	188	264.42	45.11	20.64	139	139.00	0.00	27.95
path150	150	149	149.00	149.00	0.00	0.01	149	260.28	97.75	84.65	217	313.84	44.72	43.18	228	284.64	41.83	22.85	149	163.86	22.36	326.59
path160	160	159	159.00	159.00	0.00	0.01	159	210.40	46.71	16.58	159	255.54	47.20	0.01	201	325.24	58.31	22.50	159	159.00	0.00	47.59
path175	175	174	174.00	174.00	0.00	0.02	435	747.98	133.57	71.07	302	395.64	53.68	9.90	299	381.68	60.70	23.42	174	229.64	36.17	321.49
path180	180	179	179.00	179.00	0.00	0.01	179	243.86	52.98	19.69	185	291.78	52.92	0.01	313	393.38	65.62	24.58	179	179.00	0.00	56.96
path200	200	199	199.00	199.00	0.00	0.02	199	301.60	82.49	40.77	199	323.72	58.59	0.03	404	511.74	76.73	25.15	199	199.00	0.00	85.07
path300	300	299	299.00	299.00	0.00	0.02	2571	3564.34	261.75	75.39	580	770.80	104.96	4.12	871	1093.16	211.05	46.82	562	708.86	71.01	344.41
path400	400	399	399.00	399.00	0.00	0.01	399	1104.28	461.74	116.18	413	664.62	115.74	0.04	1476	1945.86	298.83	81.61	399	463.42	86.96	379.78
path475	475	474	474.00	474.00	0.00	0.02	8655	10301.72	578.72	71.52	1175	1562.54	242.19	23.38	2467	2893.52	202.52	112.14	1628	2393.48	275.00	296.59
path600	600	599	599.00	599.00	0.00	0.02	807	2490.06	807.81	125.51	621	936.58	174.45	0.20	4354	5136.34	302.61	143.90	599	846.70	166.89	528.05
path800	800	799	799.00	799.00	0.00	0.03	1902	4205.98	1185.44	129.13	865	1289.72	222.74	0.29	9004	10250.58	553.28	201.33	888	1249.02	230.96	548.15
path1000	1000	999	999.00	999.00	0.00	0.05	3195	6271.26	1709.62	132.89	1057	1622.10	257.87	0.31	16239	17749.22	752.76	233.36	1231	1745.14	317.77	559.07
tree2x4	31	30	60.00	60.00	0.00	0.02	60	60.00	0.00	0.71	60	60.00	0.00	0.02	60	60.00	0.00	0.02	60	60.00	0.00	0.18
tree3x3	40	39	97.00	97.42	0.50	0.01	91	92.78	3.60	15.93	91	91.00	0.00	0.01	91	91.00	0.00	0.05	91	91.00	0.00	0.17
tree10x2	111	110	748.00	748.00	0.00	0.09	580	581.02	1.58	29.97	580	580.00	0.00	0.01	580	580.00	0.00	0.02	580	580.00	0.00	0.12
tree3x4	121	120	370.00	370.52	0.50	0.17	350	373.66	16.11	54.56	350	358.56	6.50	152.99	350	351.98	5.69	12.28	350	350.00	0.00	64.04
tree5x3	156	155	700.00	718.96	17.92	0.29	604	673.88	33.83	59.32	599	605.64	10.06	94.60	599	604.96	10.77	17.03	599	603.36	8.41	243.75
tree13x2	183	182	1634.00	1634.00	0.00	0.35	1264	1318.94	31.55	82.66	1232	1232.00	0.00	0.02	1232	1232.00	0.00	0.04	1232	1232.00	0.00	0.79
tree2x7	255	254	752.00	823.60	53.33	1.30	1530	2174.36	207.06	58.05	913	1069.66	72.34	5.50	810	885.58	34.58	32.06	799	914.54	38.59	325.11
tree17x2	307	306	3650.00	3650.00	0.00	1.45	3091	3403.36	178.17	66.25	2682	2682.00	0.00	0.09	2682	2682.00	0.00	0.15	2682	2682.00	0.00	4.82
tree21x2	463	462	6882.00	6882.00	0.00	4.66	6584	7429.24	461.92	71.68	4972	4972.00	0.00	0.25	4972	4972.00	0.00	0.46	4972	4972.00	0.00	25.41
tree25x2	651	650	11618.00	11618.00	0.00	12.51	12678	14571.76	1015.24	79.04	8294	8294.00	0.00	0.51	8294	8294.00	0.00	1.53	8294	8310.84	13.51	311.14
tree5x4	781	780	4293.00	4330.86	81.14	27.88	21812	25050.02	1626.75	58.63	8586	10837.32	982.64	595.74	7933	8925.84	420.21	171.46	7767	8779.52	485.34	294.06
wheel10	10	18	35.00	35.00	0.00	0.01	35	35.00	0.00	0.01	35	35.00	0.00	0.01	35	35.00	0.00	0.01	35	35.00	0.00	0.01
wheel11	11	20	41.00	41.00	0.00	0.01	41	41.00	0.00	0.01	41	41.00	0.00	0.01	41	41.00	0.00	0.01	41	41.00	0.00	0.01

Table B.20: Detailed results of MACH, MA, BVNS, MA-20, and DMAB+MA for Standard graphs (part five)

			MACH				MA				BVNS				MA-20				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*	*Best*	*Avg*	*Std*	*T*
wheel12	12	22	48.00	48.00	0.00	0.01	48	48.00	0.00	0.01	48	48.00	0.00	0.01	48	48.00	0.00	0.01	48	48.00	0.00	0.01
wheel15	15	28	71.00	71.00	0.00	0.01	71	71.00	0.00	0.01	71	71.00	0.00	0.01	71	71.00	0.00	0.01	71	71.00	0.00	0.01
wheel20	20	38	120.00	120.00	0.00	0.01	120	120.00	0.00	0.01	120	120.00	0.00	0.01	120	120.00	0.00	0.02	120	120.00	0.00	0.02
wheel40	40	78	440.00	440.00	0.00	0.01	440	440.00	0.00	0.09	440	440.00	0.00	0.27	440	440.72	5.09	2.47	440	440.00	0.00	0.60
wheel100	100	198	2600.00	2600.00	0.00	0.01	2600	2600.00	0.00	3.14	2628	2689.68	23.35	179.38	2600	2652.72	47.13	9.57	2600	2600.00	0.00	2.56
wheel120	120	238	3720.00	3720.00	0.00	0.02	3720	3724.80	23.75	5.32	3812	3870.36	20.87	145.32	3720	3797.52	54.04	22.15	3720	3720.00	0.00	3.05
wheel140	140	278	5040.00	5040.00	0.00	0.03	5040	5043.48	20.28	12.50	5154	5243.60	37.20	139.76	5062	5183.40	63.91	24.72	5040	5040.00	0.00	4.45
wheel160	160	318	6560.00	6560.00	0.00	0.04	6560	6582.88	51.08	15.68	6710	6832.44	51.65	121.85	6650	6767.40	76.61	29.51	6560	6560.00	0.00	6.51
wheel180	180	358	8280.00	8280.00	0.00	0.05	8280	8351.88	84.83	19.61	8522	8649.04	61.71	112.18	8406	8528.92	89.08	34.10	8280	8280.00	0.00	8.74
wheel200	200	398	10200.00	10200.00	0.00	0.07	10200	10282.12	93.22	27.37	10520	10672.36	70.52	91.10	10398	10539.76	105.53	38.31	10200	10200.00	0.00	13.45
wheel400	400	798	40400.00	40400.00	0.00	0.49	40400	41184.04	439.27	100.97	41360	42015.40	275.31	36.02	41630	41939.00	179.04	100.45	40400	40400.00	0.00	71.21
wheel600	600	1198	90600.00	90600.00	0.00	1.61	90952	92570.16	818.74	130.31	92722	93888.44	591.17	126.86	94668	95187.88	259.98	187.68	90600	90600.00	0.00	124.01
wheel800	800	1598	160800.00	160800.00	0.00	3.72	161962	164241.36	1261.65	133.74	164540	166256.88	744.55	304.64	169164	170340.44	540.48	230.65	160800	160801.96	9.50	275.30
wheel1000	1000	1998	251000.00	251000.00	0.00	7.27	252896	256614.00	1981.52	136.81	257820	259832.80	913.22	491.77	266086	267913.16	752.61	230.48	251000	251172.44	199.78	400.79

C

List of publications

Bibliography

[1] Albrecht, A. A., Lane, P. C., and Steinhofel, K. (2008). Combinatorial landscape analysis for k-sat instances. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 2498–2504.

[2] Arora, S. and Barak, B. (2009). *Computational complexity: a modern approach*. Cambridge University Press.

[3] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256.

[4] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (2012a). *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media.

[5] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (2012b). *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media.

[6] B., Y. and J., W. (1995). On bandwidth sums of graphs. *Acta Mathematicae Applicatae Sinica*, 11(1):69–78.

[7] Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

[Beldiceanu] Beldiceanu, N. Global constraint catalog: A dictionary for constraint programming. Technical report.

[9] Belluz, J., Gaudesi, M., Squillero, G., and Tonda, A. (2015). Operator selection using improved dynamic multi-armed bandit. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 1311–1317, New York, NY, USA. ACM.

[10] Bhatt, S. N. and Leighton, F. T. (1984). A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343.

[11] Biazzini, M., Banhelyi, B., Montresor, A., and Jelasity, M. (2009). Distributed hyper-heuristics for real parameter optimization. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 1339–1346, New York, NY, USA. ACM.

[12] Bloom, G. S. and Golomb, S. W. (1978). Numbered complete graphs, unusual rulers, and assorted applications. In Alavi, Y. and Lick, D. R., editors, *Theory and Applications of Graphs: Proceedings, Michigan, US, May 11–15, 1976*, volume 642 of *Lecture Notes in Mathematics*, pages 53–65. Springer, Berlin, Heidelberg.

[13] Blum, C. and Ochoa, G. (2021). A comparative analysis of two matheuristics by means of merged local optima networks. *European Journal of Operational Research*, 290:36 – 56.

[14] Blum, C., Puchinger, J., Raidl, G. R., and Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135 – 4151.

[15] Blum, C. and Raidl, G. R. (2016). *Hybrid Metaheuristics: Powerful Tools for Optimization*. Springer.

[16] Boussaïd, I., Lepagnot, J., and Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237:82 – 117. Prediction, Control and Diagnosis using Advanced Neural Computations.

[17] Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013).

Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.

[18] Chen, Y. and Yan, J. (2007). A study on cyclic bandwidth sum. *Journal of Combinatorial Optimization*, 14(2-3):295–308.

[19] Chopard, B. and Tomassini, M. (2018). *Performance and Limitations of Metaheuristics*, pages 191–203. Springer International Publishing, Cham.

[20] Chung, F. R. K. (1988). Labelings of graphs. In Beineke, L. W. and Wilson, R. J., editors, *Selected Topics in Graph Theory*, volume 3, chapter 7, pages 151–168. Academic Press.

[21] Cicirello, V. A. and Cernera, R. (2013). Profiling the distance characteristics of mutation operators for permutation-based genetic algorithms. In *Proceedings of the 26th International Florida Artificial Intelligence Research Society conference*, pages 46–51., St. Pete Beach, FL, USA. AAAI.

[22] Coffman, E., Garey, M., and Johnson, D. S. (1997). Approximation algorithms for np-hard problems. *SIGACT News*, 28(2):40–52.

[23] Corne, D. W. and Knowles, J. D. (2007). Techniques for Highly Multiobjective Optimisation: Some Nondominated Points are Better than Others. In *Proceedings of the 9th Genetic and Evolutionary Computation Conference*, volume 1, pages 773–780, London, UK. ACM Press.

[24] Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *International Journal of Complex Systems*, 1695(5):1–9.

[25] DaCosta, L., Fialho, A., Schoenauer, M., and Sebag, M. (2008). Adaptive operator selection with dynamic multi-armed bandits. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, pages 913–920, New York, NY, USA. ACM.

[26] Davis, L. (1985). Applying adaptive algorithms to epistatic domains. In *Proceedings of the 9th IJCAI*, volume 1, pages 162–164, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[27] Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 61–69, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[28] Diaz, J., Petit, J., and Serna, M. (2002). A survey of graph layout problems. *ACM Computing Surveys*, 34(3):313–356.

[29] Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39.

[30] Duarte, A., Martí, R., Resende, M. G. C., and Silva, R. M. A. (2011). GRASP with Path Relinking Heuristics for the Antibandwidth Problem. *Networks*, 58(3):171–189.

[31] Fialho, A., Da Costa, L., Schoenauer, M., and Sebag, M. (2008). Extreme value based adaptive operator selection. In Rudolph, G., Jansen, T., Beume, N., Lucas, S., and Poloni, C., editors, *Parallel Problem Solving from Nature – PPSN X*, pages 175–184, Berlin, Heidelberg. Springer Berlin Heidelberg.

[32] Fialho, A., Da Costa, L., Schoenauer, M., and Sebag, M. (2009). Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In Stützle, T., editor, *Learning and Intelligent Optimization*, pages 176–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

[33] Fialho, A., Da Costa, L., Schoenauer, M., and Sebag, M. (2010). Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60(1):25–64.

[34] Fisher, R. and Yates, F. (1938). *Statistical tables for biological, agricultural and medical research*. Oliver and Boyd, London.

[35] Fortnow, L. (2009). The status of the p versus np problem. *Commun. ACM*, 52(9):78–86.

[36] Fruchterman, T. M. J. and Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164.

[37] Gagliolo, M. and Schmidhuber, J. (2010). Algorithm selection as a bandit problem with unbounded losses. In *Proceedings of the 4th International Conference on Learning and Intelligent Optimization*, LION'10, pages 82–96, Berlin, Heidelberg. Springer-Verlag.

[38] Garza-Fabre, M. (2014). *On the use of alternative evaluation schemes in metaheuristics to handle the main search difficulties involved with the prediction of protein structures under the HP model*. PhD thesis, CINVESTAV-Tamaulipas.

[39] Garza-Fabre, M., Rodriguez-Tello, E., and Toscano-Pulido, G. (2013). Comparative analysis of different evaluation functions for protein structure prediction under the hp model. *Journal of Computer Science and Technology*, 28(5):868–889.

[40] Gendreau, M. and Potvin, J. Y., editors (2010). *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer, Berlin, Heidelberg.

[41] Gorsuch, R. L. (1988). *Exploratory Factor Analysis*, pages 231–258. Springer US, Boston, MA.

[42] Grimaldi, R. P. (1999). *Discrete and Combinatorial Mathematics: An Applied Introduction*. Pearson Education Inc., 5th edition.

[43] Groth, D., Hartmann, S., Klie, S., and Selbig, J. (2013). *Principal Components Analysis*, pages 527–547. Humana Press, Totowa, NJ.

[44] Guizzo, G., Fritsche, G., Vergilio, S., and Pozo, A. (2015). A hyper-heuristic for the multi-objective integration and test order problem. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 1343–1350, New York, NY, USA. ACM.

[45] Hagberg, A., Schult, D., and Swart, P. (2005). Networkx: Python software for the analysis of networks. *Mathematical Modeling and Analysis, Los Alamos National Laboratory*.

[46] Hamon, R., Borgnat, P., Flandrin, P., and Robardet, C. (2014). Heuristic for solving cyclic bandwidth sum problem by following the structure of the graph. *Tech. report, arXiv:1410.6108*.

[47] Hamon, R., Borgnat, P., Flandrin, P., and Robardet, C. (2016b). Relabelling vertices according to the network structure by minimizing the cyclic bandwidth sum. *Journal of Complex Networks*, 4(4):534–560.

[48] Hamon, R., Borgnat, P., Flandrin, P., and Robardet, C. (In press, 2016a). Relabelling vertices according to the network structure by minimizing the cyclic bandwidth sum. *Journal of Complex Networks*, 4(4):534–560.

[49] Harper, L. (1964). Optimal assignment of numbers to vertices. *Journal of the Society for Industrial and Applied Mathematics*, 12(1):131–135.

[50] Hernando, L., Mendiburu, A., and Lozano, J. A. (2013). An evaluation of methods for estimating the number of local optima in combinatorial optimization problems. *Evolutionary computation*, 21(4):625–658.

[51] Hernando, L., Mendiburu, A., and Lozano, J. A. (2016a). Estimating attraction basin sizes. In Luaces, O., Gámez, J. A., Barrenechea, E., Troncoso, A., Galar, M., Quintián, H., and Corchado, E., editors, *Advances in Artificial Intelligence*, pages 458–467, Cham. Springer International Publishing.

[52] Hernando, L., Mendiburu, A., and Lozano, J. A. (2016b). Estimating attraction basin sizes. In Luaces, O., Gámez, J. A., Barrenechea, E., Troncoso, A., Galar, M., Quintián, H., and Corchado, E., editors, *Advances in Artificial Intelligence*, pages 458–467, Cham. Springer International Publishing.

[53] Hinkley, D. V. (1971). Inference about the change-point from cumulative sum tests. *Biometrika*, 53(3):509–523.

[54] Hoos, H. H. and Stützle, T. (2004). *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann.

[55] Hordijk, W. (1996). A measure of landscapes. *Evol. Comput.*, 4(4):335–360.

[56] Humeau, J., Liefooghe, A., Talbi, E. G., and Verel, S. (2013). Paradiseo-mo: from fitness landscape analysis to efficient local search algorithms. *Journal of Heuristics*, 19(6):881–915.

[57] Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37(142):547–579.

[58] Jianxiu, H. (2001). Cyclic bandwidth sum of graphs. *Applied Mathematics - A Journal of Chinese Universities*, 16(2):115–121.

[59] Jinjiang, Y. (1995). Cyclic arrangement of graphs. In *Graph Theory Notes of New York*, pages 6–10. New York Academy of Sciences.

[60] Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proc. 6th Internat. Conf. on Genetic Algorithms*.

[61] Jourdan, L., Basseur, M., and Talbi, E.-G. (2009). Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3):620 – 629.

[62] Kallel, L., Naudts, B., and Reeves, C. R. (2001). *Properties of Fitness Functions and Search Landscapes*, pages 175–206. Springer Berlin Heidelberg, Berlin, Heidelberg.

[63] Kamada, T. and Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15.

[64] Kline, P. (2014). *An easy guide to factor analysis*. Routledge.

[65] Lai, T. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4 – 22.

[66] Lai, X. and Hao, J. (2016). Iterated maxima search for the maximally diverse grouping problem. *European Journal of Operational Research*, 254(3):780 – 800.

[67] Lam, P. C. B., Shiu, W. C., and Chan, W. H. (1997). On bandwidth and cyclic bandwidth of graphs. *Ars Combinatoria*, 47(3):147–152.

[68] Lenstra, J. K. (1997). *Local search in combinatorial optimization*. Princeton University Press.

[69] Li, Y. and Liang, Y. (2018). Compressed sensing in multi-hop large-scale wireless sensor networks based on routing topology tomography. *IEEE Access*, 6:27637–27650.

[70] Liberatore, V. (2002). Multicast scheduling for list requests. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1129–1137. IEEE.

[71] Lin, Y., Sỳkora, O., and Vrt'o, I. (1997). On cyclic cutwidths. Unpublished manuscript.

[72] López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., and Birattari, M. (2011). The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle Université libre de Bruxelles, Belgium.

[73] Lozano, J., González-Gurrola, L., Rodríguez-Tello, E., and Lacomme, P. (2016). A statistical comparison of objective functions for the vehicle routing problem with route balancing. In *2016 Fifteenth Mexican International Conference on Artificial Intelligence (MICAI)*, pages 130–135.

[74] Lozano, M., Duarte, A., Gortázar, F., and Martí, R. (2012). Variable neighborhood search with ejection chains for the antibandwidth problem. *Journal of Heuristics*, 6(18):919–938.

[75] Makedon, F. and Sudborough, I. H. (1989). On minimizing width in linear layouts. *Discrete Applied Mathematics*, 23(3):243 – 265.

[76] Malan, K. M. (2021). A survey of advances in landscape analysis for optimisation. *Algorithms*, 14(2):40.

[77] Malan, K. M. and Engelbrecht, A. P. (2013). A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148 – 163.

[78] Malan, K. M. and Engelbrecht, A. P. (2014). *Fitness Landscape Analysis for Metaheuristic Performance Prediction*, pages 103–132. Springer Berlin Heidelberg, Berlin, Heidelberg.

[79] Martí, R., Laguna, M., Glover, F., and Campos, V. (2001). Reducing the bandwidth of a sparse matrix with tabu search. *European Journal of Operational Research*, 135(2):211–220.

[80] Mladenovic, N., Urosevic, D., Pérez-Brito, D., and García-González, C. G. (2010). Variable neighbourhood search for bandwidth reduction. *European Journal of Operational Research*, 200(1):14–27.

[81] Monien, B. and Sudborough, I. H. (1990). Embedding one interconnection network in another. *Computational Graph Theory, Computing Supplementum*, 7:257–282.

[82] Moscato, P., Cotta, C., and Mendes, A. (2004). *Memetic Algorithms*, pages 53–85. Springer Berlin Heidelberg, Berlin, Heidelberg.

[83] Murovec, B. (2015). Job-shop local-search move evaluation without direct consideration of the criterion's value. *European Journal of Operational Research*, 241(2):320–329.

[84] Mutzel, P. (1995). A polyhedral approach to planar augmentation and related problems. In Spirakis, P., editor, *Algorithms — ESA '95: Proceedings of the 3rd Annual European Symposium, Corfu, Greece, September 25–27, 1995*, volume 979 of *Lecture Notes in Computer Science*, pages 494–507, Berlin, Heidelberg. Springer.

[85] Narvaez-Teran, M. V. (2016). Metaheuristics for the cyclic bandwidth sum minimization problem on graphs. Master's thesis, CINESTAV-Tamaulipas.

[86] Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., and Tack, G. (2007). Minizinc: Towards a standard cp modelling language. In Bessière, C., editor, *Principles and Practice of Constraint Programming – CP 2007*, pages 529–543, Berlin, Heidelberg. Springer Berlin Heidelberg.

[87] Newman, M. and Watts, D. (1999). Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4):341–346.

[88] Norman, M. G. and Moscato, P. (1991). A competitive and cooperative approach to complex combinatorial search. In *Proceedings of the 20th Informatics and Operations Research Meeting*, pages 3–15. Citeseer.

[89] Ochoa, G., Malan, K. M., and Blum, C. (2020). Search trajectory networks of population-based algorithms in continuous spaces. In Castillo, P. A., Jiménez Laredo, J. L., and Fernández de Vega, F., editors, *Applications of Evolutionary Computation*, pages 70–85, Cham. Springer International Publishing.

[90] Ochoa, G., Tomassini, M., Vérel, S., and Darabos, C. (2008). A study of nk landscapes' basins and local optima networks. In *Proceedings of the 10th Annual Conference on Genetic*

and Evolutionary Computation, page 555–562, New York, NY, USA. Association for Computing Machinery.

[91] Ochoa, G., Verel, S., Daolio, F., and Tomassini, M. (2014). *Local Optima Networks: A New Model of Combinatorial Fitness Landscapes*, pages 233–262. Springer Berlin Heidelberg, Berlin, Heidelberg.

[92] Oliver, I., Smith, D., and Holland, J. (1987). A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Application*, pages 224–230, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

[93] Palmer, R. (1991). Optimization on rugged landscapes. In *Molecular Evolution on Rugged Landscapes*, pages 3–25. CRC Press.

[94] Pesant, G. and Gendreau, M. (1999). A constraint programming framework for local search methods. *Journal of Heuristics*, 5(3):255–279.

[95] Piñana, E., Plana, I., Campos, V., and Martí, R. (2004). GRASP and path relinking for the matrix bandwidth minimization. *European Journal of Operational Research*, 153:200–210.

[96] Pitzer, E. and Affenzeller, M. (2012). A comprehensive survey on fitness landscape analysis. In Fodor, J., Klempous, R., and Suárez-Araujo, C. P., editors, *Recent Advances in Intelligent Engineering Systems*, volume 378 of *Studies in Computational Intelligence*, chapter 8, pages 161–191. Springer.

[97] Pitzer, E. and Affenzeller, M. (2013). Measurement of anisotropy in fitness landscapes. In Moreno-Díaz, R., Pichler, F., and Quesada-Arencibia, A., editors, *Computer Aided Systems Theory - EUROCAST 2013*, pages 340–347, Berlin, Heidelberg. Springer Berlin Heidelberg.

[98] Porumbel, D. C., Hao, J. K., and Kuntz, P. (2010). A search space "cartography" for guiding graph coloring heuristics. *Computers & Operations Research*, 37(4):769 – 778.

[99] Price, K. V. (2013). *Differential Evolution*, pages 187–214. Springer Berlin Heidelberg, Berlin, Heidelberg.

[100] Reidys, C. M. and Stadler, P. F. (2002). Combinatorial landscapes. *SIAM review*, 44(1):3–54.

[101] Richter, H. and Engelbrecht, A. (2014). *Recent advances in the theory and application of fitness landscapes*. Springer.

[102] Rodriguez-Tello, E. and Betancourt, L. C. (2012). An improved memetic algorithm for the antibandwidth problem. In Hao, J. K., Legrand, P., Collet, P., Monmarché, N., Lutton, E., and Schoenauer, M., editors, *Artificial Evolution*, pages 121–132, Berlin, Heidelberg. Springer Berlin Heidelberg.

[103] Rodriguez-Tello, E., Hao, J., and Romero-Monsivais, H. (2015). Boosting the performance of metaheuristics for the minla problem using a more discriminating evaluation function. *Tehnicki Vjesnik*, 22:11–24.

[104] Rodriguez-Tello, E., Hao, J. K., and Romero-Monsivais, H. (2015a). Boosting the performance of metaheuristics for the minla problem using a more discriminating evaluation function. *Tehnicki Vjesnik - Technical Gazette*, 22(1):11–24.

[105] Rodriguez-Tello, E., Hao, J.-K., and Torres-Jimenez, J. (2004). An improved evaluation function for the bandwidth minimization problem. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo-Guervós, J. J., Bullinaria, J. A., Rowe, J. E., Tiňo, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 652–661, Berlin, Heidelberg. Springer Berlin Heidelberg.

[106] Rodriguez-Tello, E., Hao, J.-K., and Torres-Jimenez, J. (2006). Memetic algorithms for the MinLA problem. *Lecture Notes in Computer Science*, 3871:73–84.

[107] Rodriguez-Tello, E., Hao, J. K., and Torres-Jimenez, J. (2008a). An effective two-stage simulated annealing algorithm for the minimum linear arrangement problem. *Computers & Operations Research*, 35(10):3331–3346.

[108] Rodriguez-Tello, E., Hao, J. K., and Torres-Jimenez, J. (2008b). An improved simulated annealing algorithm for bandwidth minimization. *European Journal of Operational Research*, 185(3):1319–1335.

[109] Rodriguez-Tello, E., Lardeux, F., Duarte, A., and Narvaez-Teran, V. (2019). Alternative evaluation functions for the cyclic bandwidth sum problem. *European Journal of Operational Research*, 273(3):904–919.

[110] Rodriguez-Tello, E., Narvaez-Teran, V., and Lardeux, F. (2018). Comparative study of different memetic algorithm configurations for the cyclic bandwidth sum problem. In *Parallel Problem Solving from Nature – PPSN XV*, pages 82–94, Cham. Springer International Publishing.

[111] Rodriguez-Tello, E., Narvaez-Teran, V., and Lardeux, F. (2019). Dynamic multi-armed bandit algorithm for the cyclic bandwidth sum problem. *IEEE Access*, 7:40258–40270.

[112] Rodriguez-Tello, E., Romero-Monsivais, H., Ramirez-Torres, G., and Lardeux, F. (2015b). Tabu search for the cyclic bandwidth problem. *Computers & Operations Research*, 57:17–32.

[113] Rosé, H., Ebeling, W., and Asselmeyer, T. (1996). The density of states - a measure of the difficulty of optimisation problems. In Voigt, H.-M., Ebeling, W., Rechenberg, I., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature — PPSN IV*, pages 208–217, Berlin, Heidelberg. Springer Berlin Heidelberg.

[114] Rosen, K. H. (2007). *Discrete Mathematics and Its Applications*. McGraw-Hill, Harlow Essex, 6th edition.

[115] Safro, I., Ron, D., and Brandt, A. (2006). Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms*, 60(1):24–41.

[116] Satsangi, D. (2013). *Design and development of metaheuristic techniques for some graph layout problems*. PhD thesis, Deparment of Mathematics, Dayalbagh Educational Institute, Deemed University.

[117] Satsangi, D., Srivastava, K., and Gursaran (2012). General variable neighbourhood search for cyclic bandwidth sum minimization problem. In *Proceedings of the Students Conference on Engineering and Systems*, pages 1–6. IEEE Press.

[118] Schiavinotto, T. and Stutzle, T. (2007). A review of metrics on permutations for search landscape analysis. *Computers & Operations Research*, 34(10):3143–3153.

[119] Schulte, C., Tack, G., and Lagerkvist, M. Z. (2019). Modeling and programming with gecode. Technical report.

[120] Shahrokhi, F., Sỳkora, O., Székely, L. A., and Vrt'o, I. (2001). On bipartite drawings and the linear arrangement problem. *SIAM Journal on Computing*, 30(6):1773–1789.

[121] Sipser, M. (1992). The history and status of the p versus np question. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, page 603–618, New York, NY, USA. Association for Computing Machinery.

[122] Smet, P., Bilgin, B., De Causmaecker, P., and Vanden Berghe, G. (2014). Modelling and evaluation issues in nurse rostering. *Annals of Operations Research*, 218(1):303–326.

[123] Smith, T., Husbands, P., and O'Shea, M. (2002). Fitness landscapes and evolvability. *Evolutionary computation*, 10(1):1–34.

[124] Smith-Miles, K. and Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, 39(5):875–889.

[125] Smith-Miles, K. A. (2008). Towards insightful algorithm selection for optimisation using meta-learning concepts. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 4118–4124.

[126] Stadler, P. F. (1996). Landscapes and their correlation functions. *Journal of Mathematical Chemistry*, 20(1):1–45.

[127] Sun, Y., Halgamuge, S. K., Kirley, M., and Munoz, M. A. (2014). On the selection of fitness landscape analysis metrics for continuous optimization problems. In *7th International Conference on Information and Automation for Sustainability*, pages 1–6.

[128] Talbi, E. (2009). *Metaheuristics: From design to implementation*. John Wiley & Sons, Hoboken NJ.

[129] Thurstone, L. L. (1947). Multiple-factor analysis; a development and expansion of the vectors of mind.

[130] Trandac, H. and Duong, V. (2002). A constraint-programming formulation for dynamic airspace sectorization. *Proceedings. The 21st Digital Avionics Systems Conference*, 1:1C5–1C5.

[131] Vanneschi, L. (2004). *Theory and practice for efficient genetic programming*. PhD thesis, Université de Lausanne, Faculté des sciences.

[132] Vanneschi, L., Tomassini, M., Collard, P., and Vérel, S. (2006). Negative slope coefficient: A measure to characterize genetic programming fitness landscapes. In Collet, P., Tomassini, M., Ebner, M., Gustafson, S., and Ekárt, A., editors, *Genetic Programming*, pages 178–189, Berlin, Heidelberg. Springer Berlin Heidelberg.

[133] Vanneschi, L., Tomassini, M., Collard, P., Vérel, S., Pirola, Y., and Mauri, G. (2007). A comprehensive view of fitness landscapes with neutrality and fitness clouds. In Ebner, M., O'Neill,

M., Ekárt, A., Vanneschi, L., and Esparcia-Alcázar, A. I., editors, *Genetic Programming*, pages 241–250, Berlin, Heidelberg. Springer Berlin Heidelberg.

[134] Verel, S., Collard, P., and Clergue, M. (2003). Where are bottlenecks in nk fitness landscapes? In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 1, pages 273–280 Vol.1.

[135] Weinberger, E. (1990). Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63(5):325–336.

[136] White, D. and Newman, M. E. J. (2001). Fast approximation algorithms for finding node-independent paths in networks. *SSRN Electronic Journal*.

[137] Whitley, D. and Sutton, A. M. (2012). Genetic algorithms-a survey of models and methods. In *Handbook of natural computing*, pages 637–671. Springer Berlin Heidelberg.

[138] Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *26th International Congress of Genetics*.

[139] Yuan, J. (1995). Cyclic arrangement of graphs. *Graph Theory Notes of New York, New York Academy of Sciences*, pages 6–10.

[140] Zhang, Y., Wang, S., and Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015.