Mayra Carvalho Albuquerque

Matheuristics for Variants of the Dominating Set Problem

TESE DE DOUTORADO

DEPARTAMENTO DE INFORMÁTICA
Programa de Pós–Graduação em Informática

Rio de Janeiro
February 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Mayra Carvalho Albuquerque

Matheuristics for Variants of the Dominating Set Problem

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática.

Advisor: Prof. Thibaut Victor Gaston Vidal

Rio de Janeiro
February 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Mayra Carvalho Albuquerque

Matheuristics for Variants of the Dominating Set Problem

Thesis presented to the Programa de Pós-Graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática. Approved by the undersigned Examination Committee.

Prof. Thibaut Victor Gaston Vidal
Advisor
Departamento de Informática — PUC–Rio

Prof. Hélio Côrtes Vieira Lopes
Departamento de Informática – PUC-Rio

Prof. Marco Serpa Molinaro
Departamento de Informática – PUC-Rio

Prof. Yuri Abitbol de Menezes Frota
Universidade Federal Fluminense – UFF

Prof. Igor Machado Coelho
Universidade do Estado do Rio de Janeiro – UERJ

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies
Centro Técnico Científico — PUC–Rio

Rio de Janeiro, February 8th, 2018

Mayra Carvalho Albuquerque

Mayra Carvalho Albuquerque holds a Master's in Computational and Mathematical Modeling from Centro Federal de Educação Tecnológica de Minas Gerais (CEFET/MG) and graduated in Mathematics from Universidade Federal de Viçosa (UFV). She held a scholarship from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) during the doctorate and Master's degree programs. She has experience teaching in mathematics and computer science in higher education.

Bibliographic data

Albuquerque, Mayra Carvalho

Matheuristics for Variants of the Dominating Set Problem / Mayra Carvalho Albuquerque; advisor: Thibaut Victor Gaston Vidal. — 2018.

87 f. : il. (color.); 30 cm

Tese (Doutorado em Informática) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Departamento de Informática, 2018.

Inclui bibliografia.

1. Informática – Teses. 2. Conjunto Dominante. 3. Código de Cobertura. 4. Busca Tabu. 5. Mateurísticas. 6. Busca em Vizinhança Larga. I. Vidal, Thibaut Victor Gaston. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To the teachers
who changed my life.

Acknowledgement

To Thibaut Vidal, for guiding and supporting me over the last two and a half years.

To family and friends, for their encouragement.

To CNPq, for the financial support.

Also my thanks to God.

Abstract

Albuquerque, Mayra Carvalho; Vidal, Thibaut Victor Gaston (Advisor). **Matheuristics for Variants of the Dominating Set Problem**. Rio de Janeiro, 2018. 87p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This thesis addresses the Dominating Set Problem, an NP-hard problem with great relevance in applications related to wireless network design, data mining, coding theory, among others. The minimum dominating set in a graph is a minimal set of vertices so that each vertex of the graph belongs to it or is adjacent to a vertex of this set. We study three variants of the problem: first, in the presence of weights on vertices, searching for a dominating set with smallest total weight; second, a variant where the subgraph induced by the dominating set needs to be connected, and, finally, the variant that encompasses these two characteristics. To solve these three problems, we propose a hybrid algorithm based on tabu search with additional mathematical-programming components, leading to a method sometimes called "*matheuristic*". Several additional techniques and large neighborhoods are also employed to reach promising regions in the search space. Our experimental analyses show the good contribution of all these individual components. Finally, the algorithm is tested on the covering code problem, which can be viewed as a special case of the minimum dominating set problem. The codes are studied for the Hamming metric and the Rosenbloom-Tsfasman metric. For this last case, several shorter codes were found.

Keywords

Dominating Set; Covering Code; Tabu Search; Matheuristics; Large Neighborhood Search.

Resumo

Albuquerque, Mayra Carvalho; Vidal, Thibaut Victor Gaston. **Mateurísticas para Variantes do Problema do Conjunto Dominante**. Rio de Janeiro, 2018. 87p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Esta tese faz um estudo do problema do Conjunto Dominante, um problema NP-difícil de grande relevância em aplicações relacionadas ao projeto de rede sem fio, mineração de dados, teoria de códigos, dentre outras. O conjunto dominante mínimo em um grafo é um conjunto mínimo de vértices de modo que cada vértice do grafo pertence a este conjunto ou é adjacente a um vértice que pertence a ele. Três variantes do problema foram estudadas; primeiro, uma variante na qual considera pesos nos vértices, buscando um conjunto dominante com menor peso total; segundo, uma variante onde o subgrafo induzido pelo conjunto dominante está conectado; e, finalmente, a variante que engloba essas duas características. Para resolver esses três problemas, propõe-se um algoritmo híbrido baseado na meta-heurística busca tabu com componentes adicionais de programação matemática, resultando em um método por vezes chamado de "*mateurística*", (*matheuristic*, em inglês). Diversas técnicas adicionais e vizinhanças largas foram propostas afim de alcançar regiões promissoras no espaço de busca. Análises experimentais demonstram a contribuição individual de todos esses componentes. Finalmente, o algoritmo é testado no problema do código de cobertura mínima, que pode ser visto como um caso especial do problema do conjunto dominante. Os códigos são estudados na métrica Hamming e na métrica Rosenbloom-Tsfasman. Neste último, diversos códigos menores foram encontrados.

Palavras–chave

Conjunto Dominante; Código de Cobertura; Busca Tabu; Mateurísticas; Busca em Vizinhança Larga.

Contents

1 Introduction **12**
1.1 Context and Challenges 14
1.2 Objectives and Contributions 15
1.3 Thesis Outline 16

2 A Review of Dominating Set Problems **18**
2.1 Problem Statement and Basic Concepts 18
2.2 Some Variants of the Dominating Set 20
2.3 Covering Codes 25
2.4 Other Applications of the MDS Problem 26

3 A matheuristic for the Minimum Weight Dominating Set Problem **28**
3.1 Mathematical Formulation 28
3.2 Greedy Constructive Heuristics 29
3.3 A Hybrid Tabu Search 30
3.4 Computational Experiments 38
3.5 Concluding Remarks 50

4 Extension to solve the Weight Connected Dominating Set Problem **51**
4.1 Problem Statement and Basic Concepts 51
4.2 A Branch-and-Cut Approach 51
4.3 Methodology 52
4.4 Computational Experiments 55
4.5 Experiments on the Connected Dominating Set 65
4.6 Concluding Remarks 69

5 Experiments on the Covering Code Problems **70**
5.1 Preliminaries 70
5.2 Covering Codes and the Dominating Set Problem 72
5.3 Computational Experiments 73
5.4 Concluding Remarks 78

6 Conclusions and Future Works **79**

List of figures

1.1 Example of an ad hoc network topology 12
1.2 Example of a dominating set in an ad hoc network. 13

2.1 Example of the topology of a unit disk graph 23
2.2 Timeline of some works for MWDS, MCDS and MWCDS problems 24
2.3 Example of protein-protein interaction network (reprinted from Wikimedia (2017)). 27

3.1 HTS-DS algorithm flowchart 31
3.2 Evolution of the penalty parameter 33
3.3 Computational time of each component of HTS-DS as a function of $|\mathcal{V}|$ 47

4.1 Computational time via a least-squares regression of an affine function on the log-log graph $|\mathcal{V}| \times \mathsf{T}(s)$. 63
4.2 Solution quality of the HTS-DS algorithm as a function of the size of tabu list. 64
4.3 Solution quality of the HTS-DS algorithm as a function of the penalty factor β. 64

5.1 Graph $\mathcal{G}(2, 2, 2, 2)$. 72
5.2 Representation of the code $K_3(6, 1)$. 76

List of tables

3.1 Parameter configuration of HTS-DS 39
3.2 Comparison of the greedy heuristics for small and medium Type I instances 40
3.3 Comparison of the greedy heuristics for large Type I instances 40
3.4 Comparison of the greedy heuristics for small and medium Type II instances 41
3.5 Comparison of the greedy heuristics for large Type II instances 41
3.6 List of methods considered in the experiments, and CPU model information 42
3.7 Type T1, Class SMPI – Comparison of HTS-DS with recent state-of-the-art algorithms 43
3.8 Type T1, Class LPI – Comparison of HTS-DS with recent state-of-the-art algorithms 44
3.9 Type T2, Class SMPI – Comparison of HTS-DS with recent state-of-the-art algorithms 45
3.10 Type T2, Class LPI – Comparison of HTS-DS with recent state-of-the-art algorithms 46
3.11 Analysis of HTS-DS components 49

4.1 Parameter configuration of HTS-DS 56
4.2 Type-I, Small Class – Comparison of HTS-DS with recent state-of-the-art algorithms 57
4.3 Type-I, Moderate Class – Comparison of HTS-DS with recent state-of-the-art algorithms 58
4.4 Type-I, Large Class – Comparison of HTS-DS with recent state-of-the-art algorithms 59
4.5 Type-II, Small Class – Comparison of HTS-DS with recent state-of-the-art algorithms 59
4.6 Type-II, Moderate Class – Comparison of HTS-DS with recent state-of-the-art algorithms 60
4.7 Dagdeviren's dataset, small class comparison of HTS-DS with recent state-of-the-art algorithms 61
4.8 Dagdeviren's Dataset, Moderate Class – Comparison of HTS-DS with recent state-of-the-art algorithms 62
4.9 Dagdeviren's Dataset, Large Class – Comparison of HTS-DS with recent state-of-the-art algorithms 62
4.10 Parameter configuration of HTS-DS for the MCDS problem 65
4.11 List of methods considered in the experiments, and CPU model information 66
4.12 Performance of the HTS-DS on LMS and BPFTC instances in comparison with the current state-of-the-art algorithms 67
4.13 Performance of the HTS-DS on RGG instances in comparison with the current state-of-the-art algorithms 68

5.1 Parameter configuration of HTS-DS 73

5.2 Results for covering codes $K_2^{RT}(m, s, R)$: Graph $\mathcal{G}(m, s, 2, R)$ 75
5.3 Results for covering $K_q(n, R)$: Graph $\mathcal{G}(n, q, R)$ 76
5.4 Analysis of HTS-DS components for the covering code problem 77

1
Introduction

New technologies for wireless data communication devices have been gaining tremendous rise in popularity among researchers and practitioners. A common application of these technologies comes from in wireless ad hoc networks, in which a collection of mobile devices/nodes dynamically composes a temporary network without any centralized administration. The nodes in the network act as routers when exchanges of messages occurs between two nodes. A dominating node is the one that can send a message to another node by forwarding the message through its neighbor's dominating node, within the transmission range. Figure 1.1 illustrates an ad hoc network topology. The circles show the coverage area of different mobile/fixed devices (nodes), representing the ability of nodes to communicate.

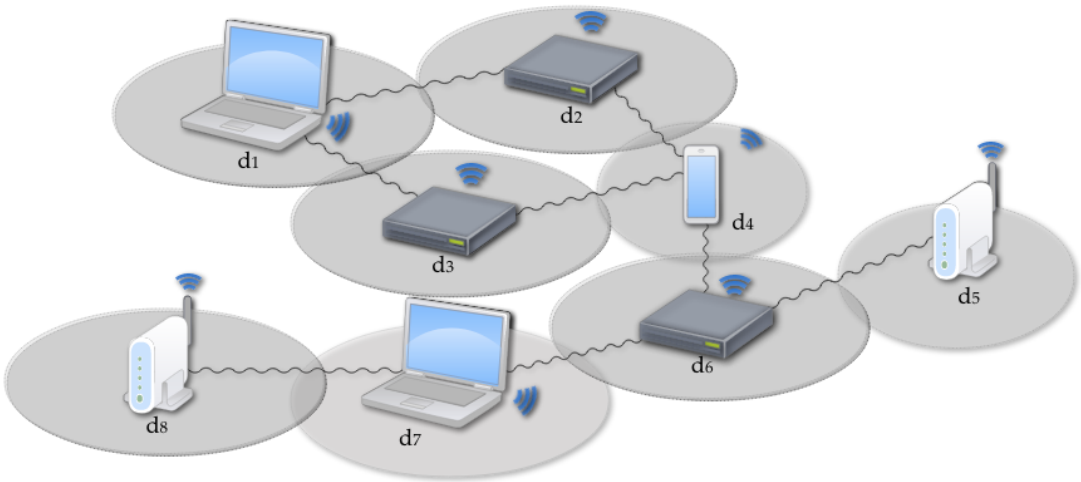


Figure 1.1: Example of an ad hoc network topology

Dominating sets are a highly relevant subject of study in the context of wireless networks. A dominating set for a graph is a subset of vertices that has the following property: each vertex either belongs to the subset or is adjacent to a vertex in the subset. The *minimum dominating set* (MDS) problem consists of determining a dominating set of minimum cardinality.

For example, Figure 1.2 illustrates a dominating set for the ad hoc network presented in Figure 1.1. The devices/nodes d_1, d_6 and d_8 guarantee that all nodes' devices receive a message. Node d_8 is in the dominating set,

even though node d_7 is already dominated by the node d_6; node d_8 must be included in the dominating set so it can be dominated, in this case, by itself.

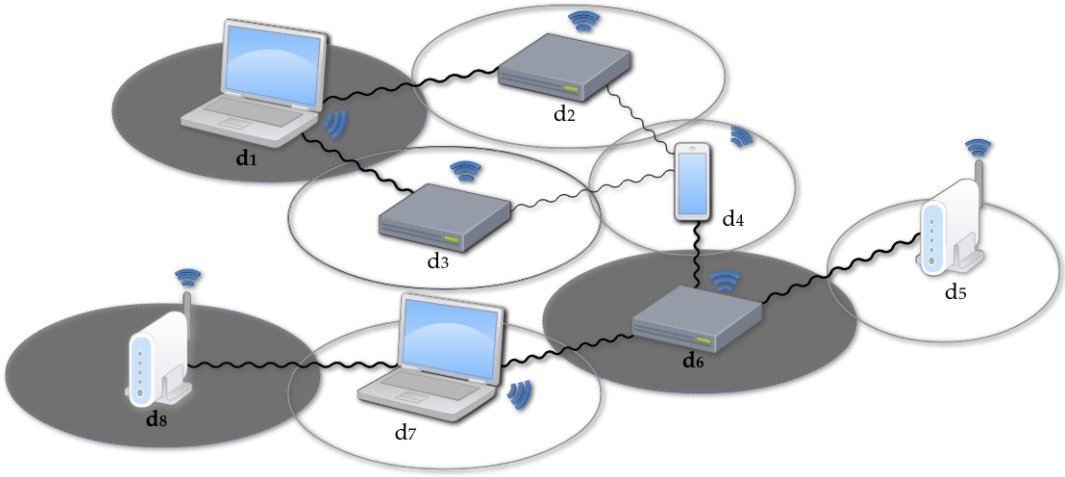


Figure 1.2: Example of a dominating set in an ad hoc network.

Dominating set problems are linked to a rich set of applications, for example, the design of wireless sensor networks (Yu et al., 2013), the study of social networks and influence propagation (Wang et al., 2011), protein interaction networks (Wuchty, 2014; Nacher & Akutsu, 2016) and covering codes (Östergård, 1997), among others.

There are many variants of the DS problem proposed in the literature. Here we focus on the three most studied ones. In a variant of this problem, called Minimum Weight Dominating Set problem (MWDS), a non-negative weight is defined for each vertex, and the objective is to find a dominating set of minimum total weight. In another variant, known as the Minimum Connected Dominating Set (MCDS), the subgraph induced by the vertices in the dominating set has to be connected with minimum cardinality. The third and last variant includes the two previous characteristics: the Minimum-Weight Connected Dominating Set Problem (MWCDS).

The Domination Set problems have been the subject of many studies in graph theory, belonging to the family of covering problems. We reformulate the covering code problem in terms of the dominating set problem. A covering code is a set of elements, called codewords, in a space, with the property that every element of the space is within a fixed distance of some codeword. The covering code problem is to find the smallest subset of codewords which covers the remaining codewords. The fixed distance of the classic covering problem is determined by the Hamming-metric. This metric can be extended

to the Rosenbloom–Tsfasman (RT) metric, arising the RT-covering problem (Rosenbloom & Tsfasman, 1997).

Many studies have been dedicated to finding lower and upper bounds for the smallest cardinality of the covering codes. In particular, Castoldi & Carmelo (2015) examined the bounds for the codes in an RT-space. Since the RT-covering problems are directly related to the dominating set problem, we also deal with this problem. Therefore, a part of this study is dedicated to improving upper bounds for the covering codes in an RT-space. Finally, a brief study of the Hamming-covering codes is carried out.

1.1 Context and Challenges

Since the minimum dominating set is a problem of the NP-hard class Garey & Johnson (1990), developing efficient algorithms in acceptable computational time is a big challenge. Most combinatorial optimization problems have been widely studied due to the lack of such efficient algorithms to solve them accurately, in an acceptable processing time. In addition to the theoretical motivation, these problems appear frequently in many real-life applications where the aim is to improve or to find a satisfactory and sufficiently good solutions.

The MDS can be viewed as a special case of set covering (SC) problem, in which each vertex corresponds to a possible set. Although the research on exact methods has culminated in very efficient solution techniques for SC, many instances of MDS variants present graphs with medium or high densities, leading to SC instances with large sets (i.e., dense matrices), which can be unusually challenging. For this reason, along with emerging applications in machine learning and social networks analysis, a line of research specific to the MDS has been growing in intensity in recent years.

Much effort has been spent on finding approximation algorithms for DS problems. Among the most relevant works are Johnson (1974), Chen et al. (2004), Zou et al. (2011), and Schaudt & Schrader (2012). Exact methods and formulations have been proposed mainly for the MCDS problem. These approaches are unable to solve most instances in a reasonable time (van Rooij & Bodlaender, 2011; Lucena et al., 2010; Gendron et al., 2014).

Nonetheless, there has been not lot of advances in the development of exact algorithms for dealing with the DS, particularly those based on mathematical programming techniques. Heuristics methods are the most common procedure employed to solve combinatorial problems due to their ability of obtaining solution quality in an acceptable time.

A special attention must be given to metaheuristics, which are well-known class of solution method for combinatorial optimization problems. These methods are often used to find satisfactory solutions to large and complex problems, both theoretical and practical. However, solution infeasibility and set redundancy are some difficulties that arise when solving MDS problems with metaheuristic approaches.

Although several authors have studied dominating set problems only in recent years there has been increasing interest in this problem, from the point of view of metaheuristic algorithms (see, e.g., Potluri & Singh, 2011; Hedar & Ismail, 2012; Bouamama & Blum, 2016). This has led to three publications last year: Dagdeviren et al. (2017), who studied the MWCDS problem; Wu et al. (2017) and Wang et al. (2017) who investigated the MCDS and MWDS problem, respectively. Most of these studies used population-based methods, but these methods can produce a premature convergence, and their computational time can increase quickly with the size of the problem. None of these methods reliably finds the optimal solutions for all instances, and their computational time tends to be high for large graphs. The identification of fundamental design components for the heuristics is of primary interest to progress toward more generalist and efficient algorithms for the DS problem.

1.2 Objectives and Contributions

To address these challenges, this thesis focuses on the study of the variants of the MDS problem. First, we consider the MWDS problem where the objective function is to minimize the total weight of the dominating set. Second, we consider the MCDS problem, in which the subgraph induced by the dominating vertex set should be connected. Finally, we consider the combination of both problems, the MWCDS problem.

The main objective of this thesis is to study the dominating set problem and its variants in undirected graphs to find high-quality solutions using algorithms based on the hybridization of metaheuristics and integer linear programming (ILP) solvers. We intend to contribute to the area of matheuristics and this problem class for many applications. The term "*matheuristic*" is generally used to refer to hybrid metaheuristics built upon some mathematical programming components (Maniezzo et al., 2009). The key to achieve a good performance for many problems is to choose an adequate combination of these methods. In addition, we designed a *periodic ramp-up* strategy, which integrates destruction and construction operators, moving the search to an under-explored area of the search space.

Essentially, much of this study has been concentrated on the MDS problem. Few studies using metaheuristic techniques have been proposed to solve this problem. There is a need to study hybrid algorithms for this problem, using a method that combines metaheuristics and mathematical programing-based methods.

Considering these observations and the general objective of this thesis, the main contributions of the work are listed below:

- We conduct a study of many variants of dominating set problems. We investigate the MWDS, MCDS and MWCDS as well as the related Covering Code Problem. We describe some practical applications and solution methods proposed in the literature for these variants.
- We propose a method that combines four successful strategies: an efficient neighborhood search, an adaptive penalty scheme to explore intermediate infeasible solutions, perturbation phases to promote exploration, and an intensification mechanism in the form of an integer programming (IP) solver, which is applied to solve partial problems in which a majority of variables are fixed, keeping free those of the current best known solution and those associated with *promising* vertices.
- We present an algorithm that encompasses several useful solution features. We introduce a tabu search matheuristic for the DS problem, by combining tabu search metaheuristic and mathematical programming. We conduct extensive experiments and show that this hybrid algorithm can solve large class of MDS problems. Finally, we improve the current methods for the minimum dominating set problem, in terms of both solution quality and execution time in most cases.
- Finally, we study the covering code problem in RT-metric and the classic codes in the Hamming metric. We apply the hybrid algorithm to find new upper bounds.

1.3 Thesis Outline

The thesis is organized as follows.

Chapter 2 addresses some the variants of the dominating set and its main applications. Chapter 3 presents the matheuristic, an integrating a tabu search metaheuristic in combination with integer programming, for the MWDS problem. In Chapter 4, this matheuristic is adapted for the MWCDS and MCDS problem. Chapter 5 describes the relation of the covering code problem and the dominating set. In addition, some upper bounds for the covering

codes in RT-metric and Hamming metric are improved with the proposed algorithm. Finally, Chapter 6 concludes, summarizing the final considerations, and suggests perspectives for future works.

2
A Review of Dominating Set Problems

Many applications dealing with DS problems can be found in the literature. However, different variants can have some specific characteristics: weighted, connected, independent, capacitated and so on. This chapter presents a literature review, taking a step back from the dominating set problem and aiming to better understand the content of this problem and its main variants and applications.

Section 2.1 provides an introduction and some essential definitions for an understanding of the DS problem, as well as, a mathematical formulation. Section 2.2 reviews the literature on variants of the DS problem and presents state-of-art methods. Section 2.3 describes the covering code problem and its relation to the DS problem. Section 2.4 presents some DS applications.

2.1
Problem Statement and Basic Concepts

DS problems constitute a fundamental class of combinatorial optimization problems. Unsurprisingly, domination in graphs has been well studied in graph theory (see, e.g., Ore, 1962; Hedetniemi et al., 1986). For a more thorough review of different aspects of domination in graph, we recommend the survey by Haynes et al. (1998).

Formally, given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$, $\mathcal{S}$ of $\mathcal{V}$ is said to be a *dominating set* of $\mathcal{G}$ if, for every vertex $i \in \mathcal{V}$, either i is in $\mathcal{S}$ or there exists a vertex $j \in \mathcal{S}$, such that $(i, j) \in \mathcal{E}$. A dominating set of smallest cardinality is known as a *minimum dominating set*.

The vertices in $\mathcal{S}$ and $\mathcal{V} \backslash \mathcal{S}$ are called dominating and dominated (or covered), respectively. A vertex $i \in \mathcal{S}$ of an undirected graph is *redundant* if all adjacent vertices are either dominated or belong to a dominating set. If a vertex j is not adjacent to at least one vertex of $\mathcal{S}$, it is said to be a non-dominated (or uncovered) vertex. The *span* of a vertex i is the number of non-dominated vertices that are adjacent to i. We denote this set by $\Delta(i)$. The *distance* between two vertices i and j of $\mathcal{V}$ is defined as the shortest path from i to j in $\mathcal{G}$. We assume that the graphs are simple and undirected.

2.1.1 Mathematical formulation of the MDS problem

A simple mathematical formulation of the MDS is displayed in Equations (2-1) - (2-3). In this formulation, the closed neighborhood $N[i]$ represents all vertices adjacent to i, including itself (i.e., $N[i] = \{i\} \cup \{j \mid (i,j) \in \mathcal{E}\}$). In words, a vertex of $\mathcal{G}$ dominates itself and all its adjacent vertices. Each decision variable x_i is set to 1 if the vertex i is included in the dominating set, and 0 otherwise.

$$\min \quad \sum_{i \in V} x_i \tag{2-1}$$

$$\text{s.t.} \quad \sum_{k \in N[i]} x_k \geq 1 \qquad \forall i \in V \tag{2-2}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{2-3}$$

The objective function (2-1) minimizes cardinality of the dominating set. Constraints (2-2) ensure that each vertex j is dominated by at least one vertex in the dominating set. These constraints are called *cover inequalities* and define a dominating set solution. Finally, (2-3) defines the integrality constraints.

2.1.2 Related works

Dominating set problems are closely tied to the set covering (SC) problem. Consider a universe $\mathcal{U}$ of elements and a collection $\mathcal{S}$ of subsets of $\mathcal{U}$. A *set cover* of $\mathcal{S}$ is a subset $\mathcal{S}'$ of $\mathcal{S}$ which covers $\mathcal{U}$. The minimum set covering (MSC) problem if to find a set covering with minimum cardinality.

Many instances of the variants MDS present graphs with medium or high densities, leading to SC instances with large sets (i.e., dense matrices), which can be unusually challenging. For this reason, along with emerging applications in machine learning and social network analysis, a line of research specific to MDS has been growing in intensity in recent years.

Several approximation algorithms have been considered to solve the DS problem. This literature in the cases of solutions with approximation algorithms is discussed in Grandoni (2006), who proposed an algorithm which solves the MDS in $O(1.9053^n)$ time, where n is the number of vertices in the graph. van Rooij & Bodlaender (2011) proposed an improvement via a complexity for the dominating set regarding the exact algorithm. More specifically, they demonstrated that an algorithm exists that solves the dominating set problem in $O(1.4969^n)$ time and polynomial space.

Many algorithmic methods have been studied for variations of the dominating set problem in different classes of graphs. For example, Chlebík & Chlebíková (2008) studied various kinds of domination problems (connected, total, independent) in bounded degree graphs. Specifically, for degree bounded approaching infinity, these bounds almost match the known upper bounds.

Exact algorithms to solve the MDS problem can found in Fomin et al. (2005); Grandoni (2006); van Rooij & Bodlaender (2011). Particularly, van Rooij & Bodlaender (2011) developed a branch-and-reduce method and obtained good results for the DS problems and variants. Although these algorithms find an optimal solution, they are time consuming in graphs with a few hundreds of vertices. Therefore, these algorithms become impractical for very large graphs. This drawback leads to considering metaheuristics to design more efficient algorithms to solve the MWDS problem, as we detail in the approaches for each variant in the next section.

2.2 Some Variants of the Dominating Set

2.2.1 Minimum-Weight Dominating Set Problem

The DS problem has many variants, the version that deals with a weight on the vertices of the graph, called the MWDS problem. The objective function seeks to minimize the total weight, without regard for the cardinality of the dominating set.

Earlier studies of the MWDS were focused on approximation algorithms. The first constant-factor approximation algorithm for this setting was proposed by Ambühl et al. (2006), with an approximation ratio of 72 for the unit disk graphs. Subsequently, this ratio has been improved, down to $(6+\epsilon)$ in Huang et al. (2009) using a "double-partition" strategy, followed by $(5+\epsilon)$ and $(4+\epsilon)$ in Dai & Yu (2009) and Erlebach & Mihalák (2010).

The first polynomial time approximation scheme (PTAS) was introduced by Zhu et al. (2012), with an approximation ratio of $(1+\epsilon)$ for the specific case where the ratio of the weights of any two adjacent nodes is upper bounded by a constant. Finally, Li & Jin (2015) introduced a PTAS for the general case without restrictions on adjacent weights.

The research on metaheuristics for the MWDS has also grown recently. An ant colony algorithm with a pheromone correction strategy (Raka-ACO) was proposed in Jovanovic et al. (2010). Subsequently, Potluri & Singh (2013) introduced a hybrid genetic algorithm (HGA) and two extensions of the ACO

algorithm with a local search which consists of removing redundant vertices. In the second algorithm, a pre-processing step is included immediately after pheromone initialization, in order to reinforce the pheromone values associated to 100 independent sets generated via a greedy algorithm.

An iterated greedy algorithm (R-PBIG) was introduced by Bouamama & Blum (2016). The method maintains a population of solutions and applies deconstruction and reconstruction steps for improvement. This approach was then hybridized with an ILP solver for solution improvement.

Finally, Wang et al. (2017) introduced a local search with additional mechanisms to prevent cycling. These above methods produce solutions having good quality for classic sets of instances, but they sometimes suffer from premature convergence in some cases (such as the population-based algorithms), or tend to be over-restrictive (cycling prevention), and require a significant amount of CPU time for the largest instances.

2.2.2 Minimum Connected Dominating Set Problem

An important variation of the minimum dominating set problem is its connected version. A CDS is a connected DS, that is, there is a path between every pair of its vertices. The vast majority of research has focused on applications of this problem for network sensors, surveyed in Blum et al. (2005) and Resende & Pardalos (2008). Yu et al. (2013) also discussed various network models and applications and presented a comprehensive survey on the problem.

The MCDS problem is equivalent to finding a spanning tree with the maximum number of leaves in the graph (Fernau et al., 2011). The Maximum Leaf Spanning Tree (MLST) problem aims is to find a spanning tree with as many leaves as possible. Therefore, the internal nodes of a spanning tree with k leaves form a connected dominating set of size $|V| - k$ and vice versa. Another problem that is directly related to the CDS is the group Steiner problem. This problem is a Steiner generalization where only a specified subset of vertices has to be dominated by a connected dominating set (Elbassioni et al., 2012).

Due to its importance, many algorithms have also been proposed to solve the CDS problem. Two approximation algorithms to construct a connected dominating set in general graphs were first proposed by Guha & Khuller (1998). Ruan et al. (2004) developed a greedy algorithm to solve the problem. Li et al. (2005) then proposed another greedy algorithm with two phases that can construct a CDS within a factor of $(4.8 + \ln 5)$ from the optimal solution. In the first phase, a minimum independent set (MIS) is found, and in the second

phase, a Steiner tree algorithm is used to connect the MIS.

Some exact approaches have been developed for the MCDS problem, such as those of Fujie (2003), Lucena et al. (2010), Fernau et al. (2011), and Gendron et al. (2014). In more details, Lucena et al. (2010) introduced two integer programming formulations: the first one is based on a Steiner reformulation of the problem, and the second considers the problem in a directed graph, seeking later a spanning arborescence with as many leaves as possible. This work was extended by Gendron et al. (2014), who investigated exact algorithms based on two approaches: a Benders decomposition and a branch-and-cut method for a class of instances of the MCDS problem. Moreover, the authors proposed a hybrid algorithm considering an initial master problem at each iteration, and adding valid inequalities from the spanning tree formulation in both approaches. This algorithm achieved the optimal solution in almost all instances, but with a high running time.

In contrast, the research on efficient metaheuristics has expanded fairly recently for the MCDS problem. Jovanovic & Tuba (2013) proposed two heuristics based on the ant colony algorithm, the first one is a Min-Max Ant System, while the second one is an ACO with a pheromone correction strategy, which was first proposed in Jovanovic et al. (2010) for the MWDS problem.

Wu et al. (2017) and Li et al. (2017) proposed, respectively, a restricted swap-based neighborhood structure and a greedy randomized adaptive search procedure (GRASP) based on a greedy function to solve the MCDS problem. Both methods improve the solutions, by using strategies to forbid cycling in the space search. These methods reach good solutions compared to the population-based method and exact methods. Currently, the work of Wu et al. (2017) is the state of the art for the MCDS. We study this problem and show the details of our approach in Chapter 4.

2.2.3 Minimum-Weight Connected Dominating Set Problem

The MWCDS problem arises when the total weight of CDS is aimed to be minimized. MWCDS can be applied in an ad hoc network, particularly in the unit-disk graph (UDG). A UDG is a graph in which every vertex corresponds to a sensor in the plane, and in which two vertices are connected by an edge if their Euclidean distance in the plane is smaller than or equal to one. Figure 2.1 presents a UDG, where each notebook represents a vertex and the edge exists if the Euclidean distance between any two netbooks is less than or equal to one. In this example, three notebooks are enough to send a message to others.

Other application of MWCDS can be found in Wang et al. (2012),

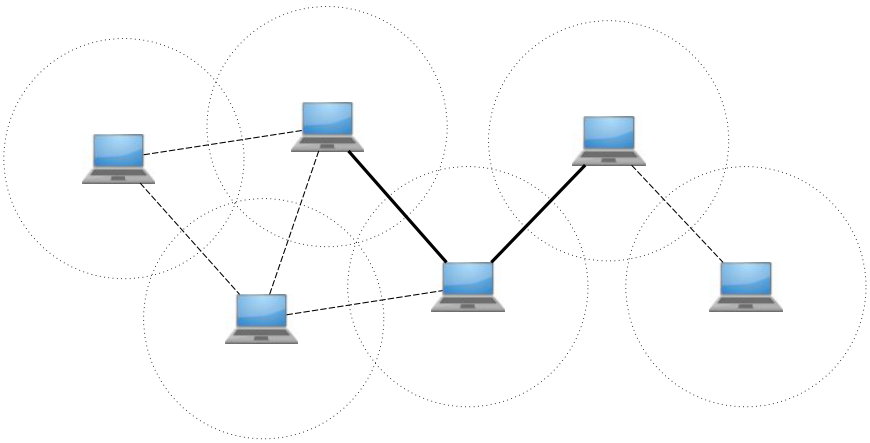


Figure 2.1: Example of the topology of a unit disk graph

focusing on the multi-document summarization, which compresses a given collection of documents into a summary of much smaller size by extracting either the main information or the information related to a query associated with the collection. A framework to solve these two problems by generation of pictorial and temporal storylines is proposed and the system is composed of the following steps: construction of a multi-view object graph; selection of a set of nodes using an approximation algorithm for the minimum-weight dominating set problem; and creation of a storyline by the use of a directed Steiner tree algorithm to connect them.

Zou et al. (2011) introduced an $(5+\epsilon)$-approximation for the MWCDS problem by using a node-weighted Steiner tree to interconnect all nodes of the MWDS on UDG. Dagdeviren et al. (2017) applied two population-based algorithms to the MWCDS problem: an hybrid genetic (HGA) algorithm and a population-based iterated greedy (PBIG) algorithm. Both methods are the first metaheuristics for the MWCDS problem in the literature, and were also applied to the MWDS problem, i.e., without constraints on the connectivity of the dominating set (Potluri & Singh, 2011; Bouamama et al., 2012).

A timeline is shown in Figure 2.2 in order to summarize the main studies cited previously for the three variants: MWDS, MCDS and MWDS. This timeline gathers papers encountered in the literature from of ear 2010 displaying the main method and authors. The proposed key method, marked in boldface, are selected to serve as support to our study on matheuristics, relatively to two main criteria: first, the resulting variants are the subject of a significant literature, including exact and heuristic methods; and second, benchmark instances are available for comparisons between methods. This timeline shows an increasing interest in the MDS problem last years.

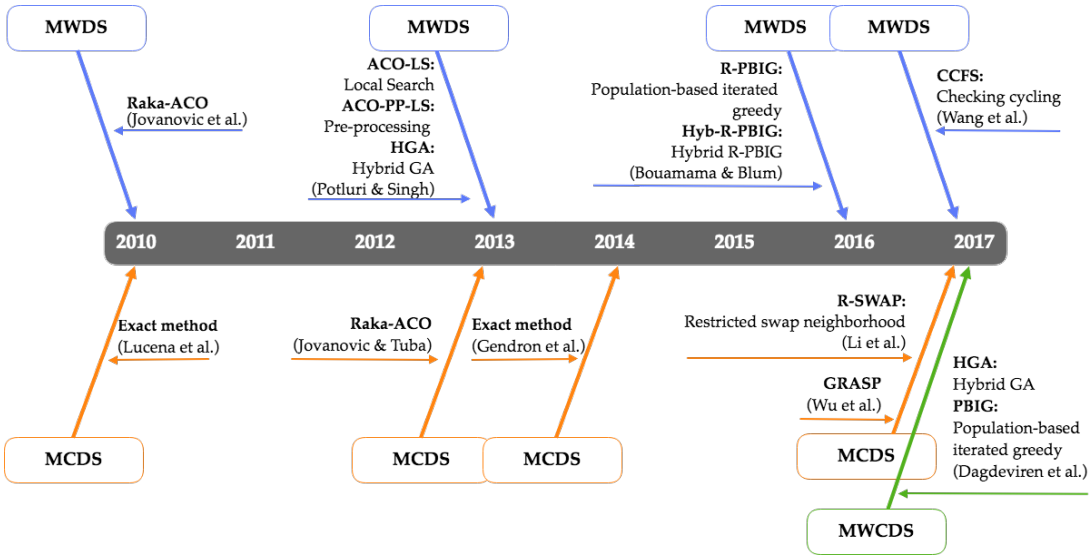


Figure 2.2: Timeline of some works for MWDS, MCDS and MWCDS problems

2.2.4 Independent Dominating Set

A vertex subset $\mathcal{S} \subseteq \mathcal{V}$ is a maximal independent set (MIS) if no two vertices in $\mathcal{S}$ are adjacent, that is, the subgraph $\mathcal{G}[\mathcal{S}]$ induced by $\mathcal{S}$ has no edges. An independent dominating set (IDS) in a graph is a set that is both dominating and independent and is maximal if no vertex can be added without violating independence. Equivalently, an independent dominating set is a maximal independent set.

Goddard & Henning (2013) presented a survey for the MIS problem and established key relationships between the independent domination number and others parameters, including the independence number and the chromatic number. Potluri & Singh (2013) experimented by adding a pre-processing step to the ACO algorithm to solve the MWDS problem. This pre-processing step generates different maximal independent sets (MIS) in order to reduce the computational time. Liu et al. (2015) show that the IDS problem is NP-complete on cubic bipartite graphs and present a fixed-parameter tractable (FFPT) algorithm for this problem.

2.2.5 Capacitated Dominating Set

The problem of finding a vertex subset of cardinality at most k that dominates all remaining vertices in a graph is the capacitated dominating set problem (CapMDS). CapMDS is a generalization of the MDS problem and aims to find a dominating set of minimum cardinality with the additional

constraint that the nodes dominated do not exceed the capacity of the dominating node. Cygan et al. (2011) and Liedloff et al. (2014) showed that this problem can be solved faster than $O(n^2)$ time.

Potluri & Singh (2012) presented a greedy heuristic and a couple of its variants to solve the CapMDS problem for UDG with high degree of connectivity and uniform capacity in general graphs.

2.3 Covering Codes

The covering code problem is a classic problem in coding theory. Carnielli (1985) presented the first relation between the covering code in Hamming-space and an optimization problem: the coloring problem. The use of metaheuristics to find the upper bounds for covering codes in Hamming space can be found in the works of Östergård (1997) and Mendes et al. (2010). Compression with distortion, data compression, decoding of errors and erasures (Koetter & Kschischang, 2007), football game (Linderoth et al., 2009), cellular telecommunications, among others, are areas in which covering codes have been applied.

The relation between covering codes and the dominating set problem, as well as their multiple applications have motivated extensive research. One remarkable feature is the wide use of links and tools from many fields of mathematics, computer science, and information theory. In particular, the covering problem in a Rosenbloom-Tsfasman (RT) space is the code induced by the Rosenbloom- Tsfasman metric, called the RT-covering problem. This problem is an extension of the classic covering problem in Hamming spaces.

From a theoretical point of view, Quistorff (2007) investigated the packing problem in RT-spaces, extending the classic function associated with the packing problem in Hamming spaces. Castoldi & Carmelo (2015) presented new lower and upper bounds for the covering code problem with an RT-metric, using a theoretical approaches. A study of the covering codes and the demonstration of their relation to the dominating set problem is presented in Chapter 5.

The classic covering code problem is based on the Hamming metric. The lottery problem is a classical application, which became one of the most famous problems in coding theory. The lottery problem is known also as football pool problem because in many countries there is a form of lottery where speculators must select the results of association football matches as being either a win for the home team, a win for the away team or a draw. On a lottery ticket, the bettor fill in n numbers, i.e., n numbers are drawn from a set of m numbers, such that $n < m$. The problem is the following: what is the minimum number

of tickets so that there is at least one ticket with at least p matching numbers? The study of this code is a challenge, and no optimal bound has been proven for this code. This is not only due to its combinatorial nature, but also to the intrinsic symmetry (Jans & Degraeve, 2008).

2.4 Other Applications of the MDS Problem

The applications of the MDS problem relate to the design and analysis of communication networks, bioinformatics, computational complexity, and algorithm design.

2.4.1 Biological

Recently, several studies have considered the MDS problem in biology (Wang et al., 2014; Wuchty, 2014; Vinayagam et al., 2016; Nacher & Akutsu, 2016). These studies seek the determination of a minimum set of driver proteins that are important for the control of the underlying protein-protein interaction (PPI) networks. Knowledge of the protein–protein interactions facilitates understanding pathogenesis and disease, which has been critical to advancements in vaccine research, therapeutic drug discovery and cell biology. Figure 2.3 shows an example of a multiplex *Human HIV-1 PPI network*, where the nodes represent proteins and edges, in the presence of a direct interaction among them.

2.4.2 Social Networks

A social network is a social structure made up of individuals (or organizations) called 'nodes', which are connected by one or more types of relationships, represented by 'links' (e.g., friendship, kinship, common interest, among others). In other words, in social networks there is a tendency for the behavior of connected users to match.

Wang et al. (2009) introduced the Positive Influence Dominating Set (PIDS) problem and analyzed its effect on a real online social network data set through simulations. This problem is defined as follows: given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and being $d(i)$ the degree of vertex i, a PIDS is a subset $\mathcal{D} \in \mathcal{V}$ such that any vertex $i \in \mathcal{V}$ is dominated by at least $\left\lceil \frac{d(i)}{2} \right\rceil$ vertices in $\mathcal{D}$. In other words, vertex i has at least $\left\lceil \frac{d(i)}{2} \right\rceil$ neighbors. Therefore, every vertex should have at least half of its neighbors in $\mathcal{D}$.

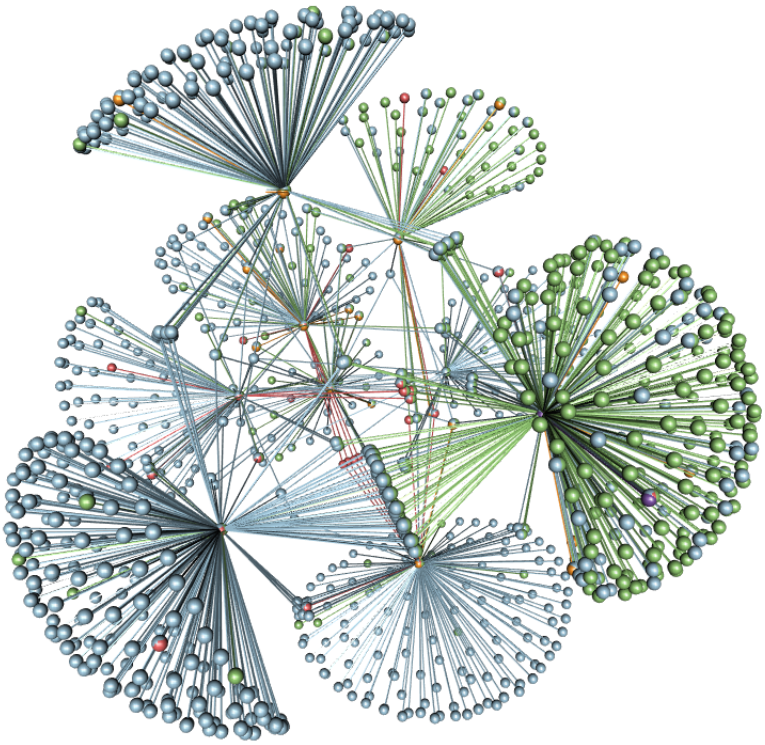


Figure 2.3: Example of protein-protein interaction network (reprinted from Wikimedia (2017)).

Many works have been published since Wang et al. (2009). For example, a greedy algorithm with an approximation ratio of the n-th harmonic number. was proposed in Wang et al. (2011). Dinh et al. (2012) showed an approximation for this problem and Kim et al. (2013) proposed an approach to identify effective leader groups of social networks based on the DS problem. Bonato et al. (2014) studied the domination number of on-line social networks is in random geometric graphs.

The existence of all these approaches shows that the development of new machine learning applications and social network analysis has accelerated the growth of the research on MDS in the recent years.

3
A matheuristic for the Minimum Weight Dominating Set Problem

In this Chapter, we introduce a tabu search matheuristic for the MWDS problem. The term "matheuristic" is generally used to refer to hybrid metaheuristics built upon some mathematical programming components (Maniezzo et al., 2009). The developed algorithm integrates a tabu search metaheuristic and the ILP model for the MWDS. The motivation behind this hybridization is to exploit the different optimization strategies in order to benefit from their synergy.

The proposed method, called HTS-DS, combines four successful strategies: an efficient neighborhood search, an adaptive penalty scheme to explore intermediate infeasible solutions, perturbation phases to promote exploration, and an intensification mechanism in the form of an integer programming (IP) solver.

Section 3.1 presents the mathematical formulation for the MWDS. Section 3.2 presents a study of constructive heuristics to build a greedy feasible solution. Section 3.3 describes the tabu search matheuristic. The hybrid algorithm can improve some solutions of the benchmark problem with up to 1000 vertices, and these results, as well as the sensitivity analysis of the procedures, are presented in the Section 3.4.

3.1
Mathematical Formulation

Let $\mathcal{W}$ be the set of positive weights represented by $\{w_1, w_2, \cdots, w_n\}$, i.e., each w_j corresponds to the weight to the vertex $j \in \mathcal{V}$. Since $N[i]$ is the closed neighborhood, the MWDS can be formulated as:

$$\min \quad \sum_{i \in V} w_i x_i \tag{3-1}$$

$$\text{s.t.} \quad \sum_{j \in N[i]} x_j \geq 1 \qquad \forall i \in V \tag{3-2}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{3-3}$$

The objective function (3-1) minimizes the total weight of the dominating

set. The remaining Equations (3-2) and (3-3) are similar to the formulations of the DS problem.

3.2 Greedy Constructive Heuristics

Greedy methods are algorithms that make a locally optimal choice at every step. Because previous choices are never revised in these methods, they are usually simple. They may not find the optimal solution, but may return good solutions and obtain some insights about the characteristics of the problem and instances. This can lead to a good quality solution in many problems. While in some problems a greedy strategy does not find a feasible solution, in the case of the DS problem, the constructive heuristics always work well, by definition of the algorithm, at obtaining a feasible solution for the problem.

Given $\mathcal{S}$ a subset of $\mathcal{V}$, we say that $\mathcal{S}$ is a feasible solution if this subset is a dominating set. In addition, the development of the procedures of our algorithm considers that the vertex i dominates itself by satisfying the condition of the closed neighborhood of the vertices. Even if a vertex is dominated, it can still belong to the set $\mathcal{S}$ by potentially dominating some vertices that still are not dominated.

We propose five constructive heuristics, which iteratively insert one vertex at a time in a partial solution. These heuristics differ in regard to the local greedy criterion to evaluate the cost of inserting a vertex in the solution. For every vertex $i \in \mathcal{V}$, we define $\Gamma(i)$ as the set of non-dominated vertices belonging to $N(i)$ (adjacent to i, or i itself). Let $W(i) = \sum_{k \in \Gamma(i)} w_k$ and, as defined above, $\Delta(i) = |\Gamma(i)|$, that is, the span of the vertex i. The strategies used to evaluate the cost of an insertion consist of the following cost value:

G_1: The span of the vertices.

$$i \leftarrow \text{argmax}_{j \in \mathcal{V} \setminus \mathcal{S}} \Delta(j) \tag{3-4}$$

G_2: The weight of the vertices.

$$i \leftarrow \text{argmax}_{j \in \mathcal{V} \setminus \mathcal{S}} \frac{1}{w_j} \tag{3-5}$$

G_3: Both the span and the weight of the vertex.

$$i \leftarrow \text{argmax}_{j \in \mathcal{V} \setminus \mathcal{S}} \frac{\Delta(j)}{w_j} \tag{3-6}$$

G_4: The sum of the weight of the non dominated vertices adjacent to the analyzed vertex.

$$i \leftarrow \text{argmax}_{j \in \mathcal{V} \setminus \mathcal{S}} \frac{W(j)}{w_j} \tag{3-7}$$

G_5: The three characteristics: the span and weight of the vertex, and the sum of the weight of the adjacent vertices.

$$i \leftarrow \text{argmax}_{j \in \mathcal{V} \setminus \mathcal{S}} \frac{\Delta(j) \times W(j)}{w_j} \tag{3-8}$$

The five greedy heuristics G_1, G_2, G_3, G_4 and G_5, presented by Algorithm 1, include iteratively a vertex $i \in V \setminus \mathcal{S}$ with highest value of the greedy cost. Starting with an empty set, at each step the algorithm adds that vertex to what yields the largest argument in the graph until a feasible solution is found. Ties (vertices with identical value) are broken randomly with uniform probability, and the process is iterated until a feasible solution is found.

Algorithm 1: Greedy constructive heuristics

Input : $G_k,\ k = 1, 2, 3, 4, 5$
Output: Feasible solution $\mathcal{S}$

```
1 begin
2     S ← ∅;                                   // dominating set
3     S_dom ← ∅;                               // dominated vertices set
4     while |S_dom| < |V| do
5         Add the best vertex i ∈ V\S, according to the greedy cost
          G_k, in the dominating set S;
6         Update the set S_dom adding the vertex i and its neighbors;
7     end
8 end
```

The experimental comparison of the proposed constructive heuristics are presented in Section 3.4. We choose the best two and embedded them to compose a procedure that is described in Section 3.3.3.

3.3 A Hybrid Tabu Search

This section presents the proposed hybrid algorithm that combines a tabu search metaheuristic with an integer programming solver. Figure 3.1 provides a general overview of the HTS-DS approach.

The algorithm starts from a random initial solution $\mathcal{S}_0$ (Phase 1 in the figure). This solution is subsequently improved by a combination of tabu

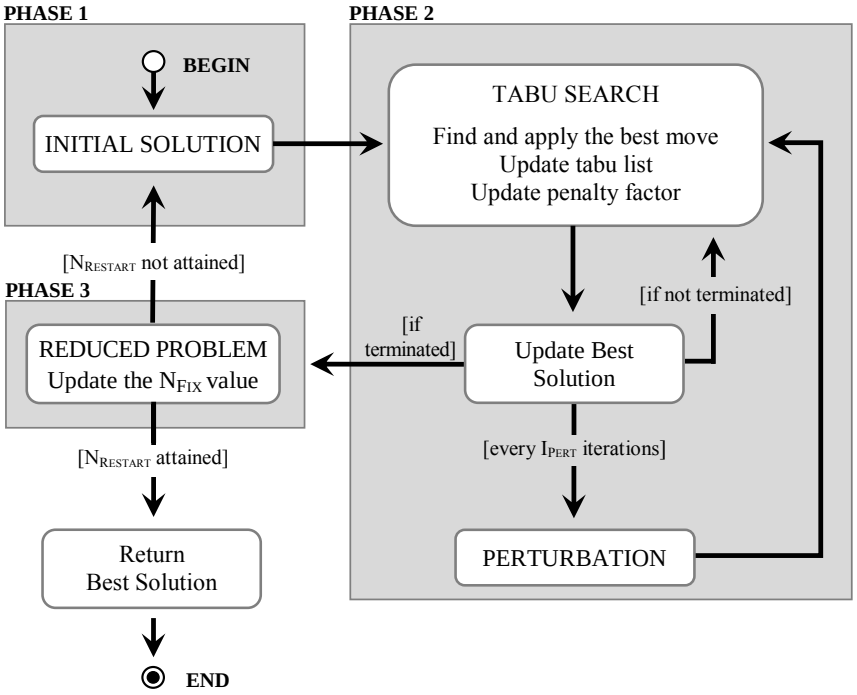


Figure 3.1: HTS-DS algorithm flowchart

search and perturbation mechanisms (PHASE 2), and serves to define a reduced problem which is solved using an integer programming solver (PHASE 3). This process is repeated N_{RESTART} times, and the best overall solution is returned. The methods into the (PHASE 2) terminates when I_{NI} successive iterations have been performed without improving the best solution, or when a iterations limit I_{MAX} is reached.

It is noteworthy that some penalized infeasible solutions, in which some vertices are not dominated, are possibly explored during the search. The objective function therefore plays an important role in the algorithm.

3.3.1 Penalized Objective Function

As illustrated in several previous studies (see, e.g., Glover & Hao, 2011; Vidal et al., 2015), an exploration of penalized infeasible solutions can help to diversify the search and prevent the method from being trapped in low-quality local optima. Therefore, HTS-DS relies on a penalized objective function that allows evaluating infeasible solutions with some non-dominated vertices. The cost of a solution $\mathcal{S}$ is calculated as:

$$f(\mathcal{S}) = W(\mathcal{S}) + \alpha \times w^{\text{MAX}} \times N^{\text{D}}(\mathcal{S}), \tag{3-9}$$

where α is the penalty factor at the current iteration of the search, w^{MAX} is the maximum weight of a vertex, and $N^{\text{D}}(\mathcal{S})$ is the number of non-dominated vertices in solution $\mathcal{S}$, and $W(\mathcal{S})$ is the total sum of weight of the dominating set $\mathcal{S}$. This way, the amount of penalty is directly proportional to the degree of infeasibility in order to promote a return to the feasible solution space.

An adequate choice of α is important for the search efficiency. In our preliminary experiments, a constant value was insufficient to help the transition to different regions of the search space. We therefore designed a *periodic ramp-up* strategy, in which α varies from a minimum level α^{MIN} to a maximum level α^{MAX} by steps of α^{STEP}, and then resumes to its minimum value and grows again. When α is small, the search will tend to remove vertices from the solution and lead to more non-dominated vertices. These vertices are subsequently covered again when the value of α increases. Due to this behavior, HTS-DS shares some common characteristics with ruin-and-recreate methods. With this analogy in mind, we set the parameter α^{STEP} to be inversely proportional to the number of vertices of the graph,

$$\alpha^{\text{STEP}} = \frac{\alpha^{\text{MAX}} - \alpha^{\text{MIN}}}{\beta|\mathcal{V}|} \tag{3-10}$$

where β is a parameter of the method which controls the number of steps between α^{MIN} and α^{MAX}.

Figure 3.2 illustrates the behavior of the parameter α in a small instance. Notice how the feasibility of the current solution directly depends on the value of α. The discovery of new best solutions, depicted with green diamonds in the graph, systematically occurs when α is a mid-range value. The incursion in the infeasible solution space, at each phase, allows the transition towards structurally different solutions.

3.3.2 Tabu Search

Now, we describe how to use a tabu search to solve a DS problem. The whole procedure includes a constructive function which is used to construct an initial solution. After that, a local search is called to improve the initial solution. The tabu list, the aspiration criteria, and the perturbation operator were used to complete our hybrid tabu search.

Initial Solution

The initial solution $\mathcal{S}_0$ is built by a random construction. Iteratively, the algorithm selects with uniform probability a random vertex which covers at

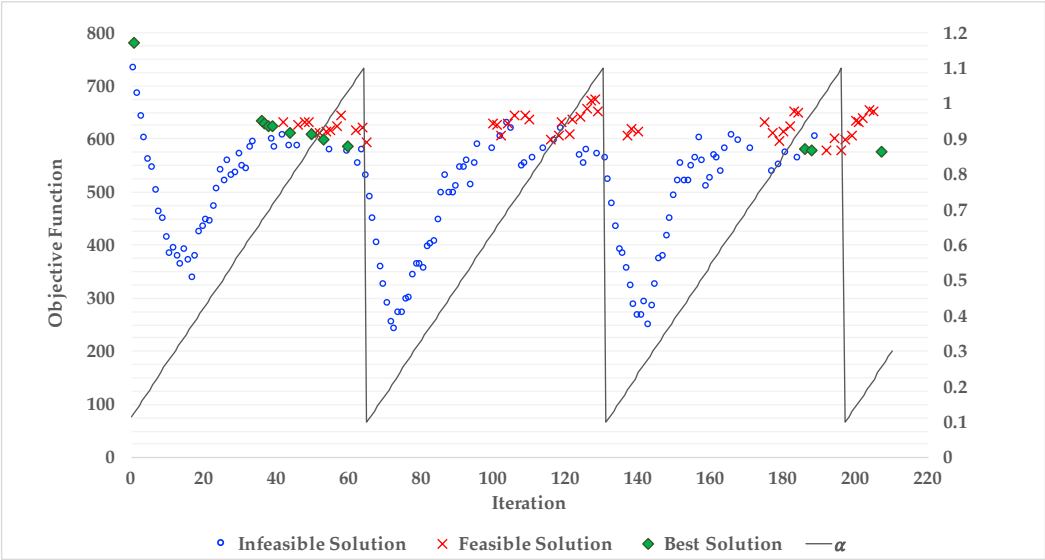


Figure 3.2: Evolution of the penalty parameter α during the search on a small instance (class SMPI – problem 50_50_0). Discovery of feasible, infeasible, and best known solutions.

least one non-dominated vertex, and inserts it in $\mathcal{S}_0$. The process stops when a dominating set is obtained.

Neighborhoods

The initial solution is then improved by a tabu search. The neighborhood of the solution $\mathcal{S}$, denoted by $\mathcal{N}(\mathcal{S})$, is based on three classical families of moves:

- ADD – adds a vertex i into the solution;
- DEL – removes a vertex j from the solution;
- SWAP – simultaneously adds a vertex i and remove a vertex j from the solution.

The movements depend on the problem and the representation of candidate solutions. In this case, we consider a list of size $|\mathcal{V}|$ in order to represent the solution. Each position is related to a vertex i and it will be active (one) if the vertex $i \in \mathcal{S}$, and otherwise zero.

Between the three possible movements, the algorithm chooses the one with the best improvement in the current solution. The moves are evaluated according to the objective function of Equation (3-9). At each iteration, the best *non-tabu* move from the entire neighborhood is applied. Remark that, depending on the status of the tabu memory, this move can be deteriorating, thus allowing escape from local minima.

These three neighborhoods differ in their size: ADD and DEL contain $\mathcal{O}(|\mathcal{V}|)$ moves while SWAP contains $\mathcal{O}(|\mathcal{V}|^2)$ moves. To better balance the search effort and improve the speed of the method, we apply *dynamic restrictions* to the neighborhood SWAP. This restriction works by evaluating first the ADD and DEL moves, ranking these moves relatively to their impact on the solution value, and restricting the search of the SWAP neighborhoods to the (i, j) pairs that belong to the top $\sqrt{|\mathcal{V}|}$ ADD and DEL moves. With this restriction, only $\mathcal{O}(|\mathcal{V}|)$ SWAP moves are evaluated.

To efficiently evaluate each move, we maintain for each vertex i a value $C(i)$ which represents how many times the vertex is dominated by another. Therefore, if $C(i) = 0$ then i is non-dominated. This allows to evaluate each move with a complexity proportional to the degree of the associated vertex (or vertices).

Tabu List

The short-term memory (tabu list) is essential to avoid cycling. In HTS-DS, this list has a fixed size of N_{TABU} and contains two types of labels: those that prohibit the insertion of a specific vertex, and those that prohibit the removal of a specific vertex. The update of the list is subject to the following rules:

- Whenever ADD(i) is applied, a label is added to prohibit the removal of i;
- Whenever DEL(j) or SWAP(i, j) is applied, a label is added to prohibit the insertion of j.

Remark that the tabu status associated to the SWAP move only prohibits the (re-)insertion of the leaving vertex j. Indeed, in preliminary analyses, we observed that symultaneausly prohibiting the removal of i overly constrains the search and slows down the progress towards high-quality solutions.

The tabu list stores the last iteration where a movement was made tabu. In this list, there is one element for each vertex of V. A movement M is tabu if the difference between its value in the tabu list and the number of the current iteration is larger than the tabu list size, usually referred as *tabu tenure*.

The tabu tenure is the number of iterations a certain elements is declared as tabu. The size of the tabu tenure depends on the number of vertices of the graph $\mathcal{G}$. Our method uses a static tabu tenure which the value of tabu duration is fixed throughout the search allowing the control of the search space.

Aspiration Criteria

As usually done in most tabu searches, we use an aspiration criterion, which revokes the tabu status of a move in the case where it leads to a new best solution. In fact, the aspiration criterion occurs when the chosen neighborhood leads to a good region of the solution space, i.e., it obtains the best solution in value. So it is desirable to go back to a visited solution s despite the fact that it is in tabu list T. This criterion is adopted in order to explore a new region. Given z, the value of the best current solution $\mathcal{S}$ encountered so far (i.e., $z = f(\mathcal{S})$) Therefore, when a solution $\mathcal{S}'$ of the neighborhood of the current solution ($\mathcal{S}' \in \mathcal{N}(\mathcal{S})$) belongs to T and meets the aspiration criterion ($A(\mathcal{S}') < z$), its tabu status is canceled and $\mathcal{S}'$ becomes a candidate during the selection of a best neighbor of $\mathcal{S}$. Here, z represents the best current solution $\mathcal{S}$ encountered so far.

Node Elimination

A solution $\mathcal{S}$ may contain redundant vertices. A vertex $i \in \mathcal{S}$ is redundant if all adjacent vertices and itself are dominated by other vertices from $\mathcal{S}$, i.e., a redundant vertex is a vertex which can be removed from the solution without increasing the number of non-dominated vertices.

After each move application, HTS-DS checks the possible existence of redundant vertices in a feasible solution. When such a situation occurs, a redundant vertex of maximum weight is removed. This process is iterated until no redundant vertex exists. Again, we used the value $C(j)$ to efficiently find the redundant vertices, since this value represents how many times the vertex is dominated by another. In this way, if $C(j) > 1$, for all vertex $j \notin \mathcal{S}$ adjacent to vertex $i \in \mathcal{S}$, then i is redundant. It allows evaluating the redundant vertex in the dominating set with a complexity proportional to its degree.

3.3.3 Perturbation Operator

After every I_{PERT} iterations of the tabu search, a perturbation mechanism is triggered in order to diversify the search further and reach different solutions. The perturbation is also based on the ruin-and-recreate strategy (Schrimpf et al., 2000). It works by removing $\lfloor \rho \times |\mathcal{S}_{\text{BEST}}| \rfloor$ vertices from the current best solution $\mathcal{S}_{\text{BEST}}$ of the tabu search, completing it with a greedy algorithm, and finally applying the node elimination procedure. The resulting solution is taken as new starting point for the search.

The greedy algorithm used for solution reconstruction works as follows. It is based on two greedy heuristics, presented in the previous section, more

specifically, the G_3 and G_4. heuristics. This algorithm will include, with 25% chance each, a vertex $i \in \mathcal{S}$ with the value:

- Corresponding to the G_3 heuristic – Equation (3-6)

  1. highest value of $\Delta(i)/w_i$,
  2. second-highest value of $\Delta(i)/w_i$

- Corresponding to the G_4 heuristic – Equation (3-7)

  3. highest value of $W(i)/w_i$,
  4. second-highest value of $W(i)/w_i$.

Again, ties (vertices with identical value) are broken randomly with uniform probability, and the process is iterated until a feasible solution is found.

The diversification phase is a perturbation mechanism in the current solution and aims to escape from a local optimum. It is applied once a fixed number of not improved iterations I_{PERT} is reached. From a general perspective, the perturbation has two processes: the destruction and construction phase, illustrated in Algorithm 2. First, the destruction phase removes some random elements from the current solution, resulting in a partial solution, and second, the construction phase applies a constructive heuristic to the partial solution to obtain another feasible solution. Finally, the method invokes the Node Elimination procedure, as described in Section 3.3.2.

Algorithm 2: Perturbation-DS

Input : $\mathcal{S}_{\text{BEST}}, \rho$
Output: $\mathcal{S}$

```
1 begin
2 |  S_P ← RemoveVertex(S_BEST, ρ);          // partial solution
3 |  S ← Reconstruct(S_P) using G_3 and G_4 heuristics.
4 |  S ← NodeElimination(S)
5 end
```

Termination Criteria

The tabu search (Phase 2 – Figure 3.1) terminates as soon as either I_{NI} consecutive iterations (moves) have been performed without improvement of the best solution, or a total of I_{MAX} iterations have been performed.

3.3.4 Resolution of a reduced integer problem

After the tabu search, an intensification mechanism is applied, in the form of an integer programming (IP) solver, to solve partial problems in which a majority of variables are fixed, keeping free those of the current best known solution and those associated to *promising* vertices. As in the Construct, Merge, Solve & Adapt (CMSA) approach described in Blum et al. (2016), the set of promising vertices is selected based on the search history. Moreover, the size of the subproblem is adapted to exploit the capabilities of the solver as efficiently as possible.

The HTS-DS algorithm constructs in Phase 3 a reduced problem based on the information of the best solutions and the past search history, and solves it once with an integer programming solver using the formulation of Equations (2-1) - (2-3) with the aim of finding a new best solution. In the reduced problem, most of the decision variables are fixed, and only a smaller group of "free" variables remain (possible choices of vertices for the dominating set).

The set size of "free" variables, N_{FREE} in the model, correspond to the $|\mathcal{S}_{\text{BEST}}|$ vertices used in the best solution of the tabu search, along with the $N_{\text{FREQUENT}} = \max\{0, |N_{\text{FREE}}| - |\mathcal{S}_{\text{BEST}}|\}$ *most frequent* vertices in the incumbent solution in the past search history. To identify the most frequent vertices, HTS-DS counts the total number of iterations $I_{\text{TOTAL}}(i)$ for which each vertex i belongs to the current solution. Such a counter can be efficiently implemented by storing the index of the current iteration when i is included, and only updating the counter with the adequate increment when i is removed.

The resulting formulation is solved by the integer programming solver subject to a time limit T_{MAX}. Initially, the size parameter N_{FREE} is set to 50. Subsequently, in order to fully exploit the capabilities of the IP solver, the parameter N_{FREE} is adapted from one general iteration of HTS-DS to the next. At the end of the resolution, there are three possible outcomes for the IP solver.

- *Case 1a) The solver finds an optimal solution, and* $N_{\text{FREE}} = |V|$. An optimal solution has been found for the MWDS problem, HTS-DS terminates.
- *Case 1b) The solver finds an optimal solution of the reduced problem, and* $N_{\text{FREE}} < |V|$. In this case, the IP solver may be able to address a larger problem in the next iteration within the allowed time, and therefore the parameter N_{FREE} is increased to $\min\{|V|, 2 \times N_{\text{FREE}}\}$.
- *Case 2) The solution is not proven optimal or no solution is produced.* In this case, the IP solver has been used beyond its capabilities, and N_{FREE}

is reduced to $N_{\text{FREE}}/2$.

The best overall solution is stored to be returned at the end of HTS-DS.

3.4 Computational Experiments

In this section, we explain the calibration of the method's parameter, and then we compare our heuristic to state-of-the-art methods. Finally, we present a sensitivity analyses of the procedures used in the hybrid algorithm.

Extensive computational experiments were conducted in order to evaluate the performance of the method relative to previous algorithms, and examine the relative contribution of each of its main components. HTS-DS was implemented in C++ using CPLEX 12.7, in which was used for the resolution of the integer linear program with their default parameter configurations. All tests were conducted on a single thread of an I7-5820K 3.3GHz processor.

3.4.1 Problem Instances

We rely on a total of 1060 problem instances originally proposed in Jovanovic et al. (2010). These instances are divided into two types (T1 and T2) and two classes (SMPI and LPI). The SMPI class includes 320 small and medium instances with 50 to 250 vertices and 50 to 5000 edges, and the LPI class includes 210 larger instances counting between 300 and 1000 vertices, with up to 20000 edges. In the instances of type T1, the vertices' weights are uniformly distributed in the interval [20, 70], while in the instances of type T2, the weight of each vertex i is randomly chosen in $[1, \delta(i)^2]$, where $\delta(i)$ is the degree of i. Ten instances were generated for each problem dimension. Therefore, in the subsequent sections, all results will be aggregated (averaged) by group of ten instances with identical characteristics.

3.4.2 Parameters Calibration

Three main parameters of HTS-DS have a strong influence on the search: the size of the tabu list N_{TABU}, the strength of the perturbation operator ρ, and the control parameters for the penalty $(\alpha_{\text{MIN}}, \alpha_{\text{MAX}}, \alpha_{\text{STEP}})$. These parameters were calibrated through preliminary experiments by varying one parameter at a time until reaching a "local optimum" in terms of parameter configuration. Then, from the final parameter setting obtained, we performed a sensitivity analysis to examine the effect of a variation of each parameter. This analysis

will be reported in Section 3.4.5 along with other results measuring the contribution of the main search components.

Finally, the parameters ($N_{\text{RESTART}}, I_{\text{MAX}}, I_{\text{NI}}, I_{\text{PERT}}, T_{\text{MAX}}$) control the number of iterations and search time of each component. Changing these parameters leads to different trade-offs between solution quality and computational effort. Therefore, for a fair experimental analysis, their values were selected to obtain solutions in a CPU time which is comparable or smaller to that of previous algorithms. The final parameter values are presented in Table 3.1.

Table 3.1: Parameter configuration of HTS-DS

Parameter	**Symbol**	**Value**
Number of restarts	N_{RESTART}	10
Maximum number of iterations of TS	I_{MAX}	20000
Maximum number of iterations without improvement of TS	I_{NI}	10000
Frequency of the perturbation	I_{PERT}	100
Perturbation strength	ρ	0.2
Size of the tabu list	N_{TABU}	12
Time limit for the IP solver	T_{MAX}	1 sec
Penalty factor	α^{MIN}	0.1
	α^{MAX}	1.1
	β	1.3

3.4.3 Results for the constructive heuristics

We tested the greedy constructive heuristics proposed in the Section 3.2. Full results on the instances are displayed in Tables 3.2, 3.3, 3.4, and 3.5. Columns $|\mathcal{S}|$ and **Avg** give, respectively, the average size and the average total weight of the dominating set (over 10 runs).

It can be seen from the tables that G_3 obtains the best results among these greedy heuristics. The G_4 and G_5 heuristics performed similarly, but for the Type 2 instances, G_4 performed better. Thus, in order to compose the constructive heuristics in our proposed algorithm, we used G_3 and G_4 heuristics, defined by the Equations (3-4) and (3-7), respectively, in the perturbation operator.

Table 3.2: Comparison of the greedy heuristics for small and medium Type I instances

		$\mathbf{G_1}$		$\mathbf{G_2}$		$\mathbf{G_3}$		$\mathbf{G_4}$		$\mathbf{G_5}$	
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**
50	50	20.0	683.1	24.9	707.0	20.3	588.0	20.0	**578.3**	20.0	583.9
	100	13.4	473.4	21.2	570.3	14.5	407.4	14.4	410.5	13.6	**405.9**
	250	6.8	239.0	12.7	300.8	7.8	198.8	7.8	197.2	6.8	**195.2**
	500	3.9	139.2	6.7	144.1	4.4	103.3	4.5	105.5	4.1	**99.9**
	750	2.8	96.6	4.3	89.4	3.3	72.0	3.4	72.9	3.0	**67.9**
	1000	2.0	71.5	3.1	63.5	2.3	47.9	2.3	48.1	2.1	**44.8**
100	100	39.7	1365.1	50.1	1445.6	40.9	1173.8	40.4	1168.0	40.2	**1165.7**
	250	23.3	795.9	37.4	977.4	24.5	673.0	24.5	676.7	23.2	**668.7**
	500	14.0	475.9	25.4	605.9	15.3	**395.8**	15.5	397.3	14.3	406.8
	750	10.2	352.1	20.2	460.1	11.6	**286.1**	12.2	294.9	11.0	293.4
	1000	8.1	269.0	16.2	358.1	9.7	234.1	9.9	235.1	8.7	**230.2**
	2000	4.7	164.9	8.4	176.3	5.5	119.6	5.6	122.3	5.0	**115.3**
150	150	60.6	2065.0	75.1	2148.8	60.3	1730.1	59.9	1724.4	59.2	**1714.7**
	250	44.5	1518.9	66.5	1818.6	47.5	1353.0	46.9	1340.5	44.4	**1318.3**
	500	28.8	995.6	48.7	1221.7	31.1	830.8	31.0	**830.4**	29.7	846.7
	750	20.8	705.1	40.4	970.4	23.5	617.5	23.6	611.4	21.7	**609.8**
	1000	17.0	590.1	33.5	775.4	19.3	496.1	19.3	488.7	17.4	**493.1**
	2000	10.0	338.1	20.3	440.1	11.2	266.5	11.6	**271.5**	10.8	276.6
	3000	7.1	237.6	13.7	288.0	8.3	**186.1**	8.4	191.1	7.7	**187.0**
200	250	71.6	2449.6	96.2	2704.6	73.8	2111.9	73.1	**2095.1**	71.2	2109.0
	500	45.9	1575.4	73.8	1925.5	50.9	1403.5	49.6	**1380.8**	47.1	1390.7
	750	34.5	1191.7	60.7	1507.7	37.7	**1019.1**	38.1	1020.5	35.5	1039.4
	1000	28.1	987.2	51.9	1244.1	31.5	813.4	31.0	**809.6**	28.5	819.8
	2000	16.7	586.2	33.1	734.2	19.4	473.9	18.9	**467.5**	17.6	485.6
	3000	12.3	434.5	24.1	517.2	14.2	**331.8**	14.3	334.8	13.1	334.9
250	250	101.6	3524.3	125.0	3565.3	102.6	2933.0	101.4	2906.0	100.4	**2891.3**
	500	67.3	2339.7	103.7	2785.2	71.0	**2011.9**	72.6	2040.9	67.9	2012.1
	750	52.4	1843.5	87.4	2240.2	56.4	1545.0	55.8	1533.8	51.9	**1510.5**
	1000	41.7	1457.2	75.3	1857.6	45.8	1230.7	46.0	1238.8	42.4	**1225.8**
	2000	24.6	858.2	49.6	1130.1	28.1	707.2	28.6	715.8	25.3	**705.4**
	3000	17.9	613.5	37.7	829.4	20.8	**499.8**	20.9	500.4	19.4	521.1
	5000	12.7	442.9	23.7	500.5	14.4	**323.6**	14.5	328.4	13.1	325.9
Avg		**27.0**	**933.8**	**42.8**	**1097.0**	**29.0**	**787.0**	**28.9**	**785.5**	**27.4**	**784.2**

Table 3.3: Comparison of the greedy heuristics for large Type I instances

		$\mathbf{G_1}$		$\mathbf{G_2}$		$\mathbf{G_3}$		$\mathbf{G_4}$		$\mathbf{G_5}$	
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**
300	300	120.2	4173.9	149.1	4262.8	122.5	3502.0	121.6	3481.5	120.5	**3465.7**
	500	89.0	3065.8	131.1	3602.8	95.5	2716.1	94.7	**2697.9**	90.4	2707.3
	750	69.1	2368.4	113.0	2965.2	75.6	2106.3	75.8	2105.4	70.6	**2070.8**
	1000	57.4	1948.3	100.0	2531.9	61.1	**1678.0**	61.9	1688.0	57.8	1692.2
	2000	33.6	1149.7	64.6	1503.6	37.0	**950.8**	37.9	965.6	35.4	988.2
	3000	25.0	860.2	49.5	1104.8	28.3	**693.7**	28.7	701.0	26.2	707.3
	5000	16.8	588.1	34.2	734.8	19.7	**456.7**	19.6	459.8	18.1	472.6
500	500	200.6	6914.0	249.3	7132.5	203.0	5835.5	202.9	5820.5	201.3	**5798.1**
	1000	133.0	4623.3	206.6	5577.8	142.3	4037.4	142.7	4039.8	134.9	**4032.6**
	2000	82.7	2846.4	149.1	3698.2	92.1	2468.3	92.3	2484.9	84.2	**2452.2**
	5000	42.5	1453.5	83.9	1879.3	47.7	**1171.1**	47.8	1177.1	42.9	1193.5
	10000	24.8	881.0	49.9	1060.9	28.4	**656.1**	29.0	670.5	26.3	666.3
800	1000	278.9	9661.7	380.2	10704.5	296.3	8518.9	294.4	8472.1	282.9	**8402.6**
	2000	186.8	6461.9	303.1	7980.5	201.3	5620.0	201.6	5639.7	189.4	**5602.4**
	5000	95.3	3338.2	186.6	4406.8	105.6	**2735.5**	106.4	2739.9	98.6	2796.8
	10000	56.3	1954.2	115.5	2543.2	64.1	**1553.0**	65.8	1590.6	58.8	1593.5
1000	1000	399.7	13793.6	496.4	14147.2	410.1	11714.0	408.5	11666.3	405.8	**11635.5**
	5000	141.4	4882.7	267.2	6469.6	157.9	**4140.4**	157.1	4168.5	145.8	4187.8
	10000	84.5	2953.2	169.8	3811.4	96.2	**2369.0**	96.1	2379.8	88.4	2410.8
	15000	60.6	2137.2	128.5	2792.9	71.0	**1702.9**	71.4	1704.0	65.1	1711.1
	20000	49.4	1701.2	102.8	2189.9	56.4	**1322.9**	58.1	1351.3	52.7	1367.8
Avg		**107.0**	**3702.7**	**168.1**	**4338.1**	**114.9**	**3140.4**	**115.0**	**3143.1**	**109.3**	**3140.7**

Table 3.4: Comparison of the greedy heuristics for small and medium Type II instances

		$\mathbf{G_1}$		$\mathbf{G_2}$		$\mathbf{G_3}$		$\mathbf{G_4}$		$\mathbf{G_5}$	
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**
50	50	20.0	103.4	24.6	75.9	21.4	**64.5**	21.3	64.9	20.8	65.8
	100	13.4	234.5	21.8	121.6	18.3	99.4	18.0	100.1	16.1	**97.0**
	250	6.8	701.4	13.6	214.3	11.2	166.8	10.9	**165.4**	9.7	167.9
	500	3.9	1358.5	7.7	261.8	6.3	**201.1**	6.3	205.5	5.6	205.0
	750	2.8	1702.4	4.7	215.5	4.2	178.1	4.2	178.1	4.1	**175.3**
	1000	2.0	2074.8	3.0	192.2	2.8	167.5	2.7	**162.8**	2.7	**162.8**
100	100	39.7	198.9	50.1	162.7	43.1	133.5	42.6	132.3	40.9	**131.2**
	250	23.3	610.3	39.5	284.7	32.7	230.8	31.6	**229.2**	28.9	235.8
	500	14.0	1251.5	27.8	443.2	22.0	343.2	21.4	**340.8**	18.6	355.8
	750	10.2	1950.3	21.3	557.5	17.4	440.9	17.0	**437.8**	14.3	459.5
	1000	8.1	2478.6	17.0	641.8	13.1	480.1	12.7	**470.3**	11.7	486.2
	2000	4.7	5470.8	9.2	810.4	7.6	632.4	7.4	**615.4**	6.5	635.4
150	150	60.6	297.4	74.9	240.7	63.6	**197.3**	63.3	198.3	61.1	198.4
	250	44.5	531.3	67.1	310.7	56.6	**254.9**	55.8	255.4	51.6	263.4
	500	28.8	1212.0	52.8	504.7	41.0	378.9	40.7	**376.8**	36.2	388.2
	750	20.8	1844.5	42.3	659.4	33.0	**502.0**	33.2	513.5	29.3	529.2
	1000	17.0	2592.5	35.7	834.9	27.3	**613.6**	27.3	622.8	23.9	620.3
	2000	10.0	5298.7	20.5	1064.3	16.1	804.1	16.4	833.4	14.6	**776.8**
	3000	7.1	6799.7	14.4	1201.1	11.3	**880.4**	11.4	888.5	10.6	908.4
200	250	71.6	520.0	95.8	350.5	82.3	296.9	82.1	297.4	77.3	**292.5**
	500	45.9	1224.0	77.1	524.3	64.2	**427.0**	63.4	**427.0**	58.6	435.2
	750	34.5	1939.8	64.8	717.3	51.6	562.8	51.1	560.5	45.7	567.9
	1000	28.1	2727.1	55.3	861.9	44.1	673.0	43.8	668.1	37.0	677.5
	2000	16.7	5666.4	34.1	1254.0	27.8	**961.8**	28.5	1004.7	25.2	995.7
	3000	12.3	8627.7	25.8	1526.3	20.5	1164.2	20.2	1140.8	18.2	**1126.3**
250	250	101.6	514.2	123.8	394.2	107.2	331.8	104.4	327.4	101.6	**325.4**
	500	67.3	1218.8	107.0	602.8	88.4	**482.2**	87.1	485.7	79.2	495.3
	750	52.4	1930.9	92.9	795.5	75.4	**634.6**	73.9	635.4	65.7	661.2
	1000	41.7	2545.1	79.3	949.8	65.0	751.7	64.9	**749.7**	55.9	788.5
	2000	24.6	5138.4	51.9	1490.0	41.2	**1151.6**	41.1	1154.3	35.7	1187.4
	3000	17.9	7794.5	39.2	1916.2	30.7	**1433.9**	30.8	1438.8	27.0	1467.1
	5000	12.7	14437.1	25.9	2365.0	19.4	**1628.7**	20.3	1741.1	17.9	1656.1
Avg		**27.0**	**2843.6**	**44.4**	**704.5**	**36.5**	**539.7**	**36.1**	**544.4**	**32.9**	**548.1**

Table 3.5: Comparison of the greedy heuristics for large Type II instances

		$\mathbf{G_1}$		$\mathbf{G_2}$		$\mathbf{G_3}$		$\mathbf{G_4}$		$\mathbf{G_5}$	
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**	$\|\mathcal{S}\|$	**Avg**
300	300	120.2	612.8	149.5	481.5	129.1	403.1	126.1	398.0	122.1	**393.4**
	500	89.0	1130.5	134.4	624.8	112.3	**512.9**	110.3	516.0	102.9	538.7
	750	69.1	1786.8	118.2	848.2	96.5	**674.4**	96.3	680.1	86.4	697.6
	1000	57.4	2437.1	106.3	1057.6	84.3	**816.0**	84.8	828.7	73.7	844.5
	2000	33.6	5282.5	69.8	1627.7	54.0	**1207.6**	54.6	1240.6	48.8	1248.9
	3000	25.0	7907.5	51.7	2050.8	40.1	**1510.1**	41.1	1555.3	35.8	1576.1
	5000	16.8	14027.7	33.3	2448.3	26.9	**1862.5**	27.8	1941.1	25.1	1869.2
500	500	200.6	1003.3	249.5	805.6	215.4	678.5	210.9	668.5	203.5	**658.4**
	1000	133.0	2362.9	217.0	1237.3	178.2	**986.2**	175.2	987.4	158.9	1012.1
	2000	82.7	5024.8	159.8	1979.8	126.5	1525.0	124.5	**1508.1**	110.6	1554.1
	5000	42.5	13505.1	89.1	3517.4	69.4	**2656.9**	70.0	2668.1	60.9	2729.9
	10000	24.8	30543.8	52.4	4961.0	40.5	**3630.6**	41.2	3723.3	36.9	3704.0
800	1000	278.9	2166.0	385.4	1447.5	327.6	1200.1	323.0	**1193.4**	306.0	1198.6
	2000	186.8	4904.6	319.7	2284.1	259.1	**1807.2**	257.3	1811.7	229.1	1853.5
	5000	95.3	13295.6	196.3	4367.9	150.8	**3226.7**	153.1	3321.5	132.7	3343.2
	10000	56.3	28192.7	118.5	6288.7	92.3	**4657.0**	92.5	4725.3	82.5	4810.9
1000	1000	399.7	2007.9	496.4	1590.4	431.2	1346.8	424.2	1336.9	411.0	**1323.5**
	5000	141.4	12989.2	283.8	4675.0	221.6	**3552.5**	222.0	3596.4	194.0	3674.6
	10000	84.5	27885.5	180.0	7279.9	139.9	**5393.4**	140.7	5432.6	121.9	5541.9
	15000	60.6	43466.1	131.8	8836.4	105.0	**6738.6**	105.8	6857.0	90.8	7030.1
	20000	49.4	57295.7	107.0	10297.0	83.9	**7602.0**	85.8	7785.6	74.4	7824.7
Avg		**107.0**	**13229.9**	**173.8**	**3271.8**	**142.1**	**2475.6**	**141.3**	**2513.1**	**129.0**	**2544.2**

3.4.4 Performance of the proposed algorithm

HTS-DS was run ten times with different random seeds on each instance. We will compare its results with that of the recent state-of-the-art algorithms for the MWDS listed in Table 3.6. This table also indicates the CPU model used by each study, along with the associated time scaling factor (based on *Passmark* benchmark) representing the ratio between its speed and the speed of our processor. In the remainder of this section, the time values reported by previous studies will be multiplied by the associated factors, in order to account for CPU differences and conduct a fair comparison.

Table 3.6: List of methods considered in the experiments, and CPU model information

Acronym	Description	CPU	Factor
RAKA	Ant colony approach of Jovanovic et al. (2010)	Not available	–
HGA	Hybrid genetic algorithm of Potluri & Singh (2013)	Not available	–
ACO-LS	Hybrid ACO with local search of Potluri & Singh (2013)	Not available	–
ACO-PP-LS	Hybrid ACO with pre-processing of Potluri & Singh (2013)	Not available	–
R-PBIG	Iterated greedy algorithm of Bouamama & Blum (2016)	Xeon 5670 2.93GHz	0.67
Hyb-R-PBIG	Hybrid algorithm of Bouamama & Blum (2016)	Xeon 5670 2.93GHz	0.67
CC^{2}FS	Configuration checking algorithm of Wang et al. (2017)	I5-3470 3.2GHz	0.95
HTS-DS	Hybrid tabu search	I7-5820K 3.3GHz	1.00

Tables 3.7–3.10 now compare the results of all methods. Each table corresponds to a different instance class (SMPI and LPI) and type (T1 and T2), and each row corresponds to a group of ten instances with identical characteristics. From left to right, the columns report the characteristics of the instances, the average solution quality and CPU time of previous algorithms, as well as the best, average solution quality and average CPU time of HTS-DS. The two rightmost columns also report the gap of HTS-DS' best (Gap_B) and average (Gap_A) solutions, relative to the best known solutions (BKS) in the literature. Let z be the solution value found by HTS-DS and z_{BKS} be the best known solution value. The percentage gap is computed as $\text{Gap}(\%) = 100 \times (z - z_{\text{BKS}})/z_{\text{BKS}}$. Finally, the best method is highlighted in boldface for each row, and the last line of the table presents some metrics (time and solution quality) averaged over all instances.

For the benchmark instances of type T1 and class SMPI, the HTS-DS retrieves all known optimal solution values (known for 17 instances in total), and even finds new best known solutions for 16% of the instances. The algorithm produces solutions of consistently high quality, with an average gap from the previous BKS of -0.003%, meaning that the average solution quality of HTS-DS is better than the best solution ever found in all prior studies in

Table 3.7: Type T1, Class SMPI – Comparison of HTS-DS with recent state-of-the-art algorithms

		RAKA	**HGA**		**ACO-LS**		**ACO-PP-LS**		**R-PBIG**		**Hyb-R-PBIG**		**CC^{2}FS**		**HTS-DS**				
$\|V\|$	$\|E\|$	**Avg**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
50	50	539.8	**531.3**	2.7	**531.3**	1.2	532.6	1.1	**531.3**	0.3	**531.3**	0.3	**531.3**	47.5	**531.3**	**531.3**	0.1	0	0
	100	391.9	371.2	2.6	371.2	1.0	371.5	0.9	371.1	0.5	**370.9**	0.5	**370.9**	47.5	**370.9**	**370.9**	0.1	0	0
	250	195.3	175.7	2.3	176.0	0.6	**175.7**	0.6	**175.7**	0.9	**175.7**	0.9	**175.7**	47.5	**175.7**	**175.7**	0.1	0	0
	500	112.8	94.9	2.0	94.9	0.5	95.2	0.5	95.0	1.5	**94.9**	1.5	**94.9**	47.5	**94.9**	**94.9**	0.1	0	0
	750	69.0	63.1	2.1	63.1	0.3	63.2	0.3	63.8	1.7	**63.1**	1.7	63.3	47.5	**63.1**	**63.1**	0.1	0	0
	1000	44.7	41.5	2.1	41.5	0.3	**41.5**	0.3	**41.5**	2.0	**41.5**	2.0	**41.5**	47.5	**41.5**	**41.5**	0.0	0	0
100	100	1087.2	1081.3	9.6	1066.9	4.3	1065.4	3.9	1061.9	0.9	**1061.0**	0.9	**1061.0**	47.5	**1061.0**	**1061.0**	0.2	0	0
	250	698.7	626.3	8.4	627.3	3.1	627.4	2.8	619.3	1.5	**618.9**	1.5	**618.9**	47.5	**618.9**	**618.9**	0.5	0	0
	500	442.8	358.3	5.8	362.5	2.3	363.2	2.0	356.5	2.0	**355.6**	2.0	**355.6**	47.5	**355.6**	**355.6**	0.6	0	0
	750	313.7	261.2	4.8	263.5	2.0	265.0	1.9	256.5	2.5	**255.8**	2.5	**255.8**	47.5	**255.8**	**255.8**	0.7	0	0
	1000	247.8	205.6	4.9	209.2	1.7	208.8	1.7	203.6	2.9	**203.6**	2.9	**203.6**	47.5	**203.6**	**203.6**	0.9	0	0
	2000	125.9	108.2	4.9	108.1	1.2	108.4	1.2	108.0	4.2	**107.4**	4.2	**107.4**	47.5	**107.4**	**107.4**	0.6	0	0
150	150	1630.1	1607.0	23.4	1582.8	9.9	1585.2	8.8	1582.5	1.6	**1580.5**	1.6	**1580.5**	47.5	**1580.5**	**1580.5**	0.5	0	0
	250	1317.7	1238.6	20.8	1237.2	8.6	1238.3	7.5	1219.5	2.2	**1218.2**	2.2	**1218.2**	47.5	**1218.2**	**1218.2**	0.6	0	0
	500	899.9	763.0	15.3	767.7	6.1	768.6	5.5	745.0	2.9	**744.6**	2.9	**744.6**	47.5	**744.6**	**744.6**	3.0	0	0
	750	674.4	558.5	12.2	565.0	5.0	562.8	4.5	548.3	3.5	546.8	3.5	**546.1**	47.5	**546.1**	**546.1**	7.0	0	0
	1000	540.7	438.7	11.8	446.8	4.5	448.3	4.0	433.6	4.0	433.1	4.0	432.9	47.5	**432.8**	**432.8**	8.1	**-0.023**	**-0.023**
	2000	293.1	245.7	9.2	259.4	3.3	255.6	3.3	241.5	5.8	**241.8**	5.8	**240.8**	47.5	**240.8**	**240.8**	7.8	0	0
	3000	204.7	169.2	8.3	173.4	3.0	175.2	2.8	168.4	7.4	167.8	7.4	**166.9**	47.5	**166.9**	**166.9**	7.5	0	0
200	250	2039.2	1962.1	41.7	1934.3	17.7	1927.0	15.3	1914.6	3.1	**1909.7**	3.1	1910.4	47.5	**1909.7**	**1909.7**	0.6	0	0
	500	1389.4	1266.3	33.4	1259.7	13.2	1260.8	11.5	1235.3	4.2	1234.0	4.2	**1232.8**	47.5	**1232.8**	**1232.8**	2.0	0	0
	750	1096.2	939.8	28.1	938.7	10.6	940.1	9.2	914.9	5.0	913.8	5.0	**911.2**	47.5	**911.2**	**911.2**	7.8	0	0
	1000	869.9	747.8	24.9	751.2	9.0	753.7	8.0	725.2	5.5	726.0	5.5	724.0	47.5	**723.5**	**723.5**	7.8	**-0.069**	**-0.069**
	2000	524.1	432.9	15.2	440.2	6.5	444.7	6.0	414.8	7.3	414.7	7.3	**412.7**	47.5	**412.7**	412.9	8.0	0	0.048
	3000	385.7	308.5	14.4	309.9	5.7	315.2	5.4	294.2	9.5	296.0	9.5	**292.8**	47.5	**292.8**	**292.8**	7.2	0	0
250	250	–	2703.4	72.7	2655.4	32.5	2655.4	28.4	2653.7	4.0	**2633.0**	4.0	2633.4	47.5	**2633.0**	**2633.0**	1.1	0	0
	500	–	1878.8	59.9	1850.3	25.1	1847.9	21.9	1812.6	5.8	1806.1	5.8	**1805.9**	47.5	**1805.9**	**1805.9**	2.4	0	0
	750	–	1421.1	55.0	1405.2	20.0	1405.5	17.4	1368.6	6.4	1366.9	6.4	1362.2	47.5	**1361.9**	**1361.9**	8.2	**-0.022**	**-0.022**
	1000	–	1143.4	47.3	1127.1	17.3	1122.9	15.2	1097.1	7.3	1092.8	7.3	1091.1	47.5	**1089.9**	1090.1	7.9	**-0.110**	**-0.092**
	2000	–	656.6	26.1	672.8	11.6	676.4	10.7	642.1	9.4	624.3	9.4	621.9	47.5	**621.6**	621.9	8.4	**-0.048**	0
	3000	–	469.3	23.0	474.1	9.5	476.3	9.0	451.5	12.0	452.5	12.0	**447.9**	47.5	448.0	448.0	7.6	0.022	0.022
	5000	–	300.5	21.8	310.4	8.4	308.7	8.1	291.5	17.0	293.1	17.0	**289.5**	47.5	**289.5**	289.6	7.8	0	0.035
	Avg	–	724.1	19.3	721.2	7.7	721.5	6.9	707.5	4.5	705.5	4.5	705.5	47.5	704.4	704.5	3.6	**-0.008**	**-0.003**

Table 3.8: Type T1, Class LPI – Comparison of HTS-DS with recent state-of-the-art algorithms

		RAKA	**HGA**		**ACO-LS**		**ACO-PP-LS**		**R-PBIG**		**Hyb-R-PBIG**		**CC^{2}FS**		**HTS-DS**				
$\|V\|$	$\|E\|$	**Avg**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
300	300	–	3255.2	116.3	3198.5	49.2	3205.9	42.2	3189.3	5.1	**3175.4**	5.1	3178.6	47.5	**3175.4**	**3175.4**	1.3	0	0
	500	–	2509.8	109.0	2479.2	41.0	2473.3	35.7	2446.9	6.8	**2435.6**	6.8	2438.1	47.5	**2435.6**	**2435.6**	1.6	0	0
	750	–	1933.9	93.2	1903.3	34.0	1913.9	30.0	1869.6	8.0	1856.8	8.0	1854.6	47.5	**1853.8**	1853.9	7.5	**-0.043**	**-0.038**
	1000	–	1560.1	80.1	1552.5	28.2	1555.8	24.5	1503.4	8.8	1498.6	8.8	1495.0	47.5	**1494.0**	1494.1	8.7	**-0.067**	**-0.060**
	2000	–	909.6	47.6	916.8	18.8	916.5	17.0	872.5	11.3	870.1	11.3	862.5	47.5	**862.4**	862.4	8.9	**-0.012**	**-0.012**
	3000	–	654.9	37.3	667.8	15.4	670.7	14.3	629.0	14.3	628.5	14.3	624.3	47.5	**624.1**	624.4	9.5	**-0.032**	0.016
	5000	–	428.3	27.4	437.4	12.5	435.9	12.0	409.4	20.0	410.0	20.0	**406.1**	47.5	**406.1**	406.3	8.1	0	0.049
500	500	5476.3	5498.3	412.3	5398.3	180.3	5387.7	156.0	5378.4	14.1	**5304.7**	14.1	5305.7	47.5	**5304.7**	**5304.7**	2.7	0	0
	1000	4069.8	3798.6	359.8	3714.8	143.7	3698.3	116.4	3642.2	19.5	**3607.3**	19.5	3607.8	47.5	3607.6	3608.6	10.4	0.008	0.036
	2000	2627.5	2338.2	219.2	2277.6	90.3	2275.9	81.7	2203.9	24.2	2197.2	24.2	2181.0	47.5	**2176.8**	2177.8	10.8	**-0.193**	**-0.147**
	5000	1398.5	1122.7	114.4	1115.3	51.3	1110.2	45.7	1055.9	37.5	1052.1	37.5	1043.3	47.5	**1042.3**	1044.2	10.9	**-0.096**	0.086
	10000	825.7	641.1	64.0	652.8	36.9	650.9	35.8	596.3	51.1	597.5	51.1	**587.2**	47.5	**587.2**	**587.2**	9.9	0	0
800	1000	8098.9	8017.7	1459.5	8117.6	769.8	8068.0	709.6	7768.6	45.5	**7655.0**	45.5	7663.4	950.0	**7655.0**	**7655.0**	5.2	0	0
	2000	5739.9	5318.7	1094.3	5389.9	572.6	5389.6	554.8	5037.9	55.8	5002.8	55.8	**4982.1**	950.0	4987.3	4991.7	14.3	0.104	0.193
	5000	3116.5	2633.4	551.5	2616.0	263.5	2607.9	237.3	2465.4	82.1	2469.2	82.1	2441.2	950.0	**2432.6**	2435.8	14.2	**-0.352**	**-0.221**
	10000	1923	1547.7	246.1	1525.7	147.5	1535.3	145.4	1420.0	114.6	1414.8	114.6	1395.6	950.0	**1393.7**	1395.1	13.9	**-0.136**	**-0.036**
1000	1000	10924.4	11095.2	2829.6	11035.5	1320.7	11022.9	1189.9	10825.2	64.9	**10574.4**	64.9	10585.3	950.0	**10574.4**	**10574.4**	8.7	0	0
	5000	4662.7	3996.6	1152.1	4012.0	627.7	4029.8	600.8	3693.1	123.4	3699.7	123.4	3671.8	950.0	**3656.6**	3662.7	15.7	**-0.414**	**-0.248**
	10000	2890.3	2334.7	566.2	2314.9	308.8	2306.6	289.9	2140.3	170.8	2138.1	170.8	2109.0	950.0	**2099.8**	2102.2	15.9	**-0.436**	**-0.322**
	15000	2164.3	1687.5	356.3	1656.3	227.7	1657.4	211.0	1549.1	188.9	1548	188.9	1521.5	950.0	**1519.7**	1521.9	16.2	**-0.118**	0.026
	20000	1734.3	1337.2	251.1	1312.8	204.9	1315.8	200.3	1219.0	194.2	1216.9	194.2	1203.6	950.0	**1200.9**	1205.6	17.9	**-0.224**	0.166
	Avg	–	2981.9	485.1	2966.4	245.0	2963.3	226.2	2853.1	60.0	2826.3	60.0	2817.0	434.3	2813.8	2815.2	10.1	**-0.096**	**-0.024**

Table 3.9: Type T2, Class SMPI – Comparison of HTS-DS with recent state-of-the-art algorithms

		RAKA	**HGA**		**ACO-LS**		**ACO-PP-LS**		**R-PBIG**		**Hyb-R-PBIG**		**CC^{2}FS**		**HTS-DS**				
$\|V\|$	$\|E\|$	**Avg**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
50	50	62.3	**60.8**	3.8	**60.8**	1.2	**60.8**	1.2	**60.8**	0.3	**60.8**	0.3	**60.8**	47.5	**60.8**	**60.8**	0.1	0	0
	100	98.4	**90.3**	3.8	**90.3**	1.1	**90.3**	1.1	**90.3**	0.5	**90.3**	0.5	**90.3**	47.5	**90.3**	**90.3**	0.1	0	0
	250	202.4	**146.7**	3.4	**146.7**	0.8	**146.7**	0.7	**146.7**	0.9	**146.7**	0.9	**146.7**	47.5	**146.7**	**146.7**	0.0	0	0
	500	312.9	**179.9**	2.8	**179.9**	0.6	**179.9**	0.6	**179.9**	1.4	**179.9**	1.4	**179.9**	47.5	**179.9**	**179.9**	0.0	0	0
	750	386.3	**171.1**	2.5	**171.1**	0.5	**171.1**	0.4	**171.1**	1.6	**171.1**	1.6	**171.1**	47.5	**171.1**	**171.1**	0.0	0	0
	1000	–	**146.5**	1.9	**146.5**	0.3	**146.5**	0.2	**146.5**	1.9	**146.5**	1.9	**146.5**	47.5	**146.5**	**146.5**	0.1	0	0
100	100	126.5	124.5	11.0	123.6	4.2	**123.5**	4.0	**123.5**	0.8	**123.5**	0.8	**123.5**	47.5	**123.5**	**123.5**	0.2	0	0
	250	236.6	211.4	11.0	210.2	3.5	210.4	3.2	**209.2**	1.4	**209.2**	1.4	**209.2**	47.5	**209.2**	**209.2**	0.1	0	0
	500	404.8	306.0	10.2	307.8	2.7	308.4	2.5	**305.7**	1.9	**305.7**	1.9	**305.7**	47.5	**305.7**	**305.7**	0.1	0	0
	750	615.1	385.3	8.6	385.7	2.4	386.3	2.3	386.9	2.3	**384.5**	2.3	**384.5**	47.5	**384.5**	**384.5**	0.2	0	0
	1000	697.3	429.1	8.5	430.3	2.0	430.3	1.9	**427.3**	2.8	**427.3**	2.8	**427.3**	47.5	**427.3**	**427.3**	0.2	0	0
	2000	1193.9	550.6	7.8	558.8	1.6	559.8	1.5	552.7	4.2	**550.6**	4.2	**550.6**	47.5	**550.6**	**550.6**	0.3	0	0
150	150	190.1	186.0	29.5	184.7	9.1	184.9	8.6	**184.5**	1.4	**184.5**	1.4	**184.5**	47.5	**184.5**	**184.5**	0.5	0	0
	250	253.9	234.9	29.1	233.2	8.6	233.4	7.8	**232.8**	2.1	**232.8**	2.1	**232.8**	47.5	**232.8**	**232.8**	0.3	0	0
	500	443.2	350.0	24.5	351.9	7.0	351.9	6.2	349.7	2.9	**349.5**	2.9	**349.5**	47.5	**349.5**	**349.5**	0.3	0	0
	750	623.3	455.8	21.8	456.9	5.9	454.7	5.6	**452.4**	3.6	**452.4**	3.6	**452.4**	47.5	**452.4**	**452.4**	0.4	0	0
	1000	825.3	547.5	19.7	551.4	5.5	549.0	5.1	547.8	4.0	**547.2**	4.0	**547.2**	47.5	**547.2**	**547.2**	0.4	0	0
	2000	1436.4	720.1	17.5	725.7	4.4	725.7	4.1	**720.1**	5.6	**720.1**	5.6	**720.1**	47.5	**720.1**	**720.1**	0.5	0	0
	3000	1751.9	792.6	15.4	794.0	3.6	806.2	3.6	793.2	7.8	**792.4**	7.8	**792.4**	47.5	**792.4**	**792.4**	0.8	0	0
200	250	293.2	275.1	53.9	272.6	17.1	272.6	15.6	**271.7**	2.9	**271.7**	2.9	**271.7**	47.5	**271.7**	**271.7**	0.5	0	0
	500	456.5	390.7	49.9	388.6	14.6	388.4	12.8	386.8	4.1	**386.7**	4.1	**386.7**	47.5	**386.7**	**386.7**	0.4	0	0
	750	657.9	507.0	45.9	501.7	12.5	501.4	11.1	497.1	4.8	**497.1**	4.8	**497.1**	47.5	**497.1**	**497.1**	0.4	0	0
	1000	829.2	601.1	37.3	605.9	11.2	605.8	10.2	596.8	5.7	**596.8**	5.7	**596.8**	47.5	**596.8**	**596.8**	0.5	0	0
	2000	1626	893.5	31.9	891.0	8.9	892.9	8.1	884.6	7.6	**884.6**	7.6	**884.6**	47.5	**884.6**	**884.6**	0.6	0	0
	3000	2210.3	1021.3	29.9	1027.0	7.2	1034.4	7.0	1019.2	9.4	**1019.2**	9.4	**1019.2**	47.5	**1019.2**	**1019.2**	1.0	0	0
250	250	–	310.1	88.0	306.5	28.0	306.7	25.5	**306**	3.3	**306.0**	3.3	306.1	47.5	**306.0**	**306.0**	1.1	0	0
	500	–	444.0	89.1	443.8	25.4	443.2	22.8	441.0	5.5	**440.7**	5.5	**440.7**	47.5	**440.7**	**440.7**	0.7	0	0
	750	–	578.2	77.1	573.1	23.0	575.9	20.6	567.9	6.7	**567.4**	6.7	**567.4**	47.5	**567.4**	**567.4**	0.7	0	0
	1000	–	672.8	74.2	671.8	20.8	675.1	19.0	669.2	7.6	**668.6**	7.6	**668.6**	47.5	**668.6**	**668.6**	0.7	0	0
	2000	–	1030.8	56.8	1033.9	15.9	1031.5	15.1	1009.5	9.7	**1007.0**	9.7	**1007.0**	47.5	**1007.0**	**1007.0**	1.1	0	0
	3000	–	1262.0	49.0	1288.5	14.0	1277.0	13.5	1251.6	12.1	**1250.6**	12.1	**1250.6**	47.5	**1250.6**	**1250.6**	1.4	0	0
	5000	–	1480.9	46.9	1493.6	11.2	1520.1	10.9	**1464.2**	17.1	**1464.2**	17.1	**1464.2**	47.5	**1464.2**	**1464.2**	2.2	0	0
	Avg	–	486.1	30.1	487.7	8.6	488.9	7.9	482.7	4.5	482.4	4.5	482.4	47.5	482.4	482.4	0.5	0	0

Table 3.10: Type T2, Class LPI – Comparison of HTS-DS with recent state-of-the-art algorithms

		RAKA	**HGA**		**ACO-LS**		**ACO-PP-LS**		**R-PBIG**		**Hyb-R-PBIG**		**CC^{2}FS**		**HTS-DS**				
$\|V\|$	$\|E\|$	**Avg**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Avg**	**T(s)**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
300	300	–	375.6	142.3	371.1	43.0	371.1	39.2	**369.9**	4.2	**369.9**	4.2	**369.9**	47.5	**369.9**	**369.9**	1.3	0	0
	500	–	484.2	143.4	480.8	40.0	481.2	35.8	478.0	6.4	**477.8**	6.4	**477.8**	47.5	**477.8**	**477.8**	0.9	0	0
	750	–	623.8	124.5	621.6	36.9	618.3	32.9	613.6	7.8	**613.3**	7.8	**613.3**	47.5	**613.3**	**613.3**	0.9	0	0
	1000	–	751.1	113.0	744.9	33.7	743.5	30.5	738.3	9.0	**737.7**	9.0	737.9	47.5	**737.7**	**737.7**	1.0	0	0
	2000	–	1106.7	87.7	1111.6	24.3	1107.5	22.7	1094.6	11.7	**1093.8**	11.7	**1093.8**	47.5	**1093.8**	**1093.8**	1.1	0	0
	3000	–	1382.1	70.9	1422.8	23.9	1415.3	20.9	1358.5	14.0	**1358.5**	14.0	**1358.5**	47.5	**1358.5**	**1358.5**	1.5	0	0
	5000	–	1686.3	66.7	1712.1	18.2	1698.6	17.4	1683.2	19.8	**1682.7**	19.8	**1682.7**	47.5	**1682.7**	**1682.7**	2.8	0	0
500	500	651.2	632.9	522.0	627.5	149.1	627.3	135.4	624.2	11.9	**623.6**	11.9	**623.6**	47.5	**623.6**	**623.6**	2.8	0	0
	1000	1018.1	919.2	478.8	913.0	137.8	912.6	122.1	901.3	18.8	**899.6**	18.8	899.8	47.5	**899.6**	**899.6**	2.0	0	0
	2000	1871.8	1398.2	354.3	1384.9	109.9	1383.9	98.7	1364.4	24.9	**1362.2**	24.9	1363.3	47.5	**1362.2**	**1362.2**	2.4	0	0
	5000	4299.8	2393.2	217.5	2459.1	91.3	2468.8	90.7	2341.5	39.5	**2326.6**	39.5	**2333.7**	47.5	**2326.6**	**2326.6**	3.4	0	0
	10000	8543.5	3264.9	191.1	3377.9	60.1	3369.4	60.4	3216.1	53.9	**3211.5**	53.9	**3211.5**	47.5	**3211.5**	**3211.5**	8.5	0	0
800	1000	1171.2	1128.2	1810.5	1126.4	498.8	1125.1	444.9	1107.6	39.9	**1103.9**	39.9	1104.3	950.0	**1103.9**	**1103.9**	4.3	0	0
	2000	1938.7	1679.2	1560.5	1693.7	516.6	1697.9	474.7	1634.6	55.9	**1630.8**	55.9	1632.3	950.0	**1630.8**	**1630.8**	4.1	0	0
	5000	4439.0	3003.6	955.7	3121.9	413.7	3120.9	407.5	2884.8	86.0	**2876.1**	86.0	2878.5	950.0	**2876.1**	**2876.6**	10.8	0	0.017
	10000	8951.1	4268.1	653.6	4404.1	249.9	4447.9	249.7	4103.7	123.2	4103.3	123.2	4105.6	950.0	**4102.8**	4104.0	21.5	**-0.012**	0.017
1000	1000	1289.3	1265.2	3481.4	1259.3	931.2	1258.6	832.8	1243.6	54.2	**1237.5**	54.2	1237.7	950.0	**1237.5**	**1237.5**	8.8	0	0
	5000	4720.1	3320.1	1995.4	3411.6	832.6	3415.1	827.4	3195.7	131.3	**3172.9**	131.3	3178.7	950.0	3178.2	3180.0	16.6	0.167	0.224
	10000	9407.7	4947.5	1250.5	5129.1	546.5	5101.9	548.6	4722.4	184.0	**4704.8**	184.0	4711.8	950.0	4707.2	4709.9	31.2	0.051	0.108
	15000	14433.5	6267.6	977.7	6454.6	398.7	6470.6	398.9	5884.2	204.5	**5856.4**	204.5	5874.2	950.0	5859.8	5862.4	36.5	0.058	0.102
	20000	19172.6	7088.5	817.3	7297.4	322.8	7340.8	325.5	6678.0	213.9	6655.9	213.9	6662.1	950.0	**6655.1**	6657.9	55.6	**-0.012**	0.030
	Avg	–	2285.1	762.6	2339.3	260.9	2341.7	248.4	2201.8	62.6	2195.2	62.6	2197.7	434.3	2195.6	2196.2	10.4	0.012	0.024

the literature (issued from multiple algorithms, runs, and parameter settings). Overall, the average solution for 29 instances groups out of 32 have been strictly improved or matched during these experiments.

In a similar fashion, for the class LPI, the HTS-DS algorithm retrieves all known optimal solutions (50 in total), and produces new best solutions for 52% of the instances, with an average percentage gap of -0.096% relative to the BKS from previous literature. HTS-DS is also faster than existing algorithms, with an average CPU time of 3.6 and 10.1 seconds versus 47.5 and 434.3 seconds for CC^{2}FS method on the classes SMPI and LPI, respectively.

The same observations stand for the benchmark instances of type T2. For this type, the instances of class SMPI are easier to solve, and most recent method find the same solutions. In contrast, the class LPI allows to observe more significant differences between methods. Again, for this benchmark, HTS-DS retrieves very accurate solutions, with an average gap of 0.024% relative to the BKS, in a time which is significantly smaller than previous approaches.

Finally, since the scalability of the algorithm is essential for large scale applications, Figure 3.3 presents a more detailed analysis of the CPU time of the two main phases of the method (Tabu search and IP) as a function of the number of vertices $|V|$, for each instance type. To eliminate some instances with few edges which tend to be easy to solve, we restricted this analysis to the subset of instances such that $|E| \geq 3|V|$.

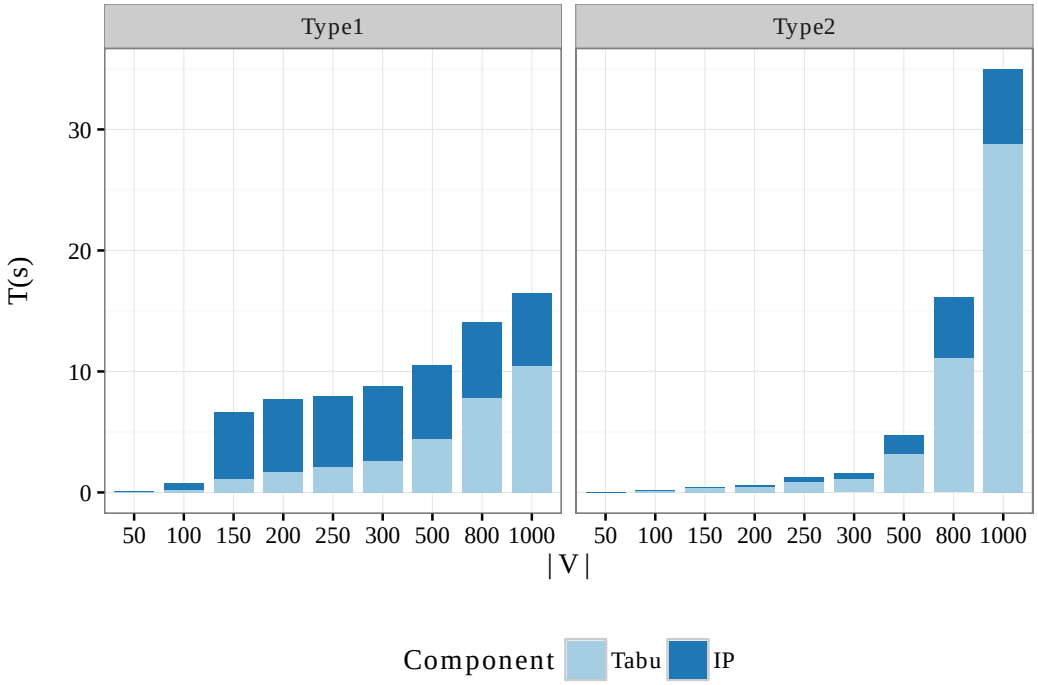


Figure 3.3: Computational time of each component of HTS-DS as a function of $|\mathcal{V}|$

From this figure, one can first observe that the average CPU time of HTS-DS remains below two seconds on a majority of instances. There are two situations where this computational time is exceeded. For the instances of type T1, the total time dedicated to the resolution of the reduced problems with the IP solver amounts to five to six seconds when $|V| \geq 150$ due to the increased difficulty of the mathematical models. For the instances of type T2, the resolution of the subproblems is faster, and the CPU time of the method only exceeds two seconds when $|V| \geq 500$.

Note that the time spent solving the subproblems cannot exceed an upper bound of 10 seconds overall, due to the time limit imposed on the IP solver (1 second) and the limited number of subproblem resolutions (up to $N_{\text{RESTART}} = 10$). Therefore, the scalability of the approach essentially depend on the efficiency of the tabu search phase, and more specifically, on the evaluation of the neighborhoods. For each instance type, we fitted the CPU time spent in the tabu search phase as a power law $f(|V|) = x|V|^y$ (by a least-squares regression of an affine function on the log-log graph). This time appears to grow as $\mathcal{O}(n^{1.81})$ on the instances of type 1, and $\mathcal{O}(n^{2.30})$ on the instances of type 2.

3.4.5 Sensitivity analyses

Finally, we performed a sensitivity analysis in order to evaluate the impact of each main component and parameter of the HTS-DS. Starting with the standard configuration described in Section 3.4.2, we modified one parameter and design choice at a time (OFAT approach) to evaluate its effect. The following configurations were considered:

Standard. Standard configuration described in Section 3.4.2.

A. No Reduced Problem. The reduced problem and IP solver is disabled.

B. No SWAP. The SWAP neighborhood is not used.

C. No Perturbation. No perturbation phase: $\rho = 0$.

D. ↑ Perturbation. Higher level of perturbation: $\rho = 0.4$.

E. ↓ Tabu. Shorter tabu tenure: $N_{\text{TABU}} = 5$.

F. ↑ Tabu. Longer tabu tenure: $N_{\text{TABU}} = 20$.

G. ↓ Beta. Shorter phases for penalty management: $\beta = 1.0$.

H. ↑ Beta. Longer phases for penalty management: $\beta = 1.5$.

All configurations were run ten times on each instance. Table 3.11 presents the gap of the best and average solutions over these runs, as well as the average time of each method configuration.

	Type T1			**Type T2**		
	$\mathbf{Gap_B}$	$\mathbf{Gap_A}$	**T(s)**	$\mathbf{Gap_B}$	$\mathbf{Gap_A}$	**T(s)**
Standard	**-0.04**	**0.00**	6.18	**0.00**	**0.01**	4.41
A. No Reduced Problem	0.06	0.13	2.74	0.09	0.12	3.56
B. No SWAP	0.07	0.14	5.10	0.01	0.02	3.36
C. No Perturbation	0.00	0.05	6.10	0.12	0.16	2.23
D. ↑ Perturbation	-0.02	0.01	6.56	0.01	0.01	8.05
E. ↓ Tabu	-0.01	0.02	6.13	0.01	0.02	4.94
F. ↑ Tabu	0.08	0.14	6.51	0.01	0.02	4.22
G. ↓ Beta	0.01	0.02	6.93	0.02	0.02	4.19
H. ↑ Beta	0.01	0.03	6.91	0.02	0.03	4.20

Table 3.11: Analysis of HTS-DS components

These experiments (configurations A–C) highlight the major contribution of the mathematical programming solver used for the solution of the reduced problems, the limited SWAP neighborhood as well as the perturbation operator. Without the subproblem solver, the average gap from the BKS rises up to 0.13% for type T1 and 0.12% for type T2, while the CPU time decreases by 55% seconds for type T1 and 19% for type T2. In a similar fashion, deactivating the SWAP or the perturbation operator translates into a significant decrease of solution quality for only moderate time gains.

The three main search parameters in charge of the perturbation rate, the tabu tenure and the management of the penalty factors also play an important role in the method. Increasing the perturbation rate to $\rho = 0.4$, for example, allows to better diversify the search but leads to a loss of information from the best solution, with a negative impact on the search (configuration D). Similarly, increasing the size of the tabu list is over-restrictive and hinders the progress towards high-quality solutions (configuration E), and decreasing it may increase the cycling probability (configuration F). Finally, the value of the parameter β has been chosen so as to find trade-off solutions at the frontier of feasibility without spending too much time per phase. Increasing or decreasing this parameter (configurations G and H) leads to solutions of lower quality.

3.5 Concluding Remarks

In this chapter, we proposed a matheuristic combining a tabu search with integer programing for the MWDS problem. The method exploits an efficient neighborhood search, an adaptive penalty scheme to explore intermediate infeasible solutions, perturbation mechanisms as well as additional intensification phases in which a reduced problem is solved by means of an integer programming solver. The size of this reduced problem is also adapted to fully exploit the capabilities of the solver.

Through extensive computational experiments, we demonstrated the good performance of HTS-DS, which outperformed previous algorithms on the classical benchmark instances of Jovanovic et al. (2010) with up to 1000 vertices. The method remains simple and scalable, with an observed CPU time growth in $\mathcal{O}(|V|^{1.81})$ and $\mathcal{O}(|V|^{2.30})$ as a function of the number of vertices for the instances of types T1 and T2, respectively, therefore making it suitable for large-scale applications. Finally, our sensitivity analyses demonstrated the essential contribution of each components of the search: the subproblem resolution, the proposed perturbation mechanisms, and the restricted SWAP neighborhood which were proposed.

4
Extension to solve the Weight Connected Dominating Set Problem

This chapter presents the HTS-DS algorithm, which can deal with both connected and weight variants of the DS problem. The algorithm consists of an extension of the one proposed for the MWDS presented in the previous chapter. We adapted the HTS-DS algorithm in order to impose the connectivity constraints inherent to the CDS problem. Computational experiments were carried out on instances with up to 1000 vertices and the best known solutions were improved for many instances, as well be shown in the next sections.

Section 4.1 defines the MWCDS problem. Section 4.2 presents a branch-and-cut algorithm for this problem. Section 4.3 describes the main simple changes and the additional procedures performed to adjust the HTS-DS to the MWCDS problem. Sections 4.4 and 4.5 show reports our computational experiments and analyses for MWCDS and MCDS problem, respectively. Finally, Section 4.6 concludes.

4.1
Problem Statement and Basic Concepts

The MWCDS problem involves finding a dominating set $\mathcal{S}$ of minimum total weight such that $\mathcal{G}[\mathcal{S}]$ is a connected subgraph. A dominating set is said to be connected if it fulfills the property by which any vertex $n \in \mathcal{S}$ can reach any other vertex $m \in \mathcal{S}$ by a path that stays entirely within $\mathcal{S}$. That is, $\mathcal{S}$ induces a connected subgraph of $\mathcal{G}$. The MCDS is a special case of the MWCDS in which all vertices have a weight of one. So, a minimum connected dominating set is the one with the minimum number of vertices.

4.2
A Branch-and-Cut Approach

This section presents a branch-and-cut (B&C) algorithm that was employed as a component of the proposed matheuristic. For each edge $(i, j) \in \mathcal{E}$, we define a binary variable y_{ij}, set to 1 if the edge is used to connect the subgraph $\mathcal{G}[\mathcal{S}]$, i.e., the vertices i and j belong to the set $\mathcal{S}$, and 0 otherwise. The set $\mathcal{E}(\mathcal{S})$ contains the edges of the subgraph $\mathcal{G}[\mathcal{S}]$. The MWCDS can be stated as follows.

$$\min \quad \sum_{i \in \mathcal{V}} w_i x_i \tag{4-1}$$

$$\text{s.t.} \quad \sum_{j \in N[i]} x_j \geq 1 \qquad \forall i \in \nu \tag{4-2}$$

$$\sum_{(i,j) \in \mathcal{E}} y_{ij} = \sum_{i \in \mathcal{V}} x_i - 1 \tag{4-3}$$

$$\sum_{(i,j) \in \mathcal{E}(\mathcal{S})} y_{ij} \leq |\mathcal{S}| - 1 \qquad \forall \mathcal{S} \subset \mathcal{V} \tag{4-4}$$

$$y_{ij} \leq x_j \qquad \forall (i,j) \in \mathcal{E} \tag{4-5}$$

$$y_{ij} \leq x_i \qquad \forall (i,j) \in \mathcal{E} \tag{4-6}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{E} \tag{4-7}$$

$$x_i \in \{0,1\} \qquad \forall i \in \mathcal{V} \tag{4-8}$$

The objective function (4-1) minimizes the size of the dominating set. Constraint (4-3) ensures that the subset $\mathcal{S}$ forms a tree. Constraints (4-4) are known as the *subtour elimination constraints*. Constraints (4-5) and (4-6) impose the connectivity of the subgraph induced by y_j. Constraints (4-7) and (4-8) define the domain of the decision variables.

To solve this formulation, we use a B&C algorithm which identifies violated subtour elimination constraints (4-4) as cutting planes. The separation problem is solved by identifying connected components first and including the associated violated cuts, and then using a max-flow algorithm within each connected component to find other possible cuts.

4.3 Methodology

To investigate the contribution of the proposed strategies for the DS problem, we derived an adaptation of the HTS-DS algorithm. The three major changes are in:

1. the neighborhood structure (the candidate list of the move) to be explored;
2. the rules (and associated memory structures) that define tabu status.
3. an additional mechanism of checking in order to find all the vertices that are articulation points in the graph induced for the solution. These mechanisms are described in the following

4.3.1 Articulation Points Checking

An *articulation point* is vertex of $\mathcal{G}$ whose removal results in a disconnected graph. Let $P(\mathcal{S})$ be the set of the articulation points of the graph induced by the dominating set $\mathcal{S}$ of a graph $\mathcal{G}$.

We can use a brute-force algorithm to find $P(\mathcal{S})$, which involves one-by-one removal of all vertices and checking if the removal of a vertex causes the graph to disconnected. For every vertex i, the following steps are taken: remove i from the graph; check if the graph remains connected by breath-first or depth-first search; and if the graph is disconnected, add i to the P list and i back to the graph. Nevertheless, the time complexity of this method is $\mathcal{O}(|\mathcal{V}| \cdot (|\mathcal{V}| + |\mathcal{E}|))$. To improve, an algorithm can be implemented in linear time $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$.

The algorithm involve a depth first search (DFS)-based approach as proposed by Hopcroft & Tarjan (1973). The DFS-tree root is an articulation point if and only if it has at least two children. In addition, a non-root vertex i of a DFS-tree is an articulation point of $\mathcal{G}$ if and only if has a child j such that there is no back edge from j or any descendant of i to a proper parent of j. Note that the leaves of a DFS-tree are never articulation points. This algorithm is invoked, at each iteration, to investigate the articulation points of the incumbent solution.

4.3.2 Penalized objective function

HTS-DS relies on a penalized objective function that allows evaluating infeasible solutions with some non-dominated vertices, but all the solutions explored during the search should form a connected graph. This encourages continuing to search for high-quality solutions even when the search is working with an infeasible solution. We used for the MWCDS problem the function of Equation (3-1). In this problem, we did not allow to violate the connectivity constraint since the feasibility in relation the connectivity of the graph cannot always be recovered efficiently.

If we consider all vertices have weight equal 1, then $W(\mathcal{S}) = |\mathcal{S}|$ and $w^{\text{MAX}} = 1$ regarding with the penalized function for the MWDS problem, in this way, the function of Equation (3-1) works for both variants. The cost of a solution $\mathcal{S}$ for the MCDS problem is calculated by:

$$f(\mathcal{S}) = |\mathcal{S}| + \alpha \times N^{\text{D}}(\mathcal{S}), \tag{4-9}$$

where α is the penalty factor, and $N^{\mathrm{D}}(\mathcal{S})$ is the number of non-dominated vertices in solution $\mathcal{S}$.

The behavior of the parameter α is the same as explained for the MWDS problem. The penalty parameters are adapted during the search in order to find a feasible solution. The further from feasibility the solution is with respect to the constraint that defines a dominating set, the higher the penalty term is for this violation the solution, and vice versa.

4.3.3 HTS-DS for the WCDS problem

The HTS-DS approach for the MWCDS maintains all the phases of the algorithm indicated in the Figure 3.1 of the previous chapter, but several components were adapted in order to improve the performance of the method.

Initial Solution

For the MWCDS problem, we consider as an initial solution the complete solution, i.e., all vertices are in the dominating set. Although all vertices belong to the dominating set, they are managed in a list randomly with uniform probability. This guarantees a non-deterministic search in the next phases since the explored moves follow the list order.

Neighborhoods

The initial solution is then improved in the tabu search. Three families of moves are considered.

- ADD – adds one vertex i into the solution, namely the vertex i that is adjacent to a vertex in the solution.
- DEL – removes a vertex j from the solution, if j is not an articulation point.
- SWAP – simultaneously adds a vertex i and removes a vertex j from the solution, with the above constraints for each move.

In the MWCDS problem, the tabu list keeps a fixed size of N_{TABU} and contains three types of labels: those that prohibit the insertion of a specific vertex, those that prohibit the removal of a specific vertex that leaves at the visited solution, and those that forbid the removal of articulation points in the current solution. Whenever a solution is found, the method search for any vertex i such that, i is an articulation point, and a label is added to prohibit the removal of i.

Therefore, our method guarantees the connectivity of all solutions. Note that it is important to use an efficient algorithm to find articulation points in a graph. As mentioned in the Section 4.3.1, this mechanism has the complexity time of $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$.

Perturbation Operator

Another adaptation of the method concerns the perturbation operator. The removal of a vertex is only allowed if this vertex is not a articulation point. Also, when we lead to the MCDS problem, we considered $w_i = 1$, so the Equations (3-4) and (3-7) return the same value the reconstruction of the solution, includes, with 50% chances each, a vertex i with:

1. the highest value of $\Delta(i)$;
2. or the second-highest value of $\Delta(i)$

Reduced Problem for the MWCDS

In Phase 3, the HTS-DS constructs a reduced problem, which exploits the best solution found in Phase 2 on a much smaller model. We study its properties depending on the choice of fixed variables. By guiding the resolution using a particular model, we are able to find good results from the fixing algorithm. The resolution of this sub-model is very flexible.

Besides the smaller group of "free" variables, which are the possible choices of vertices for the dominating set, if there is any articulation point, it will be fixed at 1. Because, obviously this vertex is a part of the solution, i.e. belongs to the dominating set. We follow the same parameters and strategy, presented in the Section 3.3.4, to fix and set 'free' the variables.

4.4 Computational Experiments

We implemented the HTS-DS algorithm in C++ for resolution of the integer linear problems, we used CPLEX 12.7. In addition, we conducted our experiments in a single thread of a computer with an I7-5820K 3.3GHz processor.

The instances originally proposed in Shyu et al. (2004) were considered. For these instances, two groups are defined, Type I and Type II instances, with respect to their weights, and divided into three groups with respect to node counts as small, moderate and large. Type I instances include vertices having weights uniformly distributed in the interval [20, 120], while in the of Type II instances, the weight of each vertex i is randomly chosen in $[1, \delta(i)^2]$,

where $\delta(i)$ is the degree of i. The sets of vertices in small, moderate and large problem instances are {10, 15, 20, 25}, {50, 100, 150, 200, 250, 300} and {500, 800, 1000}, respectively. Ten instances are generated for each group, but some of these instances produce disconnected graphs, the number of these graph will be presented in the table of results.

Our proposed algorithm was also tested in the instances suggested in Dagdeviren et al. (2017) with 10-1000 vertices, which are generated randomly. The instances include the small, moderate and large, with [10, 25], [50-250], and [250, 1000] vertices, respectively.

4.4.1 Parameters Calibration

The parameter values for the MWCDS problem were calibrated after preliminary experiments. The final parameter are presented in Table 4.1.

Table 4.1: Parameter configuration of HTS-DS

Parameter	Symbol	Value
Number of restarts	N_{RESTART}	10
Maximum number of iterations of TS	I_{MAX}	10000
Maximum number of iterations without improvement of TS	I_{NI}	5000
Frequency of the perturbation	I_{PERT}	100
Perturbation strength	ρ	0.3
Size of the tabu list	N_{TABU}	30
Time limit for the IP solver	T_{MAX}	1 sec
Penalty factor	α^{MIN}	0.1
	α^{MAX}	1.1
	β	2.0

4.4.2 Performance of the HTS-DS algorithm

Tables 4.2 - 4.6 computational results of ten runs of the HTS-DS algorithm. Some graphs are disconnected and therefore led to infeasible problem instances, so the ' #' column is the number of connected graphs in the 10 instances group for each problem class. The last row of these tables indicates the average for all the instances of the class that are evaluated, the average computation time, as well as the average gap.

We compared our method with the two population-based approaches proposed in Dagdeviren et al. (2017): a hybrid genetic algorithm (**HGA**) and a population-based iterated greedy (**PBIG**) algorithm. The results are

reported in the Tables 4.2 – 4.6 . The **CPLEX** column shows the results for the MWCDS problem using the formulation presented by Equations (4-2) - (4-8), along with Equation (3-1) as the objective function. The **T(s)** column reports the execution time of the method. We limited the time of the ILP solver to 3600 seconds. The value 'TL' in this column represents the instances in which the solver reached this time. The **Sol** and **GAP** columns present the solution and the gap returned by the ILP solver. The symbols '–' in the GAP column mean that the solver cannot return a solution in the limit time.

From left to right, the columns report the characteristics of the instances, the average solution quality and CPU time of previous algorithms, the results of the ILP solver, as well as the best, average solution quality and average CPU time of HTS-DS. The two rightmost columns also report the gap of HTS-DS' best $\mathbf{(Gap_B)}$ and average $\mathbf{(Gap_A)}$ solutions, relative to the best known solutions in the literature. Let z be the solution value found by HTS-DS and z_{BKS} be the best known solution value. Then the percentage gap is computed as $\text{Gap}(\%) = 100 \times (z - z_{\text{BKS}})/z_{\text{BKS}}$. Finally, the best method is highlighted in boldface for each row, and the last line of the table presents some metrics (time and solution quality) averaged over all instances.

Table 4.2: Type-I, Small Class – Comparison of HTS-DS with recent state-of-the-art algorithms

			HGA		**PBIG**		**CPLEX**				**HTS-DS**			
$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	#	**Avg**	**T(s)**	**Avg**	**T(s)**	**#S**	**Sol**	**T(s)**	**GAP**	**Best**	**Avg**	**T(s)**	$\mathbf{Gap_{B,A}}$
10	10	2	341.0	0.1	341.0	0.05	2	341.0	0	0	341.0	341.0	0.40	0
	20	10	157.2	0.1	157.2	0.03	10	157.2	0.01	0	157.2	157.2	0.36	0
	30	10	85.9	0.1	85.9	0.01	10	85.9	0.01	0	85.9	85.9	0.24	0
	40	10	45.0	0.1	45.0	0	10	45.0	0.01	0	45.0	45.0	0.14	0
15	20	3	409.7	0.2	409.7	0.05	3	409.7	0.03	0	409.7	409.7	0.57	0
	40	10	163.0	0.2	163.0	0.19	10	163.0	0.04	0	163.0	163.0	0.41	0
	60	10	104.3	0.2	104.3	0.06	10	104.3	0.02	0	104.3	104.3	0.33	0
	80	10	67.6	0.2	67.6	0.02	10	67.6	0.02	0	67.6	67.6	0.27	0
	100	10	25.7	0.2	25.7	0.01	10	25.7	0.01	0	25.7	25.7	0.21	0
20	20	–	–	–	–	–	–	–	–	–	–	–	–	–
	40	9	300.4	0.3	300.4	0.36	9	300.4	0.08	0	300.4	300.4	0.96	0
	60	10	213.7	0.3	213.7	0.34	10	213.7	0.10	0	213.7	213.7	0.74	0
	80	10	135.8	0.3	135.8	0.23	10	135.8	0.03	0	135.8	135.8	0.62	0
	100	10	111.4	0.3	111.4	0.21	10	111.4	0.03	0	111.4	111.4	0.50	0
	120	10	92.1	0.3	92.1	0.15	10	92.1	0.05	0	92.1	92.1	0.45	0
25	40	4	508.3	0.5	508.3	0.42	4	508.3	0.12	0	508.3	508.3	1.20	0
	80	10	228.0	0.4	228.0	0.52	10	228.0	0.14	0	228.0	228.0	0.89	0
	100	10	193.4	0.4	193.4	0.53	10	193.4	0.15	0	193.4	193.4	0.79	0
	150	10	120.0	0.4	120.0	0.44	10	120.0	0.09	0	120.0	120.0	0.58	0
	200	10	80.3	0.4	80.3	0.10	10	80.3	0.04	0	80.3	80.3	0.46	0
		Avg	**178.0**	0.2	**178.0**	0.19		**178.0**	0.05	0	**178.0**	**178.0**	0.53	0

Table 4.3: Type-I, Moderate Class – Comparison of HTS-DS with recent state-of-the-art algorithms

			HGA		**PBIG**		**CPLEX**				**HTS-DS**				
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	#	**Avg**	**T(s)**	**Avg**	**T(s)**	**#S**	**Sol**	**T(s)**	**GAP**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
50	50	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	100	4	900.0	0.16	900.0	1.73	4	**897.0**	409.53	0	**897.0**	897.0	2.23	**-0.33**	**-0.33**
	250	10	291.3	0.08	290.6	1.99	10	**288.7**	6.13	0	**288.7**	288.7	1.48	**-0.65**	**-0.65**
	500	10	141.6	0.09	**141.2**	2.20	10	**141.2**	1.29	0	**141.2**	141.2	1.15	**0**	**0**
	750	10	87.7	0.11	**87.6**	1.77	10	**87.6**	0.45	0	**87.6**	87.6	0.82	**0**	**0**
	1000	10	**51.4**	0.13	**51.4**	0.18	10	**51.4**	0.12	0	**51.4**	51.4	0.70	**0**	**0**
100	100	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	250	2	1408	0.71	**1362.5**	8.08	2	1376.5	1934.32	4.3	**1362.5**	1362.7	4.50	**0**	0.01
	500	10	687.4	0.48	641.9	9.35	10	**641.5**	1181.17	**0**	**641.5**	641.5	3.27	**-0.06**	**-0.06**
	750	10	439.8	0.41	**410.4**	9.22	10	**410.4**	795.01	**0**	**410.4**	410.4	2.83	**0**	**0**
	1000	10	336.9	0.46	308.6	9.76	10	**308.2**	379.98	**0**	**308.2**	308.2	2.67	**-0.13**	**-0.13**
	2000	10	144.4	0.94	141.9	10.83	10	**141.8**	79.93	**0**	**141.8**	141.8	1.78	**-0.07**	**-0.07**
150	150	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	250	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	500	6	1752	2.20	1661.3	25.43	1	2032.0	TL	22.6	**1647.2**	1648.9	5.93	**-0.85**	-0.75
	750	10	1070.2	1.54	1013.4	24.49	3	2109.3	TL	45.2	**1001.1**	1011.1	5.44	**-1.21**	-0.23
	1000	10	761.4	1.30	701.6	21.17	8	711.8	TL	24.4	**701.0**	701.6	4.88	**-0.09**	0
	2000	10	392.3	1.54	345	23.59	9	464.4	TL	23.8	**342.7**	343.5	3.65	**-0.67**	-0.43
	3000	10	256.2	2.52	234.1	22.78	0	-	-	-	**231.5**	**231.5**	3.09	**-1.11**	**-1.11**
200	250	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	500	4	3056.75	5.85	2966.3	48.89	0	-	-	-	**2931.8**	**2931.8**	10.98	**-1.16**	**-1.16**
	750	10	1917.5	4.88	1811.0	50.80	0	-	-	-	**1793.0**	**1793.0**	10.30	**-0.99**	**-0.99**
	1000	9	1451.4	3.69	1345.6	47.17	0	-	-	-	**1340.1**	**1340.1**	8.50	**-0.41**	**-0.41**
	2000	10	698	2.77	627.1	41.93	0	-	-	-	**624.0**	**624.0**	6.46	**-0.49**	**-0.49**
	3000	10	478	3.61	417.7	40.76	0	-	-	-	**416.8**	**416.8**	5.86	**-0.22**	**-0.22**
250	250	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	500	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	750	7	3068.1	10.40	2850.3	95.07	0	-	-	-	**2834.4**	**2834.4**	24.15	**-0.56**	**-0.56**
	1000	9	2227.8	8.49	2056.8	82.35	0	-	-	-	**2038.0**	2040.1	23.23	**-0.91**	-0.81
	2000	10	1091.9	5.07	976.3	75.06	0	-	-	-	**965.9**	967.3	18.56	**-1.07**	-0.92
	3000	10	752.2	5.28	656.9	71.77	1	896.0	TL	42.5	**650.4**	652.1	17.10	**-0.99**	-0.73
	5000	10	439.7	8.74	394.9	71.07	9	515.3	TL	44.1	**390.2**	391.5	15.36	**-1.19**	-0.86
300	300	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	500	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	750	2	4449.5	18.85	4293.5	156.64	0	-	-	-	**4273.0**	**4273.0**	27.20	**-0.48**	**-0.48**
	1000	9	3315.4	16.75	3111.0	156.31	0	-	-	-	**3068.8**	3068.9	26.53	-1.36	-1.35
	2000	10	1639.4	10.33	1472.6	125.22	0	-	-	-	**1440.3**	1440.5	22.41	-2.19	-2.18
	3000	10	1083.7	8.27	945.3	111.10	2	1219.0	TL	39.4	**936.9**	937.8	19.94	-0.89	-0.79
	5000	10	646.0	11.36	564.2	109.02	2	829.5	TL	51.0	**555.1**	558.1	17.76	-1.61	-1.08
		Avg	1167.9	4.57	1092.7	48.52		-	-	-	**1083.7**	1084.5	9.96	**-0.66**	-0.56

Table 4.4: Type-I, Large Class – Comparison of HTS-DS with recent state-of-the-art algorithms

			HGA		**PBIG**		**CPLEX**				**HTS-DS**				
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	#	**Avg**	**T(s)**	**Avg**	**T(s)**	**#S**	**Sol**	**T(s)**	**GAP**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
500	500	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	1000	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	2000	1	4579	74.50	4239	1166.97	0	–	–	–	**4201**	4221	51.30	**-0.90**	-0.42
	5000	1	1803	33.91	1576	452.28	0	–	–	–	**1565**	1569	42.40	**-0.70**	-0.44
	10000	1	922	45.10	868	398.50	0	–	–	–	**852**	**852**	36.90	**-1.84**	**-1.84**
800	500	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	1000	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	2000	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	5000	1	4740	246.81	4334	1922.66	0	–	–	–	**4215**	4235	71.30	**-2.75**	-2.28
	10000	1	2459	131.81	2081	1465.53	0	–	–	–	**2080**	**2080**	59.80	**-0.05**	**-0.05**
1000	1000	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	5000	1	7319	540.86	6762	4535.20	0	–	–	–	**6709**	6712	115.00	**-0.78**	-0.74
	10000	1	3596	296.01	3013	3121.87	0	–	–	–	**3053**	3059	76.60	1.33	1.53
	15000	1	2483	295.59	2178	2656.38	0	–	–	–	**2188**	2190	69.00	0.46	0.55
	20000	1	1895	308.91	1658	2394.79	0	–	–	–	**1617**	1619	65.50	**-2.47**	-2.35
	Avg		3310.7	219.28	2967.7	2012.69		–	–	–	**2942.2**	2948.6	65.31	**-0.86**	-0.67

Table 4.5: Type-II, Small Class – Comparison of HTS-DS with recent state-of-the-art algorithms

			HGA		**PBIG**		**CPLEX**				**HTS-DS**				
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	#	**Avg**	**T(s)**	**Avg**	**T(s)**	**#S**	**Sol**	**T(s)**	**GAP**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
10	10	3	26.7	0.18	26.7	0	3	26.7	0.01	0	26.7	26.7	0.50	0	0
	20	10	21.3	0.10	21.4	0.03	10	21.3	0.01	0	21.3	21.3	0.33	0	0
	30	10	25.5	0.10	25.6	0.01	10	25.5	0.01	0	25.5	25.5	0.22	0	0
	40	10	21.1	0.09	21.2	0	10	21.1	0.01	0	21.1	21.1	0.16	0	0
15	20	6	33.8	0.19	33.8	0.13	6	33.8	0.01	0	33.8	33.8	0.70	0	0
	40	10	48.3	0.16	49.2	0.27	10	48.3	0.02	0	48.3	48.3	0.47	0	0
	60	10	45.1	0.18	45.1	0.11	10	45.1	0.01	0	45.1	45.1	0.30	0	0
	80	10	37.9	0.16	37.9	0.01	10	37.9	0.01	0	37.9	37.9	0.27	0	0
	100	10	17.0	0.16	17.0	0.01	10	17.0	0.01	0	17.0	17.0	0.20	0	0
20	20	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	40	7	63.6	0.31	64.3	0.32	7	63.6	0.07	0	63.6	63.6	0.93	0	0
	60	10	54.1	0.26	54.1	0.37	10	54.1	0.09	0	54.1	54.1	0.67	0	0
	80	10	60.2	0.24	60.2	0.39	10	60.2	0.01	0	60.2	60.2	0.54	0	0
	100	10	64.8	0.25	64.8	0.30	10	64.8	0.04	0	64.8	64.8	0.49	0	0
	120	10	50.1	0.24	50.1	0.10	10	50.1	0.03	0	50.1	50.1	0.34	0	0
25	40	4	82.5	0.48	82.5	0.40	4	82.5	0.11	0	82.5	82.5	1.35	0	0
	80	10	80.0	0.38	80.0	0.55	10	80.0	0.13	0	80.0	80.0	0.90	0	0
	100	10	78.8	0.35	78.8	0.56	10	78.8	0.15	0	78.8	78.8	0.72	0	0
	150	10	84.2	0.36	84.2	0.50	10	84.2	0.11	0	84.2	84.2	0.54	0	0
	200	10	105.0	0.40	105.0	0.16	10	105.0	0.08	0	105.0	105.0	0.43	0	0
	Avg		**52.63**	0.24	52.73	0.22		**52.63**	0.05	0	**52.63**	**52.63**	0.53	0	0

Table 4.6: Type-II, Moderate Class – Comparison of HTS-DS with recent state-of-the-art algorithms

			HGA		**PBIG**		**CPLEX**				**HTS-DS**				
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	#	**Avg**	**T(s)**	**Avg**	**T(s)**	**#S**	**Sol**	**T(s)**	**GAP**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
50	50	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	100	4	**166.0**	0.18	**166.0**	1.72	4	**166.0**	537.33	0.0	**166.0**	**166.0**	10.12	**0.00**	**0.00**
	250	10	182.8	0.08	182.8	2.10	10	**181.4**	364.28	3.0	**181.4**	181.7	9.65	**-0.77**	-0.60
	500	10	**204.4**	0.09	**204.4**	2.53	10	**204.4**	0.32	**0.0**	204.4	**204.4**	8.97	**0.00**	**0.00**
	750	10	**215.3**	0.11	**215.3**	1.42	10	**215.3**	0.15	**0.0**	215.3	**215.3**	5.43	**0.00**	**0.00**
100	50	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	100	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	250	7	306.1	0.80	307.4	10.27	5	443.6	TL	41.9	**301.3**	301.4	18.24	**-1.57**	-1.54
	500	10	364.4	0.51	**348.9**	10.00	10	429.5	2147.82	18.4	**348.9**	**348.9**	14.27	**0.00**	0.00
	750	10	464.5	0.42	432.2	1.11	10	431.2	1336.42	2.5	**431.3**	431.7	13.31	**-0.21**	-0.12
150	50	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	100	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	250	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	500	9	490.0	2.36	**479.1**	26.26	1	818.0	TL	49.7	494.8	494.8	19.83	3.27	3.27
	750	10	622.5	1.84	596.9	27.82	3	775.3	TL	33.4	**595.0**	595.6	17.99	**-0.32**	-0.22
200	50	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	100	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	250	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	500	3	678.7	6.68	664.0	53.55	0				**662.7**	663.2	31.40	**-0.20**	-0.12
	750	10	733.2	5.45	**709.0**	54.61	0				735.3	735.8	28.30	3.71	3.78
250	250	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	500	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	750	6	924.7	12.07	**896.5**	107.57	0				920.8	922.9	33.73	2.71	2.94
	1000	9	1014.6	10.66	**967.8**	105.83	0				976.6	977.4	30.57	0.90	0.99
	2000	10	1272.9	5.35	**1167.8**	93.20	0				1190.5	1192.2	20.83	1.94	2.09
	5000	10	1666.7	8.57	**1471.9**	85.99	9	1714.1	2915.01	14.7	1572.8	1572.8	15.10	6.86	6.86
300	250	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	500	0	–	–	–	–	–	–	–	–	–	–	–	–	–
	750	1	999.0	22.29	**981.0**	183.42	0				1015.0	1019.9	42.40	3.47	3.97
	1000	9	1092.6	20.36	**1058.7**	175.36	0				1082.0	1083.8	38.27	2.20	2.37
	2000	10	1395.6	11.24	**1294.7**	158.44	0				1302.9	1304.2	29.62	0.63	0.73
	5000	10	1934.2	11.47	**1622.4**	131.93	6	1822.3	2964.34	19.1	1744.8	1744.8	17.77	7.54	7.54
		Avg	775.17	6.34	**724.57**	64.90		–	–	–	744.30	742.75	21.36	1.59	1.68

HTS-DS was run ten times with different random seeds on each instance. The recent state-of-the-art algorithms for the MWCDS problem have the associated time scaling factor 0.29 (based on *Passmark* benchmark). All average time values compared in this section will be multiplied by the associated factor, in order to account for CPU differences and conduct a fair comparison.

For all datasets of the small class, the HTS-DS retrieves all known optimal solution values. In the Shyu's Type I dataset, the algorithm produces solutions of high quality, with an average gap from the previous BKS of -0.56% for the moderate class, meaning that the average solution quality of HTS-DS is better than the best solution ever found in all previous algorithms in the literature. In a similar fashion, for the large class, the HTS-DS algorithm improved 7 of the 9 instance groups, i.e., it produces best solutions for 77.8% of the instances, with an average percentage gap of -0.67% relative to the BKS from previous literature. For both moderate and large classes, HTS-DS is also faster than PBIG algorithm, with an average CPU time of 9.96 versus 14.07 seconds, and 65.31 versus 586.47 seconds for the moderate and large classes, respectively, (time value adjusted with the factor 0.29). In turn, for Shuy's Type II dataset, the HTS-DS algorithm improves or matches 53% of the instance groups of the moderate class. For the instances with $|\mathcal{V}| \geq 250$, the HTS-DS cannot improved the solution.

The similar observations stand for the benchmark instances of Dagdeviren's dataset, the results are presented in the Tables 4.7 - 4.9.

Table 4.7: Dagdeviren's dataset, small class comparison of HTS-DS with recent state-of-the-art algorithms

		HGA		**PBIG**		**CPLEX**			**HTS-DS**				
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	**Avg**	**T(s)**	**Avg**	**T(s)**	**Sol**	**GAP**	**T(s)**	**Best**	**Avg**	**T(s)**	$\mathbf{Gap_B}$	$\mathbf{Gap_A}$
10	20	122.7	0.035	122.7	0.02	122.7	0	0.01	122.7	122.7	0.63	0	0
	40	53.5	0.025	54.6	0	53.5	0	0	53.5	53.5	0.39	0	0
15	30	250.1	0.052	250.1	0.21	250.1	0	0.05	250.1	250.1	0.95	0	0
	60	91.1	0.048	91.1	0.06	91.1	0	0.02	91.1	91.1	0.76	0	0
	90	62.5	0.046	66.6	0.01	62.5	0	0.01	62.5	62.5	0.52	0	0
20	40	329.8	0.098	329.8	0.37	329.8	0	0.12	329.8	329.8	1.68	0	0
	80	164.0	0.096	164.7	0.36	164.0	0	0.11	164.0	164.0	1.42	0	0
	120	79.5	0.078	79.5	0.07	79.5	0	0.03	79.5	79.5	1.13	0	0
	160	55.5	0.073	56.9	0.01	55.5	0	0.02	55.5	55.5	0.97	0	0
25	50	489.4	0.149	489.4	0.54	489.4	0	0.02	489.4	489.4	0.54	0	0
	100	182.9	0.110	182.9	0.51	182.9	0	0.12	182.9	182.9	1.38	0	0
	150	117.7	0.117	117.7	0.44	117.7	0	0.12	117.7	117.7	1.30	0	0
	200	75.2	0.118	75.2	0.13	75.2	0	0.05	75.2	75.2	1.15	0	0
	250	60.8	0.110	60.8	0.03	60.7	0	0.03	60.7	60.7	0.87	0	0
	Avg	**152.5**	0.083	153.0	0.20	**152.5**	0	0.05	**152.5**	**152.5**	0.98	0	0

Table 4.8: Dagdeviren's Dataset, Moderate Class – Comparison of HTS-DS with recent state-of-the-art algorithms

		HGA		**PBIG**		**CPLEX**			**HTS-DS**				
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	**Avg**	**T(s)**	**Avg**	**T(s)**	**Sol**	**GAP**	**T(s)**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
50	100	908.1	0.11	905.0	1.87	**902.7**	0	90.11	**902.7**	902.8	12.27	**-0.25**	-0.24
	200	412.8	0.08	**407.5**	1.88	**407.5**	0	11.67	**407.5**	**407.5**	12.42	**0**	**0**
	300	263.5	0.07	**257.1**	2.02	**257.1**	0	6.98	**257.1**	**257.1**	12.07	**0**	**0**
	400	186.0	0.08	**183.0**	2.14	**183.0**	0	1.92	**183.0**	**183.0**	8.84	**0**	**0**
	500	148.5	0.10	147.4	2.16	**147.3**	0	1.28	**147.3**	147.4	7.75	**-0.07**	0
100	200	1856.5	0.68	1804.7	8.89	1806.3	0.02	122.10	**1796.3**	1797.9	21.44	**-0.47**	-0.38
	400	873.2	0.52	845.9	8.78	–	–	–	**843.9**	844.0	18.73	**-0.24**	-0.22
	600	599.2	0.43	559.8	8.58	–	–	–	**559.3**	559.4	17.49	**-0.09**	-0.07
	800	450.0	0.40	412.7	8.59	412.8	0.01	661.30	412.7	412.7	16.00	**0**	**0**
	1000	338.9	0.44	314.8	8.86	314.8	0	414.43	314.8	314.8	14.66	**0**	**0**
150	300	2927.6	2.17	2817.0	25.94	–	–	–	**2786.4**	2788.2	38.03	**-1.09**	-1.02
	600	1300.7	1.57	1237.0	21.46	1501.7	0.27	TL	**1222.1**	1223.0	30.95	**-1.20**	-1.13
	900	878.8	1.29	826.2	21.55	–	–	–	**818.4**	818.4	28.11	**-0.94**	**-0.94**
	1200	694.3	1.18	623.1	21.73	773.5	0.27	TL	**621.1**	621.3	26.75	**-0.32**	-0.29
	1500	508.5	1.21	460.4	21.46	470.8	0.06	2331.63	**456.8**	456.8	26.05	**-0.78**	**-0.78**
200	400	3884.6	4.98	3683.5	54.85	–	–	–	**3650.9**	3653.0	52.50	**-0.89**	-0.83
	800	1797.2	3.90	1674.8	44.52	–	–	–	**1654.9**	1655.8	44.94	**-1.19**	-1.13
	1200	1193.5	2.84	1115.5	42.15	–	–	–	**1105.4**	1107.3	46.77	**-0.91**	-0.74
	1600	891.1	2.59	784.7	41.68	–	–	–	**780.1**	780.2	39.86	**-0.59**	-0.57
	2000	694.7	2.62	614.6	38.89	–	–	–	**614.4**	615.1	34.15	**-0.03**	0.08
	Avg	1040.4	1.36	983.7	19.40	–	–	–	**976.76**	977.29	25.49	**-0.45**	-0.41

Table 4.9: Dagdeviren's Dataset, Large Class – Comparison of HTS-DS with recent state-of-the-art algorithms

		HGA		**PBIG**		**CPLEX**			**HTS-DS**				
$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	**Avg**	**T(s)**	**Avg**	**T(s)**	**Sol**	**GAP**	**T(s)**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
250	500	4716.1	9.9	4585.0	110.7	–	–	–	**4490.2**	4495.0	37.46	**-2.07**	-1.96
	1000	2481.8	9.1	2228.0	90.4	–	–	–	**2220.3**	2226.2	29.00	**-0.35**	-0.08
	1500	1548.1	6.0	1384.0	72.1	–	–	–	**1365.7**	**1365.7**	25.70	**-1.32**	**-1.32**
	2000	1104.9	4.9	1062.4	79.6	–	–	–	**1020.3**	**1020.3**	21.89	**-3.96**	**-3.96**
	2500	960.3	4.9	822.2	62.3	–	–	–	**822.1**	**822.1**	20.74	**-0.01**	**-0.01**
500	1000	9444.3	85.9	8859.4	795.1	–	–	–	**8747.7**	8768.2	67.83	**-1.26**	-1.03
	2000	4693.7	68.8	4393.7	621.3	–	–	–	**4279.7**	4283.7	62.86	**-2.59**	-2.50
	3000	3256.9	49.5	2915.8	512.9	–	–	–	**2871.2**	2879.5	56.93	**-1.53**	-1.24
	4000	2517.9	37.7	2164.3	467.7	–	–	–	**2145.7**	2149.8	51.80	**-0.86**	-0.67
	5000	1766.5	34.3	1552.7	363.4	–	–	–	**1531.6**	1539.7	46.44	**-1.36**	-0.84
750	1500	15491.2	305.2	14298.5	2393.7	–	–	–	**14139.3**	14264.9	123.58	**-1.11**	-0.23
	3000	6979.4	264.0	6250.9	1718.4	–	–	–	**6173.5**	**6173.5**	102.63	**-1.24**	**-1.24**
	4500	4878.3	199.3	4383.5	1538.5	–	–	–	**4383.1**	4383.5	96.45	**-0.01**	0
	6000	3602.2	143.8	3226.9	1524.5	–	–	–	**3163.1**	3165.8	83.69	**-1.98**	-1.89
	7500	2823.6	119.0	**2435.0**	1452.0	–	–	–	2435.9	2436.1	76.32	0.04	0.05
1000	2000	19786.5	706.5	18235.3	6052.6	–	–	–	**18187.5**	18216.1	182.82	**-0.26**	-0.11
	4000	9532	609.6	8475.9	4651.5	–	–	–	**8375.1**	8379.8	141.87	**-1.19**	-1.13
	6000	5938.7	429.5	**5341.9**	3798.9	–	–	–	5348.5	5348.5	133.35	0.12	0.12
	8000	4557	345.4	**3983.5**	3471.9	–	–	–	3988.9	3997.7	121.77	0.14	0.36
	10000	3755.9	280.6	3188.9	3113.4	–	–	–	**3143.5**	3149.3	115.20	**-1.42**	-1.24
	Avg	5491.77	185.69	4989.4	1644.5	–	–	–	4941.6	**4953.5**	**79.9**	**-1.11**	-0.94

For the Dagdeviren's instances, the HTS-DS algorithm performed better than the HGA and PBIG algorithms in a average GAP. The HTS-DS algorithm retrieves all solutions for the instances of moderate class, with an average gap of -0.41% relative to the best knows solutions. Again, for this benchmark, HTS-DS retrieves 17 solutions, with an average relative gap of -0.94%, in a time which is significantly shorter than previous approaches: 79.90 versus 479.20 for the HTS-DS and PBIG algorithm, respectively.

We fitted the CPU time spent in the tabu search phase by a least-squares regression of an affine function on the log-log graph, in order to analyze the scalability method, for the Shyu's Type II benchmark. The time appears to grow as $\mathcal{O}(n^{1.41})$ on the instances of Shyu's dataset for the Type 2, as shown in Figure 4.1. It leads to conclude that our method is scalable.

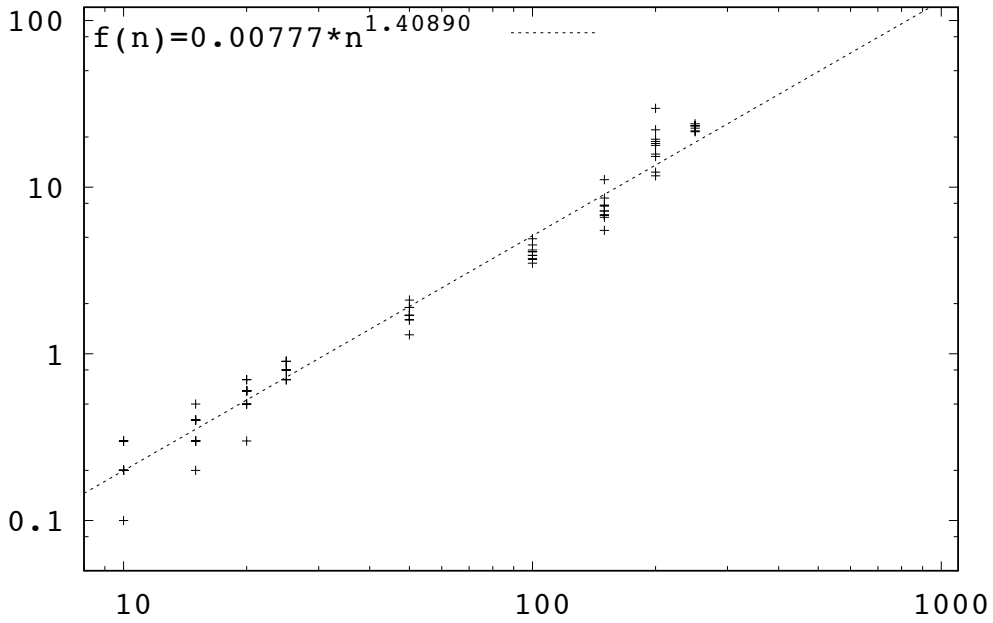


Figure 4.1: Computational time via a least-squares regression of an affine function on the log-log graph $|\mathcal{V}| \times \mathrm{T}(s)$.

4.4.3 Sensitivity analyses

We identified that all components play an important role in the proposed algorithm. With this established, we analyzed the behavior of HTS-DS algorithm according to a change in its parameters for the MWCDS problem.

Starting with the standard configuration described in Section 4.4.1, we modified the parameters related to the size tabu list and penalty factor. The following configurations were considered: for the size of the tabu list, the rank was $[10,\ 40]$ and for the penalty factor, $\beta = [0.5,\ 2.3]$. For the MWCDS

problem, we measure how the tabu tenure parameter and the penalty factor influence the algorithm. For this, we ran the algorithm 10 times for the group of Type I and Type II instances. We modified the parameter to evaluate its effect.

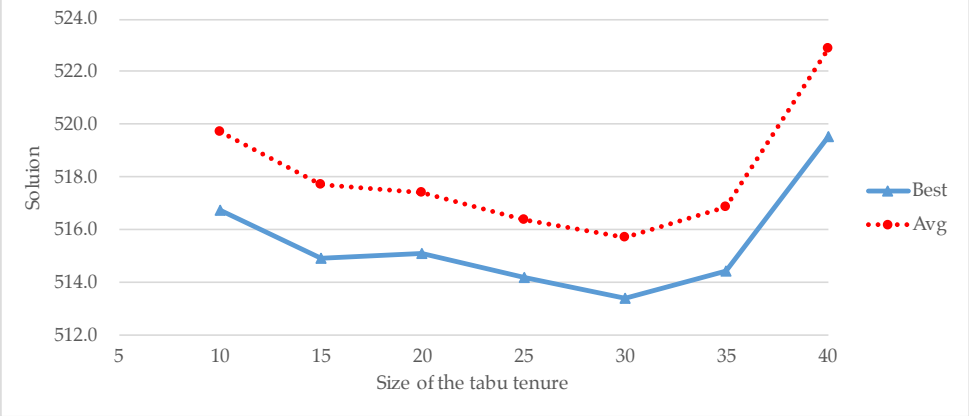


Figure 4.2: Solution quality of the HTS-DS algorithm as a function of the size of tabu list.

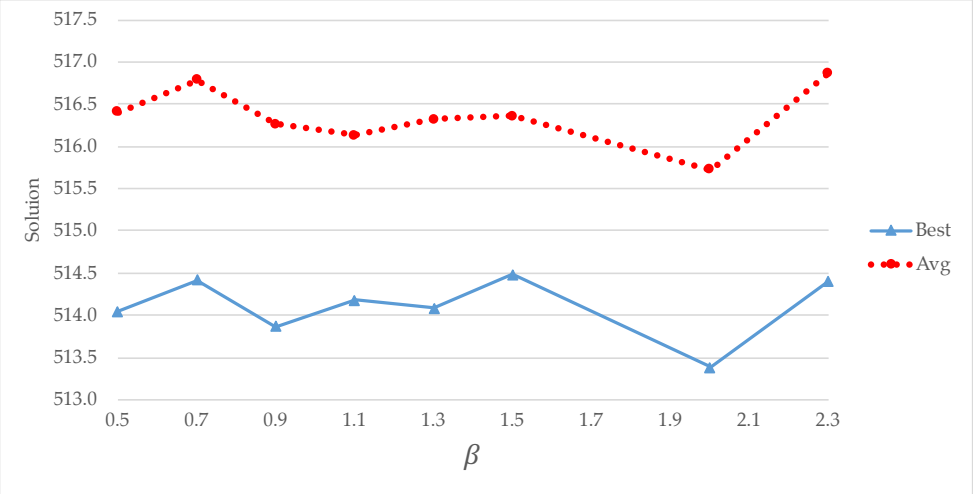


Figure 4.3: Solution quality of the HTS-DS algorithm as a function of the penalty factor β.

Figure 4.2 shows that the tabu list size next to 30 is the best option. The reason is that the search leads to unexplored area and mainly avoiding cycling. However, if this value is less than 30, the search leads to solutions of lower quality. In turn, Figure 4.3 show that similar behavior when the value of the parameter β varies between 0.5 and 1.5. The best parameter value is 2.0.

4.5 Experiments on the Connected Dominating Set

Since the methodological adaptations are the same, we applied to the MCDS problem the main procedures of the HTS-DS algorithm. We considered $w_i = 1$ for each vertex i in this problem. This section discusses the design computational experiments to test the efficiency of the HTS-DS algorithm for solving the MCDS problem. Three benchmark sets are considered: LMS, BPFTC and RGG graphs. The first benchmark originates from Lucena et al. (2010) and contains 41 random graphs from the interval of vertices [30, 200], including edge densities from the interval [5%, 70%]. For the next benchmark for the MCDS problem, the instance has its name starting with the IEEE and RTS. This benchmark, called BPFTC graph, is introduced from the computational study in Fan & Watson (2012). Finally, RGG benchmark is the same graphs presented by Jovanovic & Tuba (2013), with up to 400 vertices. This benchmark contains 41 large graphs and is obtained by randomly placing n vertices in a N $\times$ N square and connecting two vertices with an edge if their distance is smaller than a given threshold R (radius). The algorithm was executed 10 times for each instance.

4.5.1 Parameters Calibration

The final parameter values for the MCDS problem are presented in Table 4.10. These values are were calibrated after preliminary experiments.

Table 4.10: Parameter configuration of HTS-DS for the MCDS problem

Parameter	Symbol	Value
Number of restarts	N_{RESTART}	10
Maximum number of iterations of TS	I_{MAX}	20000
Maximum number of iterations without improvement of TS	I_{NI}	10000
Frequency of the perturbation	I_{PERT}	100
Perturbation strength	ρ	0.2
Size of the tabu list	N_{TABU}	10
Time limit for the IP solver	T_{MAX}	1 sec
Penalty factor	α^{MIN}	0.1
	α^{MAX}	1.1
	β	0.7

For the class with few vertices, usually smaller than 50 vertices, we consider the size of the tabu list in 10% of the number vertices ($\lfloor 0.1 * |\mathcal{V}| \rfloor$)

this will ensure that the algorithm is correct, for example, the instances IEEE-14-Bus and IEEE-30-Bus.

4.5.2 Performance of the proposed algorithm for the CDS problem

Table 4.12 presents the results for the instances of the MCDS problem in comparison to the computational results proposed by Gendron et al. (2014) and Wu et al. (2017). The columns in this table present the results for the following methods: branch-and-cut (columns SABC and IPBC); Benders decomposition (columns SABE and IPBE) and hybrid algorithms (columns SAHY and IPHY). The character " - " is used whenever an algorithm hits the limit time of 3600 sec, without producing an optimality guarantee. In turn, Table 4.13 presents the computational results obtained on RGG instances.

HTS-DS can reach the best known solution for all LMS instances. In contrast, for the BPFTC instances, HTS-DS can found 98% of the best known solutions in less time in comparison to the current exact methods.

Table 4.11 presents the recent state-of-the-art algorithms in which the results will be compared with the HTS-DS algorithm. This table indicates the CPU model used by each study, along with the associated time scaling factor. The time values reported by previous studies will be multiplied by the associated factors.

Table 4.11: List of methods considered in the experiments, and CPU model information

Acronym	Description	CPU	Factor
ACO-MMAS	Max-min ant system ant colony approach of Jovanovic & Tuba (2013)	Not available	–
ACO-PCS	Ant colony algorithm with a pheromone correction strategy of Jovanovic & Tuba (2013)	Not available	–
GRASP	Greedy randomized adaptive search procedure Li et al. (2017)	Xeon E7-48302.13 GHz	0.25
RSN	Restricted swap neighborhood search of Wu et al. (2017)	i3 3.0Ghz	0.38
HTS-DS	Hybrid tabu search of this paper	i7-3960K 3.3GHz	1.00

Table 4.12: Performance of the HTS-DS on LMS and BPFTC instances in comparison with the current state-of-the-art algorithms

			OPT	CPLEX			SABC				IPBC			SAHE			IPHE			SAHY			IPHY			R-SWAP		HTS-DS		
Inst	$\|V\|$	$\|E\|$	**f**	**f**	**Gap**	**T(s)**	**IUB**	**UB**	**Gap**	**T(s)**	**UB**	**Gap**	**T(s)**	**UB**	**Gap**	**T(s)**	**UB**	**Gap**	**T(s)**	**UB**	**Gap**	**T(s)**	**UB**	**Gap**	**T(s)**	**best**	**T(s)**	**best**	**Gap**	**T(s)**
v30_d10	30	44	15	15	0	2.20	15	15	0	0.03	15	0	0.02	15	0	1222.10	15	0	756.87	15	0	6.11	15	0	2.88	15	1	15	0	1.4
v30_d20	30	87	7	7	0	0.30	7	7	0	0.02	7	0	0.02	7	0	0.01	7	0	0.00	7	0	0.01	7	0	0.02	7	1	7	0	1.4
v30_d30	30	131	4	4	0	0.30	5	4	0	0.05	4	0	0.06	4	0	0.02	4	0	0.02	4	0	0.03	4	0	0.02	4	1	4	0	1.2
v30_d50	30	218	3	3	0	0.10	3	3	0	0.01	3	0	0.03	3	0	0.00	3	0	0.00	3	0	0.01	3	0	0.01	3	1	3	0	0.9
v30_d70	30	305	2	2	0	0.00	2	2	0	0.02	2	0	0.00	2	0	0.00	2	0	0.00	2	0	0.01	2	0	0.00	2	1	2	0	0.9
v50_d5	50	61	31	31	0	4.00	31	31	0	0.01	31	0	0.02	31	0	3600.00	31	0	3600.00	31	0	88.63	31	0	9.46	31	1	31	0	2.4
v50_d10	50	123	12	12	0	3.60	13	12	0	0.82	12	0	0.20	12	0	34.30	12	0	3.09	12	0	2.69	12	0	3.89	12	1	12	0	1.7
v50_d20	50	245	7	7	0	4.70	7	7	0	0.77	7	0	0.97	7	0	0.21	7	0	0.09	7	0	0.40	7	0	0.16	7	1	7	0	1.6
v50_d30	50	368	5	5	0	8.10	5	5	0	0.32	5	0	0.25	5	0	0.18	5	0	0.11	5	0	0.45	5	0	0.31	5	1	5	0	1.4
v50_d50	50	613	3	3	0	0.30	3	3	0	0.23	3	0	0.06	3	0	0.00	3	0	0.01	3	0	0.01	3	0	0.01	3	1	3	0	1.2
v50_d70	50	858	2	2	0	0.20	2	2	0	0.24	2	0	0.01	2	0	0.00	2	0	0.00	2	0	0.01	2	0	0.01	2	1	2	0	1.1
v70_d5	70	121	27	28	12.4	3619.00	29	27	0	2.06	27	0	0.39	29	7.41	3600.00	29	7.41	3600.00	27	0	188.65	27	0	674.75	27	1	27	0	2.2
v70_d10	70	242	13	13	0	339.30	14	13	0	18.68	13	0	5.25	13	0	1.06	13	0	2.17	13	0	25.16	13	0	1.26	13	1	13	0	2.1
v70_d20	70	483	7	7	0	86.30	7	7	0	2.68	7	0	1.88	7	0	0.38	7	0	0.17	7	0	1.15	7	0	0.58	7	1	7	0	2
v70_d30	70	725	5	5	0	66.70	5	5	0	1.20	5	0	0.99	5	0	0.54	5	0	0.21	5	0	0.82	5	0	0.37	5	1	5	0	1.9
v70_d50	70	1208	3	3	0	1.50	3	3	0	0.64	3	0	0.40	3	0	0.01	3	0	0.02	3	0	0.02	3	0	0.02	3	1	3	0	1.6
v70_d70	70	1691	2	2	0	1.10	2	2	0	0.99	2	0	0.04	2	0	0.00	2	0	0.01	2	0	0.02	2	0	0.02	2	1	2	0	1.5
v100_d5	100	248	24	26	19.3	3622.80	25	24	0	58.77	24	0	64.13	25	4.17	3600.00	25	4.17	3600.00	25	4.17	3600.00	24	0	142.49	24	1	24	0	2.9
v100_d10	100	495	13	13	0	1942.00	13	13	0	28.25	13	0	39.71	13	0	0.49	13	0	0.33	13	0	2.68	13	0	1.70	13	1	13	0	2.5
v100_d20	100	990	8	8	0	690.50	8	8	0	283.23	8	0	414.49	8	0	1.88	8	0	1.26	8	0	6.48	8	0	2.70	8	1	8	0	2.4
v100_d30	100	1485	6	6	0	708.90	6	6	0	329.05	6	0	638.89	6	0	3.83	6	0	2.46	6	0	11.17	6	0	4.42	6	1	6	0	2.4
v100_d50	100	2475	4	4	0	197.90	4	4	0	48.00	4	0	41.51	4	0	1.55	4	0	0.76	4	0	3.23	4	0	1.56	4	1	4	0	1.9
v100_d70	100	3465	3	3	0	83.70	3	3	0	13.20	3	0	12.02	3	0	1.55	3	0	0.03	3	0	1.57	3	0	0.91	3	1	3	0	1.8
v120_d5	120	357	25	31	32.3	3586.60	26	25	0	1465.05	25	0	199.01	25	0	3.36	25	0	18.16	25	0	102.61	25	0	35.10	25	1	25	0	3.1
v120_d10	120	714	13	13	0	3272.30	15	15	15.38	3600.00	13	0	3600.00	13	0	23.97	13	0	3.86	13	0	56.31	13	0	18.68	13	1	13	0	2.8
v200_d20	120	1428	8	9	28.0	3584.90	8	8	0	1316.70	8	0	3600.00	8	0	5.02	8	0	3.79	8	0	16.47	8	0	8.31	8	2	8	0	2.9
v120_d30	120	2142	6	6	0	1132.80	6	6	0	790.91	6	0	1913.36	6	0	5.25	6	0	4.44	6	0	14.21	6	0	7.56	6	1	6	0	3
v120_d50	120	3570	4	4	0	609.90	4	4	0	246.93	4	0	202.30	4	0	4.21	4	0	2.52	4	0	8.73	4	0	4.57	4	1	4	0	2.4
v120_d70	120	4998	3	3	0	137.30	3	3	0	36.84	3	0	28.90	3	0	2.25	3	0	0.04	3	0	2.82	3	0	2.22	3	1	3	0	2.1
v150_d5	150	559	26	35	40.0	3581.30	27	27	3.85	3600.00	27	3.85	3600.00	27	3.85	3600.00	26	0	771.07	27	3.85	3600.00	26	0	3600.00	26	2	26	0	3.6
v150_d10	150	1118	14	18	36.9	3614.00	15	15	7.14	3600.00	15	7.14	3600.00	14	0	51.09	14	0	28.28	14	0	652.35	14	0	195.76	14	4	14	0	3.4
v150_d20	150	2235	9	10	38.1	3620.30	9	9	0	3600.00	9	0	3600.00	9	0	367.30	9	0	271.65	9	0	2116.24	9	0	903.76	9	2	9	0	3.4
v150_d30	150	3353	6	7	40.1	3610.90	6	6	0	2972.83	6	0	3600.00	6	0	21.12	6	0	11.25	6	0	34.61	6	0	24.77	6	3	6	0	3.5
v150_d50	150	5588	4	4	0	2146.90	4	4	0	724.92	4	0	477.10	4	0	7.81	4	0	5.78	4	0	17.57	4	0	10.79	4	1	4	0	3.2
v150_d70	150	7823	3	3	0	380.20	3	3	0	62.56	3	0	49.69	3	0	4.30	3	0	0.06	3	0	5.01	3	0	2.95	3	3	3	0	2.8
v200_d5	200	995	27	-	0	3600.00	29	29	7.41	3600.00	29	7.41	3600.00	29	7.41	3600.00	27	0	1658.85	29	7.41	3600.00	28	3.70	3600.00	27	4	27	0	4.8
v200_d10	200	1990	16	21	49.0	3616.10	16	16	0	3600.00	16	0	3600.00	16	0	3600.00	16	0	3600.00	16	0	3600.00	16	0	3600.00	16	2	16	0	4.6
v200_d20	200	3980	9	-	0	3600.00	9	9	0	3600.00	9	0	3600.00	9	0	1686.30	9	0	1945.80	9	0	3600.00	9	0	3600.00	9	2	9	0	4.8
v200_d30	200	5970	7	10	64.5	3594.50	7	7	0	3600.00	7	0	3600.00	7	0	3210.83	7	0	1847.88	7	0	3600.00	7	0	3600.00	7	2	7	0	5
v200_d50	200	9950	4	6	65.0	3594.00	4	4	0	3363.33	4	0	1887.43	4	0	24.79	4	0	19.33	4	0	44.42	4	0	28.54	4	2	4	0	4.4
v200_d70	200	13930	3	3	0	1235.90	3	3	0	340.20	3	0	275.84	3	0	10.53	3	0	0.13	3	0	9.17	3	0	5.63	3	2	3	0	4
IEEE-14-Bus	14	20	5	5	0	0.00	5	5	0	0.01	5	0	0.00	5	0	0.00	5	0	0.00	5	0	0.00	5	0	0.00	5	1	5	0	0.5
IEEE-30-Bus	30	41	11	11	0	0.10	11	11	0	0.01	11	0	0.01	11	0	0.00	11	0	0.00	11	0	0.00	11	0	0.01	11	1	11	0	2
IEEE-57-Bus	57	78	31	31	0	107.10	31	31	0	0.02	31	0	0.03	31	0	3600.00	31	0	3600.00	31	0	24.40	31	0	192.43	31	1	31	0	2.3
RTS96	73	108	32	32	0	73.20	34	32	0	2.56	32	0	4.22	34	6.25	3600.00	34	6.25	3600.00	32	0	118.53	32	0	26.56	32	1	32	0	2.9
IEEE-118-Bus	118	179	43	43	0	1844.40	43	43	0	0.22	43	0	0.51	43	0	3600.00	43	0	3600.00	43	0	65.65	43	0	2.88	43	1	43	0	4.8
IEEE-300-Bus	300	409	129	-	-	3600.00	139	139	7.75	3600.00	136	5.43	3600.00	139	7.75	3600.00	139	7.75	3600.00	139	7.75	3600.00	138	6.98	3600.00	129	57	131	1.55	19.4
AVG			13.70	-	-	1394.17	14.23	14.04	0.88	947.07	13.94	0.51	975.74	14.09	0.78	908.43	14.02	0.54	769.37	14.00	0.49	613.37	13.91	0.23	508.90	**13.70**	**2.57**	13.74	0.03	2.90

Table 4.13: Performance of the HTS-DS on RGG instances in comparison with the current state-of-the-art algorithms

Area(N*N)	**R**	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	**ACO-MMAS**		**ACO-PCS**		**GRASP**			**RSN**			**HTS-DS**				
Nodes				**Best**	**Avg**	**Best**	**Avg**	**Best**	**Avg**	**T(s)**	**Best**	**Avg**	**T(s)**	**Best**	**Avg**	**T(s)**	**Gap$_B$**	**Gap$_A$**
400	60	80	262	20	21.6	19	21.2	19	19.8	1.8	**18**	**18.0**	7.5	**18**	**18.0**	6.0	0	0
80	70	80	329	16	17.0	15	16.2	**14**	15.1	1.9	**14**	**14.0**	6.0	**14**	**14.0**	6.4	0	0
	80	80	400	**12**	14.0	**12**	13.1	**12**	12.0	1.9	**12**	**12.0**	4.5	**12**	**12.0**	6.6	0	0
	90	80	474	11	11.8	11	11.6	**10**	10.6	2.1	**10**	**10.0**	4.1	**10**	**10.0**	6.8	0	0
	100	80	563	**8**	9.0	**8**	8.9	**8**	8.2	2.1	**8**	**8.0**	3.0	**8**	**8.0**	5.7	0	0
	110	80	647	8	8.5	8	8.5	**7**	7.8	2.2	**7**	**7.0**	3.0	**7**	**7.0**	5.3	0	0
	120	80	735	7	7.5	7	7.2	**6**	6.1	2.4	**6**	**6.0**	2.6	**6**	**6.0**	5.5	0	0
600	80	100	335	23	24.7	22	23.6	22	22.9	2.2	**21**	**21.0**	8.6	**21**	**21.0**	7.2	0	0
100	90	100	368	22	23.8	21	23.6	20	20.7	2.4	**19**	**19.0**	8.3	**19**	**19.0**	7.4	0	0
	100	100	435	17	20.0	17	19.0	17	17.9	2.5	**16**	**16.0**	6.4	**16**	16.0	7.3	0	0
	110	100	500	15	17.2	15	16.8	15	15.9	2.4	15	15.0	5.6	**14**	14.7	7.3	**-6.7**	**-2.0**
	120	100	575	15	16.2	14	15.5	**13**	13.8	4.3	**13**	13.0	5.6	**13**	13.0	7.5	0	0
700	70	200	756	46	50.7	46	49.6	45	46.5	10.9	39	39.0	17.3	**38**	38.8	10.9	**-2.6**	**-0.5**
200	80	200	921	41	43.7	41	43.9	35	37.5	10.1	**32**	32.0	15.4	**32**	32.0	10.7	0	0
	90	200	1113	34	36.0	33	35.7	30	30.9	9.3	26	26.0	12.8	26	26.5	11.7	0	1.9
	100	200	1305	28	30.8	28	31.0	25	25.8	8.7	**22**	**22.0**	10.5	23	23.0	11.9	4.5	4.5
	110	200	1501	23	27.4	22	26.4	22	22.7	8.8	**20**	**20.0**	8.6	**20**	20.1	11.9	0	0.5
	120	200	1730	21	23.6	21	23.4	18	19.1	8.6	**17**	**17.0**	7.9	**17**	17.1	11.7	0	0.6
1000	100	200	756	46	50.7	46	49.6	45	46.5	11.0	**38**	38.4	17.3	**38**	39.0	10.9	0	2.6
200	110	200	871	43	44.9	42	44.8	37	39.5	10.2	**34**	**34.0**	16.1	**34**	**34.0**	10.4	0	0
	120	200	997	37	39.9	37	39.8	34	35.4	10.2	**29**	**29.0**	13.9	**29**	30.0	11.1	0	3.4
	130	200	1127	32	34.7	32	34.9	29	30.5	9.5	**26**	**26.0**	12.0	**26**	**26.0**	11.7	0	0
	140	200	1269	30	31.3	29	31.3	25	25.7	8.7	**23**	**23.0**	11.3	**23**	**23.0**	13.1	0	0
	150	200	1400	28	29.6	26	28.8	23	24.3	8.8	**21**	**21.0**	10.5	22	22.0	13.0	4.8	4.8
	160	200	1541	24	26.6	25	26.5	22	22.3	8.6	**19**	**19.0**	9.0	**19**	19.1	12.8	0	0.5
1500	130	250	903	60	64.5	60	64.3	57	58.6	14.4	**49**	**49.0**	22.5	50	50.2	14.2	2.0	2.4
250	140	250	1011	53	57.2	52	57.0	50	52.3	13.7	**44**	**44.0**	19.9	**44**	44.2	13.9	0	0.5
	150	250	1119	51	54.9	51	54.4	46	48.5	12.6	**40**	**40.0**	19.2	**40**	41.0	13.7	0	2.5
	160	250	1246	47	50.5	45	49.8	43	43.7	12.2	**36**	**36.0**	17.7	**36**	36.0	16.0	0	0
2000	200	300	1577	55	58.6	52	58.8	49	50.4	15.1	**42**	**42.0**	20.7	**42**	42.0	18.2	0	0
300	210	300	1710	51	53.5	50	52.8	45	46.1	14.8	**38**	38.3	19.2	**38**	39.0	19.1	0	2.6
	220	300	1849	47	48.9	45	48.4	40	42.1	14.6	**35**	**35.0**	17.7	**35**	35.0	19.6	0	0
	230	300	1990	44	47.5	44	46.9	39	39.8	13.9	**33**	**33.0**	16.5	**33**	33.2	19.0	0	0.6
2500	200	350	1461	79	82.0	79	81.5	73	75.4	20.5	**61**	61.1	29.7	**61**	61.9	20.4	0	1.5
350	210	350	1555	75	79.1	74	78.2	67	70.0	19.9	**56**	56.0	28.2	57	57.0	20.9	1.8	1.8
	220	350	1668	68	72.6	69	73.8	62	67.0	19.3	**52**	52.0	25.5	53	53.0	23.0	1.9	1.9
	230	350	1787	66	69.2	66	68.9	59	60.9	18.8	**50**	50.0	24.8	**50**	51.0	22.6	0	2.0
3000	210	400	1522	99	101.6	98	104.0	90	94.7	28.6	**75**	75.3	37.2	**75**	76.0	25.2	0	1.3
400	220	400	1621	88	95.4	91	97.6	82	87.9	26.9	71	71.0	33.0	**70**	71.2	25.4	**-1.4**	0.3
	230	400	1750	86	91.4	86	90.3	78	81.6	26.0	**65**	65.6	32.3	66	66.0	25.3	1.5	1.5
	240	400	1880	82	85.8	80	84.1	74	76.1	24.8	**61**	61.5	30.8	62	62.5	25.6	1.6	2.5
			Avg	40.4	43.3	40.0	43.0	36.8	38.4	10.9	**32.0**	32.1	15.2	32.1	32.4	13.5	0.2	0.9

In particular, HTS-DS obtained solutions that are at least as good as the best solutions in the literature for 24 out of 41 test instances, that is, HTS-DS retrieves 59% of the solutions. In addition, HTS-DS improved the best-known solutions in the literature for 3 instances.

In terms of computation time, our algorithm clearly outperforms other heuristics, except when is compared with the RSN algorithm, which obtains better or comparable solutions.

4.6 Concluding Remarks

In this chapter, we described the adapted HTS-DS algorithm for the MWCDS problem. This algorithm is based on a neighborhood defined by insertions and eliminations of the vertices that are not an articulation points of the graph. We use neighborhoods structures and efficient checking mechanism to find articulation point leading to a very fast and accurate algorithm. Computational experiments on benchmark problems are reported, comparing the behavior of the proposed algorithm with that of other heuristics from the literature. The method seems suitable for large-scale applications as observed by the CPU time growth in $\mathcal{O}(|V|^{1.41})$ in function of the number of vertices for the instances of the MWCDS problem. HTS-DS is compared with other heuristics, obtaining better or comparable solutions, and contributes to significant improvements in execution time.

We investigated the HTS-DS algorithm for the MWCDS problem. For this, we compared to the two population-based optimization algorithms for the MWCDS problem known in the literature. Comprehensive computational experiments and comparisons with state-of-the-art methods showed that the proposed algorithm outperforms the current state-of-the-art methods in terms of both solution quality and computational efficiency for many dataset. The HTS-DS algorithm can improve many new best known solutions from the literature.

5
Experiments on the Covering Code Problems

A covering problem consists in minimizing the number of vertices to cover a graph while satisfying some side constraints. A famous application of the problem is the lottery game (Li & van Rees, 2002), which can be reduced to the MDS problem. We will therefore consider classical benchmark instances of this problem, and attempt to generate better upper bounds with the HTS-DS algorithm.

This chapter addresses the covering code problem in Hamming space, as well as in the RT space. Section 5.1 introduces and defines some concepts of the problem. Section 5.2 describes the relation between the covering code problem and the dominating set problem. Section 5.3 presents the new upper bounds for the covering codes obtained with the HTS-DS algorithm. Finally, Section 5.4 concludes.

5.1
Preliminaries

A code for a message set $\mathcal{S}$ is a mapping from each message to a bit string. Here, each bit string is called a *codeword*. In addition, a code is distinct if each codeword is distinguishable from every other, i.e., the mapping from source messages to codewords is one-to-one.

5.1.1
Covering codes in Hamming-space

A *code* $\mathcal{C}$ is a subset of the set of binary n-strings. The Hamming distance $d(s,t)$ between two strings s and t is the number of coordinates at which they differ. A covering code of radius R is a code $\mathcal{C}$ such that, for any binary n-string s, $d(s,c) \leq R$ for some element c in $\mathcal{C}$. R should always be as small as possible.

5.1.2
Covering codes in RT-space

The covering code problem in RT space is an extension of the Hamming covering problem, called RT-covering problem and formalized in Castoldi & Carmelo (2015). Basic notations on the Rosenbloom-Tsfasman metric are presented as follows.

For $s \geq 1$ and $q \geq 2$, let $\mathbb{Z}_q^s$ be the set of all vectors $z = (z_1, \ldots, z_s)$ of length s, with $z_i \in \mathbb{Z}_q$ for any $1 \leq i \leq s$. Given $z = (z_1, \ldots, z_s)$ and $w = (w_1, \ldots, w_s)$, let

$$d_s(z, w) = \begin{cases} 0 & \text{if } z = w \\ \max\{i : z_i \neq w_i\} & \text{if } z \neq w. \end{cases}$$

A closer look shows that d_s is a metric in $\mathbb{Z}_q^s$. For $m \geq 1$, let $(\mathbb{Z}_q^s)^m$ be the set of all vectors $x = (x_1, \ldots, x_{ms})$ with $x_i \in \mathbb{Z}_q$. An arbitrary vector $x = (x_1, t \ldots, x_{ms})$ in $(\mathbb{Z}_q^s)^m$ can be represented by $x = (x^1, \ldots, x^m)$, formed by the symbols $x^i = (x_{(i-1)s+1}, \ldots, x_{is})$, where $1 \leq i \leq m$. Sometimes a vector $x = (x_1, \ldots, x_{ms})$ is simply denoted by $x = x_1 \ldots x_{ms}$.

Let $x = (x^1, \ldots, x^m)$ and $y = (y^1, \ldots, y^m)$ in $(\mathbb{Z}_q^s)^m$. The application

$$d_{RT}(x, y) = \sum_{i=1}^{m} d_s(x^i, y^i)$$

induces the *RT metric* in $(\mathbb{Z}_q^s)^m$. The space $(\mathbb{Z}_q^s)^m$ endowed with the RT metric is called the *Rosenbloom-Tsfasman space* or *RT space* for short. The particular case $s = 1$ corresponds to the well-known Hamming metric.

As usual, in a metric space, the RT-ball centered in x of radius R is the set

$$B^{RT}(x, R) = \{y \in (\mathbb{Z}_q^s)^m : d_{RT}(x, y) \leq R\},$$

and its cardinality is denoted by $V_q^{RT}(m, s, R)$.

Given a subset C of $(\mathbb{Z}_q^s)^m$, we say that C is an *R-covering* of the RT space $(\mathbb{Z}_q^s)^m$ when

$$\bigcup_{c \in C} B^{RT}(c, R) = (\mathbb{Z}_q^s)^m.$$

The minimum cardinality of an R-covering of the RT-space $(\mathbb{Z}_q^s)^m$ is denoted by $K_q^{RT}(m, s, R)$. The case $K_q^{RT}(m, 1, R)$ corresponds to the classic number $K_q(m, R)$.

Two propositions were demonstrated by Castoldi & Carmelo (2015), as follows. For every $r < s$ and $R \leq mr$:

Proposição 5.1 $K_q^{RT}(m, s, R) \leq q^{m(s-R)} K_q^{RT}(m, r, R)$

Proposição 5.2 $K_q^{RT}(m, s, R) \leq K_q^{RT}(m, s - r, R - mr)$

5.2
Covering Codes and the Dominating Set Problem

The covering code problem in RT space can be reformulated as a dominating set problem in graphs. This is demonstrated by the following theorem.

Teorema 5.3 *[Östergård (1997)] The minimum covering code problem in RT spaces corresponds to a class of the minimum dominating set problems.*

Proof. Given an RT space $(\mathbb{Z}_q^s)^m$ and $0 \leq R \leq ms$, we construct the following graph $\mathcal{G}(q, m, s, R) = (\mathcal{V}, \mathcal{E})$. We associate the vector u in $(\mathbb{Z}_q^s)^m$ with the vertex u in $\mathcal{V}$. The edge $e = \{u, v\}$ is in $\mathcal{E}$ if and only if $0 < d_{RT}(u, v) \leq R$. Note that, dominating sets in $\mathcal{G}(q, m, s, R)$ are in one-to-one correspondence with the R-covering codes of the R-space $(\mathbb{Z}_q^s)^m$. A minimum R-covering code of the RT-space $(\mathbb{Z}_q^s)^m$ can be found by solving the minimum dominating set problem in $\mathcal{G}(q, m, s, R)$. □

The graph $\mathcal{G}(2, 2, 2, 2)$ is illustrated in Figure 5.1. A dominating set in the graph $\mathcal{G}(2, 2, 2, 2)$ is $\{0000, 0101, 1111\}$. By Theorem 5.3, this set is a 2-covering of the RT space $(\mathbb{Z}_2^2)^2$.

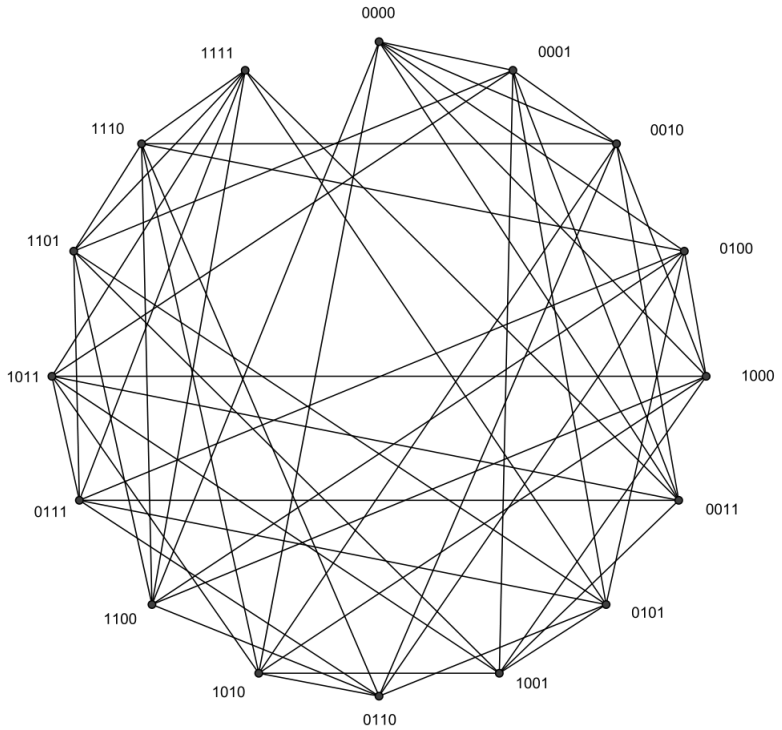


Figure 5.1: Graph $\mathcal{G}(2, 2, 2, 2)$.

A dominating set in the graph $\mathcal{G}(q, m, 1, R)$ is equivalent to an R-covering code in the Hamming-space $\mathbb{Z}_q^m$.

5.3 Computational Experiments

We conducted computational experiments to find new bounds for these covering codes. HTS-DS algorithm as presented for the MWDS problem was used. The weight of each vertex was set to one. HTS-DS was implemented in C++, and CPLEX 12.62 was used for the resolution of the integer linear programs. All tests were conducted in a single thread of an i7-3960K 3.3GHz processor.

The computational tests were performed for 48 RT-covering codes considering only the codes with $q = 2$. The study about the covering codes in Hamming space had about 5 codes for this class. We considered the codes for which lower and upper bounds are known to compare with our results.

5.3.1 Parameters Calibration

Table 5.1 shows the values of the parameters used by the HTS-DS algorithm, which were calibrated after preliminary experiments. In the codes where the number of vertices is smaller than 16, we used a tabu list of 10% of the number of the vertices $|\mathcal{V}|$.

Table 5.1: Parameter configuration of HTS-DS

Parameter	Symbol	Value
Number of restarts	N_{RESTART}	5
Maximum number of iterations of TS	I_{MAX}	20000
Maximum number of iterations without improvement of TS	I_{NI}	10000
Frequency of the perturbation	I_{PERT}	1000
Perturbation strength	ρ	0.2
Size of the tabu list	N_{TABU}	20
Time limit for the IP solver	T_{MAX}	1 sec
Penalty factor	α^{MIN}	0.1
	α^{MAX}	1.1
	β	1.0

We performed sensitivity analysis from the final parameter setting obtained. This analysis will be reported in Section 5.3.3.

5.3.2 Performance of the HTS-DS algorithm for the covering codes

Tables 5.2 and 5.3 present the results for the covering code problem in RT space and Hamming space, respectively. In this table, $K(m, s, r)$ denotes

the code of the test-problem, $|\mathcal{V}|$ is the number of vertices, $|\mathcal{E}|$ is the number of edges, and **BKB** represents the best known bound reported in the literature. More specifically, the **LB** and **UB** columns show the best lower and upper bounds. These bounds were found proved using theoretical results in Castoldi & Carmelo (2015). In turn, the bounds for the Hamming-covering codes are available in Kéri (2018) and in the book by Cohen et al. (1997).

The formulation presented by the Equations (3-1) - (3-3) (in Chapter 3) are used by the ILP solver. These results are presented in the **CPLEX** columns. **T(s)** column presents the execution time of the method. We limited the time of the ILP solver to 3600 seconds. The value 'TL' in this column represents the codes in which the solver reached this time. The **Sol** and **GAP** columns present the solution and the gap returned by CPLEX. The values '-' in the column GAP means that the algorithm did not find any feasible solution.

Best, **Avg** and **T(s)** indicate, respectively, the best solution, the average solution and the average computational time in seconds associated with the HTS-DS algorithm. We indicate in boldface the best upper bound among the approaches for each code. If an upper bound is better than the known bound or it is an optimum for the code, it is underlined.

The results of the ILP solver shows the difficulty of solving this problem. The optimality of 29 codes out off 48 was proved by CPLEX, but it took a long computational time. HTS-DS appears to be more efficient in terms of computational time, being able to improve the quality of the solution of 32 codes. In particular, HTS-DS significantly improved the codes $K(3,4,R)$ and $K(4,3,R)$, where $2 \leq R \leq 8$.

The propositions presented in Section 5.1.2 reveal that some codes in RT-space can be improved, using the new bound found by the HTS-DS algorithm.

i) $K_2^{RT}(2,6,4) \leq 160$.

The HTS-DS algorithm produces $K_2^{RT}(2,6,4) \leq 172$. When considering $r = 5$, it follows by Proposition 5.1 that: $K_2^{RT}(2,6,4) \leq 2^{2(6-5)}K_2^{RT}(2,5,4) \leq 4 \cdot 40 = 160$, which significantly improves the previous bound.

ii) $K_2^{RT}(2,6,5) \leq 80$

Like the code identified above, this code improves the bound from 82 to: $K_2^{RT}(2,6,5) \leq 2^{2(6-5)}K_2^{RT}(2,5,5) \leq 4 \cdot 20 = 80$. Therefore, the covering code $K_2^{RT}(2,6,5)$ has upper bound equal to 80.

iii) $K_2^{RT}(3,4,6) \leq 16$.

Table 5.2: Results for covering codes $K_2^{RT}(m, s, R)$: Graph $\mathcal{G}(m, s, 2, R)$

			BKB		**CPLEX**			**HTS-DS**		
K**(m,s,R)**	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	**LB**	**UB**	**Sol**	**GAP**	**T(S)**	**Best**	**Avg**	**T(s)**
K(2,2,1)	16	16	6	**8**	**8**	0.0%	0.0	**8**	8.0	0.1
K(2,3,1)	16	64	22	**32**	**32**	0.0%	0.0	**32**	32.0	0.3
K(2,3,2)	64	224	8	**12**	**12**	0.0%	0.4	**12**	12.0	0.3
K(2,3,3)	64	608	4	**8**	**6**	0.0%	0.2	**6**	6.0	0.3
K(2,4,1)	256	256	86	**128**	**128**	0.0%	0.1	**128**	128.0	1.0
K(2,4,2)	256	896	32	**48**	**48**	0.0%	1.9	**48**	48.0	1.0
K(2,4,3)	256	2432	13	32	**24**	0.0%	1.0	**24**	24.0	1.1
K(2,4,4)	256	6016	6	12	**10**	0.0%	0.7	**10**	10.0	1.4
K(2,4,5)	256	10112	4	8	**6**	0.0%	0.2	**6**	6.0	1.0
K(2,5,1)	1024	1024	342	**512**	**512**	0.0%	0.3	**512**	512.0	6.5
K(2,5,2)	1024	3584	128	**192**	**192**	0.0%	8.9	**192**	192.0	4.4
K(2,5,3)	1024	9728	52	128	**96**	0.0%	18.6	**96**	96.0	4.8
K(2,5,4)	1024	24064	22	48	**40**	31.0%	TL	**40**	42.0	5.7
K(2,5,5)	1024	56832	10	32	**20**	27.0%	TL	**20**	20.0	6.4
K(2,5,6)	1024	97792	6	12	**10**	0.0%	2.5	**10**	10.7	6.5
K(2,5,7)	1024	163328	4	8	**6**	0.0%	0.2	**6**	6.0	10.3
K(2,6,2)	4096	14336	512	768	**768**	0.0%	2.1	**768**	768.0	65.4
K(2,6,3)	4096	38912	205	512	-	-	TL	**384**	384.0	32.5
K(2,6,4)	4096	96256	86	192	-	-	TL	**174**	174.0	28.8
K(2,6,5)	4096	227328	37	128	**82**	53.0%	TL	**82**	82.4	38.7
K(2,6,6)	4096	522240	16	48	40	52.0%	TL	**39**	39.0	45.4
K(2,6,7)	4096	915456	10	32	**20**	38.0%	TL	21	21.3	54.8
K(2,6,8)	4096	1570816	6	12	**10**	0.0%	6.8	**10**	11.0	98.9
K(3,3,1)	512	768	-	-	**128**	0.0%	0.1	**128**	128.0	2.2
K(3,3,2)	512	3072	40	96	**64**	0.0%	0.9	**64**	64.0	2.1
K(3,3,3)	512	9472	14	65	**16**	0.0%	5.7	**16**	16.0	1.6
K(3,3,4)	512	20224	7	16	**12**	0.0%	453.3	**12**	12.0	3.4
K(3,3,5)	512	38656	4	12	**6**	0.0%	5.3	**6**	6.0	3.7
K(3,3,6)	512	65280	2	8	**4**	0.0%	1.2	**4**	4.0	4.6
K(3,4,2)	4096	24576	316	768	528	34.0%	TL	**512**	512.0	36.4
K(3,4,3)	4096	75776	108	512	267	73.0%	TL	**128**	128.0	20.0
K(3,4,4)	4096	210944	40	128	200	80.0%	TL	**64**	69.3	38.1
K(3,4,5)	4096	456704	19	96	61	69.0%	TL	**36**	36.8	58.2
K(3,4,6)	4096	915456	10	48	25	59.0%	TL	**21**	21.0	74.4
K(3,4,7)	4096	1701888	5	16	**12**	43.0%	TL	**12**	12.0	111.2
K(3,4,8)	4096	2881536	3	12	**6**	0.0%	568.2	7	7.0	191.9
K(4,2,1)	256	512	52	**64**	**64**	0.0%	0.0	**64**	64.0	1.0
K(4,2,2)	256	2304	14	48	**16**	0.0%	1.8	**16**	16.0	0.9
K(4,2,3)	256	5888	6	16	**8**	0.0%	0.8	**8**	8.0	1.3
K(4,2,4)	256	12160	3	8	**4**	0.0%	1.2	**4**	4.0	1.7
K(4,3,1)	4096	8192	820	**1024**	**1024**	0.0%	0.8	**1024**	1024.0	84.4
K(4,3,2)	4096	36864	216	768	386	68.0%	TL	**256**	256.0	26.5
K(4,3,3)	4096	126976	66	256	235	72.0%	TL	**124**	124.5	31.0
K(4,3,4)	4096	325632	26	128	185	86.0%	TL	**54**	54.0	49.5
K(4,3,5)	4096	735232	12	48	102	89.0%	TL	**26**	26.0	90.3
K(4,3,6)	4096	1472512	6	32	30	81.0%	TL	**14**	14.0	158.2
K(4,3,7)	4096	2521088	4	16	12	72.0%	TL	**8**	8.0	208.0
K(4,3,8)	4096	3930112	3	12	**4**	0.0%	957.2	**4**	4.0	234.3
		Avg	72.6	151.3	118.9	-	-	**109.2**	109.4	38.6

The HTS-DS algorithm produces $K_2^{RT}(3,4,6) \leq 21$. An application of Proposition 5.2 when $r = 1$ results in: $K_2^{RT}(3,4,6) \leq K_2^{RT}(3,3,3) \leq 16$.

Table 5.3: Results for covering $K_q(n,R)$: Graph $\mathcal{G}(n,q,R)$

			BKB		**CPLEX**			**HTS-DS**		
$K(n,q,R)$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	**LB**	**UB**	**Sol**	**GAP(%)**	**T(s)**	**Best**	**Avg**	**T(s)**
K(6,3,1)	729	9477	71	**73**	81	29.0%	TL	**73**	**73**	16.6
K(10,1,1)	1024	5120	107	**120**	136	71.0%	TL	127	127	19.4
K(10,1,2)	1024	28160	24	**30**	34	46.0%	TL	**30**	**30**	21.5
K(11,1,2)	2048	67584	37	**44**	90	66.0%	TL	**44**	**44**	37.6
K(11,1,3)	2048	236544	15	**16**	28	68.0%	TL	**16**	**16**	78.7
		Avg	50.8	56.6	73.8	-	-	**58**	**58**	34.76

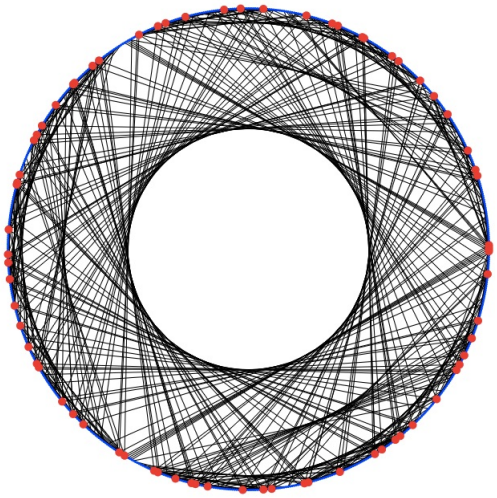
The results for the covering code problem in Hamming space are presented in Table 5.3. With the exception of $K(10,1,1)$, the searches had no difficulty in reaching the known upper bound, indicating the quality of the approach using the HTS algorithm.

Figure 5.2(a) contains the graph for the code $K_3(6,1)$. Although it is not possible to see all the edges (black part) and nodes (blue dots) due to the graph's dimension, it is possible to note its interesting topology as well as the intrinsic symmetry and characteristics. Figure 5.2(b) shows the solution found by our algorithm. The edges between non-dominated vertices were omitted.

Figure 5.2: Representation of the code $K_3(6,1)$.

5.2(a): Original graph

5.2(b): Solution's graph



5.3.3 Sensitivity analyses

As in previous chapter, we conducted a sensitivity analysis and on the impact of the method parameters. Starting from the same standard configuration as described in the Section 5.3.1, we modified one parameter and design choice at a time (OFAT approach) to evaluate its effect. The following configurations were considered:

Standard. Standard configuration described in Section 5.3.1.

A. No Reduced Problem. The reduced problem and IP solver is disabled.

B. $T_{\mathbf{MAX}}$. longer time limit for the reduced problem: $T_{\text{MAX}} = 5$sec.

C. ↓ Perturbation. Lower level of perturbation: $\rho = 0.1$.

D. ↑ Perturbation. Higher level of perturbation: $\rho = 0.3$.

E. ↓ Tabu. Shorter tabu tenure: $N_{\text{TABU}} = 10$.

F. ↑ Tabu. Longer tabu tenure: $N_{\text{TABU}} = 40$.

All configurations were run ten times for each code in RT space. Table 5.4 presents the final average for the best and average solution, (Avg_{B}) and (Avg_{A}) respectively, as well as the average time of each configuration.

	$\mathbf{Avg_B}$	$\mathbf{Avg_A}$	**T(s)**
Standard	**109.2**	**109.4**	38.6
A. No Reduced Problem	111.3	1114.4	**22.7**
B. ↑ T_{MAX}	**109.2**	**109.4**	97.3
C. ↓ Tabu	111.8	111.8	33.6
D. ↑ Tabu	111.7	111.8	35.9
E. ↓ Perturbation	109.5	109.8	32.3
F. ↑ Perturbation	109.3	**109.4**	35.5

Table 5.4: Analysis of HTS-DS components for the covering code problem

These experiments highlight the contribution of the reduced problem. They also should that the algorithm is sensitive to the adjustment of the tabu tenure.

Due to the major contributions of the mathematical programming solver in the reduced problem, we tried to increase T_{MAX} for this mechanism (configurations A and B). However, the solutions remained the same, except for code $K_2^{RT}(2, 6, 7)$, which improved from 21 to 20 with this new time limit. Thus, we maintained the configurations standard, and found that configuration A leads to a higher computational time with the same solution quality.

The impact of the tabu tenure in this problem is greater than the other settings, suggesting that the changes in this parameters influence the search, improving or worsening the quality of the solution, drastically.

5.4 Concluding Remarks

In this chapter, we applied the HTS-DS to solve covering code problems, with the main objective of finding upper bounds for the codes in RT and Hamming spaces. The solutions of the ILP solver require a high computational effort. Due to the inherent symmetry, many alternative optimal solutions exist, and it is typically a challenge to solve these codes, as reported in the work of Jans & Degraeve (2008) leading to an impact on the solution times using the ILP solver. By applying our method, the HTS-DS can produce new bounds for important codes.

Improving the upper bound of R-covering in Hamming spaces and RT spaces is usually a difficult task. Evidence for this is that for many codes $K_q^{RT}(m, s, R)$, the exact value is not yet known. Thus, these results show the importance of a computational approach, such as the one used in this work, since the algorithm was able to improve the upper bound for 32 out of the 48 codes. Several lower and upper bounds established for the smallest cardinality of covering code in an RT-space in the work of Castoldi & Carmelo (2015) were improved.

Our hybrid method creates a reduced problem of fixing a specific variable set. This can partly break the symmetry and decrease the solution times. Fixing one variable allows the branch-and-bound method to prune a node that leads to any of the multiple optimal solutions and without needs to evaluate all them. Still, symmetry issues should be investigated further in the future.

Finally, we showed how the hybrid algorithms for the dominating set perform in this special case and whether the problem of symmetry can be tackled within these algorithms.

6
Conclusions and Future Works

Combinatorial optimization is a branch of applied mathematics and computer science dealing with problems in which the minimum (or maximum) of functions defined over a discrete domain are sought. Some of these problems have received special attention from researchers because of their ability to model a large number of phenomena in practical applications. This thesis focused on an important problem class in combinatorial optimization: the dominating set problem and its variants, as well as the covering code problem.

This research topic, the DS problem, has received a good deal of attention in recent years due to its enormous potential, along with emerging applications in machine learning and social network analysis. First of all, we exploited the past search history and the *frequency* of some vertices in the dominating set to fix variables in the sub-problem. This strategy is closely related to the Construct, Merge, Solve & Adapt (CMSA) approach described in Blum et al. (2016). Moreover, other instance and solution metrics (e.g., ratio between weight and degree) may be used to further guide the search. Second, the relations between heuristic search and mathematical programming techniques can certainly be better exploited, by possibly sharing information or forming other types of subproblems.

Our experimental analyses show that considering a new neighborhood based on the swap of two vertices enhances the search performance and allows to obtain solutions of better quality. In addition, our *periodic ramp-up* strategy allows a better diversification of the search. The MIP-based approach of fixing variables allows the method to focus further on high-quality solutions to find a good trade-off between solution quality and computational effort. Therefore, the proposed matheuristic helps to obtain solutions in an acceptable time frame. Our method is competitive with those found in the literature. We showed that this hybrid approach is, in general, more effective in terms of solution quality and/or running time since they benefit from synergy.

New hybridizations between exact and heuristic approaches can be attempted to exploit the capabilities of both methods. Alternative forms of perturbation can be investigated for the connected version. The procedures contained in the proposed hybrid algorithm could also be extended to successfully solve similar problem to the dominating set problem and other variants.

Further contributions of our work include the improvement of some

bounds for the covering code problems. These problems need a method which more efficiently tackles the symmetries that are inherent to the problem.

Finally, the current machine learning literature, and especially the study of graphs arising from social networks, opens the way to very large scale problems, with possibly millions of vertices. These deserve careful study. Solution methods for such problems need to be carefully crafted to retain only essential search components working in linear or log-linear complexity. Overall, these are all important research avenues that can be explored in the near future.

Bibliography

Ambühl, C., Erlebach, T., Mihalák, M. & Nunkesser, M. (2006), Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs, *in* 'APPROX-RANDOM', Vol. 4110, Springer-Verlag, Berlin, Heidelberg, pp. 3–14.

Blum, C., Pinacho, P., López-Ibáñez, M. & Lozano, J. (2016), 'Construct, Merge, Solve & Adapt. A new general algorithm for combinatorial optimization', *Computers & Operations Research* **68**, 75–88.

Blum, J., Ding, M., Thaeler, A. & Cheng, X. (2005), Connected dominating set in sensor networks and MANETs, *in* D.-Z. Du & P. M. Pardalos, eds, 'Handbook of Combinatorial Optimization', Springer US, Boston, MA, pp. 329–369.

Bonato, A., Lozier, M., Mitsche, D., Pérez-Giménez, X. & Pralat, P. (2014), 'The domination number of on-line social networks and random geometric graphs', *CoRR*.

Bouamama, S. & Blum, C. (2016), 'A hybrid algorithmic model for the minimum weight dominating set problem', *Simulation Modelling Practice and Theory* **64**, 57–68.

Bouamama, S., Blum, C. & Boukerram, A. (2012), 'A population-based iterated greedy algorithm for the minimum weight vertex cover problem', *Applied Soft Computing* **12**(6), 1632–1639.

Carnielli, W. A. (1985), 'On covering and coloring problems for rook domains', *Discrete Mathematics* **57**(1-2), 9–16.

Castoldi, A. G. & Carmelo, E. L. M. (2015), 'The covering problem in Rosenbloom-Tsfasman spaces', *Electronic Journal of Combinatorics* **22**(3), P3.30.

Chen, N., Meng, J., Rong, J. & Zhu, H. (2004), 'Approximation for dominating set problem with measure functions', *Computing and Informatics* **23**(1), 37–49.

Chlebík, M. & Chlebíková, J. (2008), 'Approximation hardness of dominating set problems in bounded degree graphs', *Information and Computation* **206**(11), 1264–1275.

Cohen, G., Honkala, I., Litsyn, S. & Lobstein, A. (1997), *Covering codes*, North-Holland Mathematical Library, Elsevier Science.

Cygan, M., Pilipczuk, M. & Wojtaszczyk, J. O. (2011), 'Capacitated domination faster than $o(n^2)$', *Information Processing Letters* **111**(23), 1099–1103.

Dagdeviren, Z. A., Aydin, D. & Cinsdikici, M. (2017), 'Two population-based optimization algorithms for minimum weight connected dominating set problem', *Applied Soft Computing* **59**, 644–658.

Dai, D. & Yu, C. (2009), 'A $5 + \epsilon$-approximation algorithm for minimum weighted dominating set in unit disk graph', *Theoretical Computer Science* **410**(8-10), 756–765.

Dinh, T. N., Shen, Y., Nguyen, D. T. & Thai, M. T. (2012), 'On the approximability of positive influence dominating set in social networks', *Journal of Combinatorial Optimization* **27**(3), 487–503.

Elbassioni, K., Jelić, S. & Matijević, D. (2012), 'The relation of connected set cover and group steiner tree ', *Theoretical Computer Science* **438**, 96–101.

Erlebach, T. & Mihalák, M. (2010), A $(4+\epsilon)$-approximation for the minimum-weight dominating set problem in unit disk graphs, *in* E. Bampis & K. Jansen, eds, 'Approximation and Online Algorithms', Springer, Berlin, Heidelberg, pp. 135–146.

Fan, N. & Watson, J.-P. (2012), Solving the connected dominating set problem and power dominating set problem by integer programming, *in* G. Lin, ed., 'Combinatorial Optimization and Applications', Springer, Berlin, Heidelberg, pp. 371–383.

Fernau, H., Kneis, J., Kratsch, D., Langer, A., Liedloff, M., Raible, D. & Rossmanith, P. (2011), 'An exact algorithm for the maximum leaf spanning tree problem', *Theoretical Computer Science* **412**(45), 6290–6302.

Fomin, F. V., Kratsch, D. & Woeginger, G. J. (2005), Exact (exponential) algorithms for the dominating set problem, *in* J. Hromkovič, M. Nagl & B. Westfechtel, eds, 'Graph-Theoretic Concepts in Computer Science', Springer, Berlin, Heidelberg, pp. 245–256.

Fujie, T. (2003), 'An exact algorithm for the maximum leaf spanning tree problem', *Computers & Operations Research* **30**(13), 1931–1944.

Garey, M. R. & Johnson, D. S. (1990), *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA.

Gendron, B., Lucena, A., da Cunha, A. S. & Simonetti, L. (2014), 'Benders Decomposition, Branch-and-Cut, and Hybrid Algorithms for the Minimum Connected Dominating Set Problem', *INFORMS Journal on Computing* **26**(4), 645–657.

Glover, F. & Hao, J.-K. (2011), 'The case for strategic oscillation', *Annals of Operations Research* **183**(1), 163–173.

Goddard, W. & Henning, M. A. (2013), 'Independent domination in graphs: A survey and recent results', *Discrete Mathematics* **313**(7), 839 – 854.

Grandoni, F. (2006), 'A note on the complexity of minimum dominating set', *Journal of Discrete Algorithms* **4**(2), 209–214.

Guha, S. & Khuller, S. (1998), 'Approximation algorithms for connected dominating sets', *Algorithmica* **20**(4), 374–387.

Haynes, T. W., Hedetniemi, S. T. & Slater, P. J. (1998), *Fundamentals of domination in graphs*, 208 ed., Marcel Dekker Inc., New York, NY, USA.

Hedar, A. R. & Ismail, R. (2012), 'Simulated annealing with stochastic local search for minimum dominating set problem', *International Journal of Machine Learning and Cybernetics* **3**(2), 97–109.

Hedetniemi, S., Laskar, R. & Pfaff, J. (1986), 'A linear algorithm for finding a minimum dominating set in a cactus', *Discrete Applied Mathematics* **13**(2-3), 287–292.

Hopcroft, J. & Tarjan, R. (1973), 'Algorithm 447: efficient algorithms for graph manipulation', *Communications of the ACM* **16**(6), 372–378.

Huang, Y., Gao, X., Zhang, Z. & Wu, W. (2009), 'A better constant-factor approximation for weighted dominating set in unit disk graph', *Journal of Combinatorial Optimization* **18**(2), 179–194.

Jans, R. & Degraeve, Z. (2008), 'A note on a symmetrical set covering problem: The lottery problem', *European Journal of Operational Research* **186**(1), 104–110.

Johnson, D. S. (1974), 'Approximation algorithms for combinatorial problems', *Journal of Computer and System Sciences* **9**(3), 256–278.

Jovanovic, R. & Tuba, M. (2013), 'Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem', *Computer Science and Information Systems* **10**(1), 133–149.

Jovanovic, R., Tuba, M. & Simian, D. (2010), 'Ant colony optimization applied to minimum weight dominating set problem', *Proceedings of the 12th WSEAS International Conference on Automatic Control, Modelling & Simulation* pp. 322–326.

Kéri, G. (2018), 'Tables for Bounds on Covering Codes'. URL `http://old.sztaki.hu/~keri/codes/`, [Online; accessed 01-January-2018].

Kim, D., Li, D., Asgari, O., Li, Y. & Tokuta, A. O. (2013), *A dominating set based approach to identify effective leader group of social network*, Springer, Berlin, Heidelberg, pp. 841–848.

Koetter, R. & Kschischang, F. R. (2007), 'Coding for errors and erasures in random network coding', *IEEE International Symposium on Information Theory - Proceedings* pp. 791–795.

Li, J. & Jin, Y. (2015), A ptas for the weighted unit disk cover problem, *in* M. M. Halldórsson, K. Iwama, N. Kobayashi & B. Speckmann, eds, 'Automata, Languages, and Programming', Springer, Berlin, Heidelberg, pp. 898–909.

Li, P. C. & van Rees, G. H. J. (2002), 'Lotto design tables', *Journal of Combinatorial Designs* **10**(5), 335–359.

Li, R., Hu, S., Gao, J., Zhou, Y., Wang, Y. & Yin, M. (2017), 'Grasp for connected dominating set problems', *Neural Computing and Applications* **28**(1), 1059–1067.

Li, Y., Thai, M. T., Wang, F., Yi, C.-W., Wan, P.-J. & Du, D.-Z. (2005), 'On greedy construction of connected dominating sets in wireless networks', *Wireless Communications and Mobile Computing* **5**(8), 927–932.

Liedloff, M., Todinca, I. & Villanger, Y. (2014), 'Solving capacitated dominating set by using covering by subsets and maximum matching', *Discrete Applied Mathematics* **168**, 60–68.

Linderoth, J., Margot, F. & Thain, G. (2009), 'Improving bounds on the football pool problem by integer programming and high-throughput computing', *INFORMS Journal on Computing* **21**(3), 445–457.

Liu, C.-H., Poon, S.-H. & Lin, J.-Y. (2015), 'Independent dominating set problem revisited', *Theoretical Computer Science* **562**, 1–22.

Lucena, A., Maculan, N. & Simonetti, L. (2010), 'Reformulations and solution algorithms for the maximum leaf spanning tree problem', *Computational Management Science* **7**(3), 289–311.

Maniezzo, V., Stützle, T. & Voß, S., eds (2009), *Matheuristics: Hybridizing metaheuristics and mathematical programming*, Annals of Information Systems, Springer, New York.

Mendes, C., Monte Carmelo, E. L. & Poggi, M. (2010), 'Bounds for short covering codes and reactive tabu search', *Discrete Applied Mathematics* **158**, 522–533.

Nacher, J. C. & Akutsu, T. (2016), 'Minimum dominating set-based methods for analyzing biological networks.', *Methods* **102**, 57—-63.

Ore, O. (1962), *Theory of graphs*, American Mathematical Society colloquium publications, American Mathematical Society.

Östergård, P. R. (1997), 'Constructing covering codes by tabu search', *Journal of Combinatorial Designs* **5**(1), 71–80.

Potluri, A. & Singh, A. (2011), Two hybrid meta-heuristic approaches for minimum dominating set problem, *in* B. K. Panigrahi, P. N. Suganthan, S. Das & S. C. Satapathy, eds, 'Swarm, Evolutionary, and Memetic Computing', Springer, Berlin, Heidelberg, pp. 97–104.

Potluri, A. & Singh, A. (2012), A greedy heuristic and its variants for minimum capacitated dominating set, *in* M. Parashar, D. Kaushik, O. F. Rana, R. Samtaney, Y. Yang & A. Zomaya, eds, 'Contemporary Computing', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 28–39.

Potluri, A. & Singh, A. (2013), 'Hybrid metaheuristic algorithms for minimum weight dominating set', *Applied Soft Computing* **13**(1), 76–88.

Quistorff, J. (2007), 'On Rosenbloom and Tsfasman's generalization of the Hamming space', *Discrete Mathematics* **307**(21), 2514–2524.

Resende, M. & Pardalos, P. (2008), *Handbook of Optimization in Telecommunications*, Springer US.

Rosenbloom, M. Y. & Tsfasman, M. A. (1997), 'Codes for the m-metric.', *Problems of Information Transmission* **33**(1), 45–52.

Ruan, L., Du, H., Jia, X., Wu, W., Li, Y. & Ko, K. I. (2004), 'A greedy approximation for minimum connected dominating sets', *Theoretical Computer Science* **329**(1-3), 325–330.

Schaudt, O. & Schrader, R. (2012), 'The complexity of connected dominating sets and total dominating sets with specified induced subgraphs', *Information Processing Letters* **112**(24), 953–957.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H. & Dueck, G. (2000), 'Record breaking optimization results using the ruin and recreate principle', *Journal of Computational Physics* **159**(2), 139–171.

Shyu, S. J., Yin, P.-Y. & Lin, B. M. (2004), 'An ant colony optimization algorithm for the minimum weight vertex cover problem', *Annals of Operations Research* **131**(1), 283–304.

van Rooij, J. M. & Bodlaender, H. L. (2011), 'Exact algorithms for dominating set', *Discrete Applied Mathematics* **159**(17), 2147–2164.

Vidal, T., Crainic, T., Gendreau, M. & Prins, C. (2015), 'Time-window relaxations in vehicle routing heuristics', *Journal of Heuristics* **21**(3), 329–358.

Vinayagam, A., Gibson, T. E., Lee, H.-J., Yilmazel, B., Roesel, C., Hu, Y., Kwon, Y., Sharma, A., Liu, Y.-Y., Perrimon, N. & Barabási, A.-L. (2016), 'Controllability analysis of the directed human protein interaction network identifies disease genes and drug targets', *Proceedings of the National Academy of Sciences* **113**(18), 4976–4981.

Wang, D., Li, T. & Ogihara, M. (2012), Generating pictorial storylines via minimum-weight connected dominating set approximation in multi-view graphs, *in* 'AAAI Conference on Artificial Intelligence'.

Wang, F., Camacho, E. & Xu, K. (2009), Positive influence dominating set in online social networks, *in* D.-Z. Du, X. Hu & P. M. Pardalos, eds, 'Combinatorial Optimization and Applications', Springer, Berlin, Heidelberg, pp. 313–321.

Wang, F., Du, H., Camacho, E., Xu, K., Lee, W., Shi, Y. & Shan, S. (2011), 'On positive influence dominating sets in social networks', *Theoretical Computer Science* **412**(3), 265–269.

Wang, H., Zheng, H., Browne, F. & Wang, C. (2014), Minimum dominating sets in cell cycle specific protein interaction networks, *in* '2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)', pp. 25–30.

Wang, Y., Cai, S. & Yin, M. (2017), 'Local search for minimum weight dominating set with two-level configuration checking and frequency based scoring function', *Journal of Artificial Intelligence Research* **58**, 267–295.

Wikimedia (2017), 'Multiplex human hiv-1 protein-protein interaction network'. URL `https://commons.wikimedia.org/w/index.php?title=File:Multiplex_Human_HIV-1_protein-protein_interaction_network_(edge-colored_visualization).png`, [Online; accessed 01-January-2018].

Wu, X., Lü, Z. & Galinier, P. (2017), 'Restricted swap-based neighborhood search for the minimum connected dominating set problem', *Networks* **69**(2), 222–236.

Wuchty, S. (2014), 'Controllability in protein interaction networks', *Proceedings of the National Academy of Science* **111**, 7156–7160.

Yu, J., Wang, N., Wang, G. & Yu, D. (2013), 'Connected dominating sets in wireless ad hoc and sensor networks-A comprehensive survey', *Computer Communications* **36**(2), 121–134.

Zhu, X., Wang, W., Shan, S., Wang, Z. & Wu, W. (2012), 'A PTAS for the minimum weighted dominating set problem with smooth weights on unit disk graphs', *Journal of Combinatorial Optimization* **23**(4), 443–450.

Zou, F., Wang, Y., Xu, X. H., Li, X., Du, H., Wan, P. & Wu, W. (2011), 'New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs', *Theoretical Computer Science* **412**(3), 198–208.