Master's Thesis

Biased Random-Key Genetic Algorithms for the Minimum Broadcast Time Problem

Alfredo Lima Moura Silva
alfredolms@ic.ufal.br

Advisers:

Dr. Rian Gabriel Santos Pinheiro
Dr. Bruno Costa e Silva Nogueira

Maceió
November 12, 2021

Alfredo Lima Moura Silva

Biased Random-Key Genetic Algorithms for the Minimum Broadcast Time Problem

A thesis submitted by Alfredo Lima Moura Silva in partial fulfillment of the requirements for the degree of Master of Science in Informatics at the Federal University of Alagoas, Computing Institute.

Advisers:

Dr. Rian Gabriel Santos Pinheiro

Dr. Bruno Costa e Silva Nogueira

Maceió
November 12, 2021

A thesis submitted by Alfredo Lima Moura Silva in partial fulfillment of the requirements for the degree of Master of Science in Informatics at the Federal University of Alagoas, Computing Institute, approved by the examination committee which signs bellow.

Dr. Rian Gabriel Santos Pinheiro - Adviser
Computing Institute
Federal University of Alagoas

Dr. Bruno Costa e Silva Nogueira - Adviser
Computing Institute
Federal University of Alagoas

Dr. André Luiz Lins de Aquino - Examiner
Computing Institute
Federal University of Alagoas

Dr. Fábio Protti - Examiner
Computing Institute
Fluminense Federal University

Maceió
November 12, 2021

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

S586b Silva, Alfredo Lima Moura.
Biased random-key genetic algorithms for the minimum broadcast time problem / Alfredo Lima Moura Silva. – 2021.
61 f. : il.

Orientador: Rian Gabriel Santos Pinheiro.
Coorientador: Bruno Costa e Silva Nogueira.
Dissertação (mestrado em Informática) - Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2021.

Bibliografia: f. 53-58.
Apêndices: f. 59-61.

1. Otimização combinatória. 2. Tempo mínimo de transmissão. 3. Metaheurística. 4. Chaves aleatórias enviesadas (Algoritmos genéticos). I. Título.

CDU: 004.421

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Informática – PPGI
Instituto de Computação/UFAL
Campus A. C. Simões BR 104-Norte Km 14 BL 12 Tabuleiro do Martins
Maceió/AL - Brasil CEP: 57.072-970 | Telefone: (082) 3214-1401

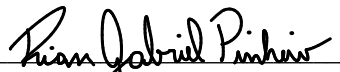
Folha de Aprovação

ALFREDO LIMA MOURA SILVA

BIASED RANDOM-KEY GENETIC ALGORITHMS FOR THE MINIMUM BROADCAST TIME PROBLEM

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas e aprovada em 12 de novembro de 2021.

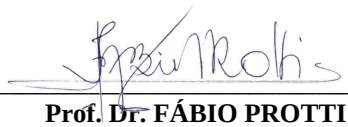
Banca Examinadora:



Prof. Dr. RIAN GABRIEL SANTOS PINHEIRO
UFAL – Instituto de Computação
Orientador

Prof. Dr. BRUNO COSTA E SILVA NOGUEIRA
UFAL – Instituto de Computação
Coorientador

Prof. Dr. ANDRE LUIZ LINS DE AQUINO
UFAL – Instituto de Computação
Examinador Interno



Prof. Dr. FÁBIO PROTTI
UFF- Universidade Federal Fluminense
Examinador Externo

Acknowledgement

I thank God for all the gifts and graces that I had received.

I would like to special acknowledge my advisors Rian Pinheiro and Bruno Nogueira. My mother Neide, stepfather Helder Fernando and grandmother Maria José (*in memoriam*) for all the support.

I thank for the new friends I made during this time. To Professors who are members of the examining board. Prof. Dr. Fábio Protti and Prof. Dr. André Luiz for their time dedicated to reading and contributing to making this work better. To the Federal University of Alagoas (Ufal), OptLab, and EDGE - Innovation Center for every opportunity, they were essential pieces in my academic background.

The power of compound interest the most powerful force in the universe.

– Einstein, *A.*

Resumo

O problema TEMPO MÍNIMO DE TRANSMISSÃO (TMT) é um problema conhecido de disseminação de dados, com o objetivo é encontrar um esquema de transmissão que minimize o número de passos necessários para executar a operação de transmissão.

O TEMPO MÍNIMO DE TRANSMISSÃO COM PESO (TMTP) é uma generalização do TMT, de forma que cada operação tem um custo. Ambos os problemas têm diversas aplicações em sistemas distribuídos, por exemplo, o processo de atualização de dispositivos em uma rede ponto-a-ponto.

Este trabalho propõe Algoritmos Genéticos de Chave-Aleatória Enviesados (AGCAE) para o TMT e o TMTP. Um algoritmo híbrido (AGCAE + Programação Linear Inteira) para o TMT. Algoritmos para calcular um limite inferior para o TMT e o TMTP. Uma abordagem de refinamento, e métodos para criar instâncias com ótimos conhecidos para TMT e TMTP. Além disso, um método para diminuir os grafos para o TMTP.

Foi realizado experimentos com o AGCAE em instâncias comumente utilizadas na literatura e também em instâncias sintéticas massivas (até 1000 vértices), o que permite cobrir muitas possibilidades de topologias reais da indústria.

A proposta comparou métodos exatos e heurísticas do estado-da-arte dos problemas. Os algoritmos superaram as heurísticas mais conhecidas para TMT e TMTP.

Os experimentos demonstraram que estes algoritmos são uma boa alternativa quando métodos exatos não podem ser aplicados. Para todas as instâncias do TMT com valor ótimo conhecido, os algoritmos alcançaram o valor ótimo ou no máximo uma unidade para encontrar o tempo mínimo de transmissão. Para todas as instâncias do TMTP, as abordagens alcançaram ou melhoraram os resultados da literatura.

Keywords: Otimização combinatória, Tempo Mínimo de Transmissão, Metaheurísticas, Algoritmos Genéticos de Chave Aleatória Enviesados.

Abstract

The MINIMUM BROADCAST TIME (MBT) is a well-known data dissemination problem whose goal is to find a broadcast scheme that minimizes the number of steps needed to execute the broadcast operation. The WEIGHTED MINIMUM BROADCAST TIME (WMBT) is a generalization of the MBT, such that each operation has a cost. Both problems have many applications in distributed systems, e.g., the devices update process in a peer-to-peer network.

This work proposes Biased Random-Key Genetic Algorithms (BRKGA) for the MBT and WMBT. A hybrid algorithm (BRKGA + Integer Linear Programming) for the MBT. Algorithms to calculate a lower bound for the MBT and WMBT. A refinement approach and methods to create instances with known optima for the MBT and WMBT. Moreover, reducing rules for the WMBT.

We carry out experiments with our BRKGA on instances commonly used in the literature and also on massive synthetic instances (up to 1000 vertices), allowing us to cover many possibilities of real industry topologies. Our proposal compared state-of-the-art exact methods and heuristics. Our algorithms outperformed the best-known heuristics for the MBT and WMBT. The experiments demonstrated they are a very good alternative when exact methods cannot be applied. For all instances for the MBT with known optima value, our approaches either attained the optimal value or missed it by at most one broadcast step. For all instances for the WMBT, our approaches attained or improved the results of literature.

Keywords: Combinatorial Optimization, Minimum Broadcast Time, Metaheuristics, Biased Random-Key Genetic Algorithms.

Contents

Figure List . . . viii
Table List . . . ix
Algorithm List . . . x

1 Introduction **1**
1.1 Motivation . . . 2
1.2 Message-passing systems . . . 3
1.3 Optimization . . . 5
1.3.1 Exact algorithms . . . 5
1.3.2 Approximate algorithms . . . 5
1.3.3 Metaheuristics . . . 6
1.3.4 Matheuristics . . . 8
1.4 Objectives . . . 8
1.5 Structure of the Thesis . . . 9

2 Minimum Broadcast Time **11**
2.1 Related Work . . . 12
2.2 Algorithmic approaches for the MBT . . . 13
2.2.1 Synthetic instances for the MBT . . . 13
2.2.2 Lower bound algorithm . . . 14
2.2.3 BRKGA decoders for the MBT . . . 16
2.2.4 Proposed matheuristic . . . 19
2.3 Computacional results of the MBT . . . 23
2.3.1 Instances . . . 23
2.3.2 Parameter settings and experimental protocol . . . 24
2.3.3 Experiments on Harary Graphs . . . 25
2.3.4 Experiments on others instances from literature . . . 27
2.3.5 Experiments on Small-World and synthetic instances . . . 27
2.3.6 Experiments with SCHA decoding and the proposed matheuristic . . . 30

3 Weighted Minimum Broadcast Time **33**
3.1 Related Work . . . 35
3.2 Algorithmic approaches for the WMBT . . . 36
3.2.1 Mathematical models for WMBT . . . 36
3.2.2 Lower bound algorithm for the WMBT . . . 39
3.2.3 Exact greedy algorithm for the WMBT for forest . . . 39
3.2.4 Greedy algorithm for the WMBT for general instances . . . 40
3.2.5 Reducing rules . . . 40
3.2.6 BRKGA for the WMBT . . . 43

3.3 Computacional results of the WMBT 44
3.3.1 Instances 46
3.3.2 Parameter settings and experimental protocol 46
3.3.3 Comparing the deterministic algorithms 47
3.3.4 Validating the reducing rules 48
3.3.5 Comparing the metaheuristics 49

4 Final Considerations **51**
4.1 Future works 52
4.2 Scientific production 52

References **53**

A Detailed instances **59**
A.1 Instances for the MBT. 59
A.2 Instances for the WMBT. 61

List of Figures

1.1 Example scenario . . . 1
1.2 Feasible broadcast solution. . . . 2
1.3 Optimal broadcast solution. . . . 2
1.4 Network with connection time and transmission time . . . 4
1.5 Telephone and postal models . . . 4
1.6 BRKGA . . . 8
1.7 Overview of this work. . . . 10
2.1 Examples of binomial trees. . . . 13
2.2 Building a synthetic instance. . . . 14
2.3 Building a synthetic instance with 2 sources. . . . 14
2.4 Input graph . . . 17
2.5 FRFS decoding procedure. . . . 18
2.6 FRFS-SCHA decoding procedure. . . . 19
2.7 Proposed matheristic: pool of solutions merging and solving process. . . . 21
3.1 Example of weighted-vertex model. . . . 33
3.2 Example of weighted-edge model. . . . 34
3.3 Example of weighted-edge-and-vertex model. . . . 34

List of Tables

2.1 List of articles about MBT in the literature. 12
2.2 Chromosomes of input 17
2.3 Range considered by IRACE and best parameter settings obtained. 24
2.4 Comparative results of ACS, Treeblock and BRKGA_FRFS. 25
2.5 Comparative results of BRKGA_FRFS and ILPs. 26
2.6 Comparative results of NEWH, NTBA, ILP, ACS and BRKGA 27
2.7 Comparative results of ACS, ILP and BRKGA_FRFS 28
2.8 Comparative results of BRKGAs and hybrids 30
3.1 List of articles about WMBT in the literature. 35
3.2 Range considered by IRACE and best parameter settings obtained. 46
3.3 Comparative results of deterministic algorithms. 48
3.4 Comparative results of BRKGA-DJs with and without RR 49
3.5 Comparative results of BRKGA-MSFs with and without RR 49
3.6 Comparative results of BRKGAs with and without RR 49
3.7 Comparative results of ACS, BRKGAs and HARUTYUNYAN-WSCHA 49

A.1 Description of the test instances. 59
A.2 Description of the test instances. 61

List of Algorithms

1 Lower bound algorithm for the MBT. 15
2 FRFS decoder. 16
3 SCHA. 18
4 FRFS-SCHA decoder. 19
5 Proposed matheuristic for the MBT. 22
6 Greedy algorithm for Weighted-Vertex-MBT 36
7 Lower bound algorithm for WMBT. 40
8 WMBT for forest instances. 41
9 Exact greedy algorithm for WMBT 42
10 Algorithm to remove edges of graph 42
11 BRKGA for the WMBT. 43
12 DJ decoder 44
13 MSF decoder 45
14 Create a random weighted graph with optima known. 47

1

Introduction

Broadcasting is the distribution of data in a network. According to Hedetniemi et al. (1988), it has been studied since the early 1950s with Bavelas (1950). He studied the effectiveness of different communication patterns in helping small groups solve common tasks. He considered measures such as the time required to perform an information transmission task and several problems, e.g., the gossiping problem. In the 1970s, the broadcasting problem became more and more popular among researchers.

The MINIMUM BROADCAST TIME (MBT) problem (Farley et al., 1979) consists of finding the shortest sequence of messages that allows the data to reach every node in the network. At each time step, every node can transmit the message to at most a single neighbor. The goal is to find a broadcast scheme that minimizes the number of steps needed to execute the broadcast operation.

Figures 1.1-1.3 show a simple example of the MBT, where a dashed vertex indicates that the vertex did not receive the message, and an arrow indicates a message sent at time t. In this example, $V_0 = \{g_1\}$ model a set of network gateways, whereas $D = \{d_1, d_2, d_3\}$ a set of common network devices. A simple (feasible) solution for the presented problem is depicted in Figure 1.2 and suggests a three-step broadcast. However, Figure 1.3 shows an optimal solution that requires only two steps to broadcast the data throughout the network.

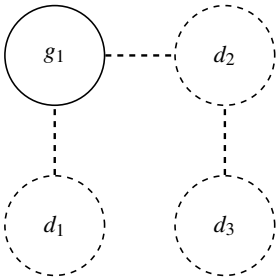


Figure 1.1: Example scenario

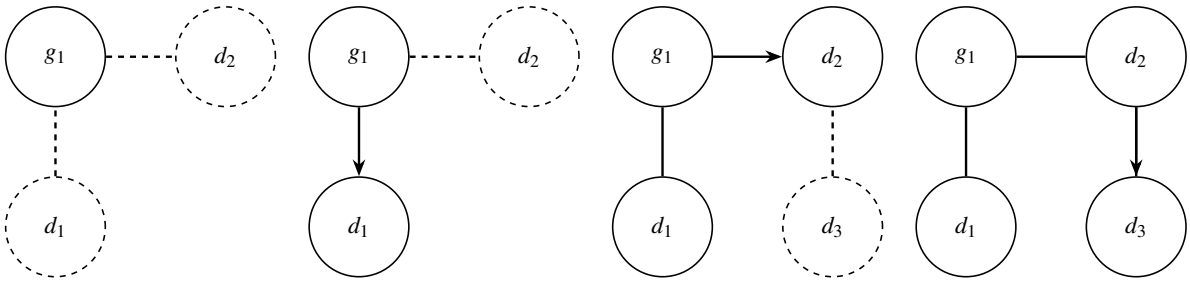


Figure 1.2: Feasible broadcast solution.

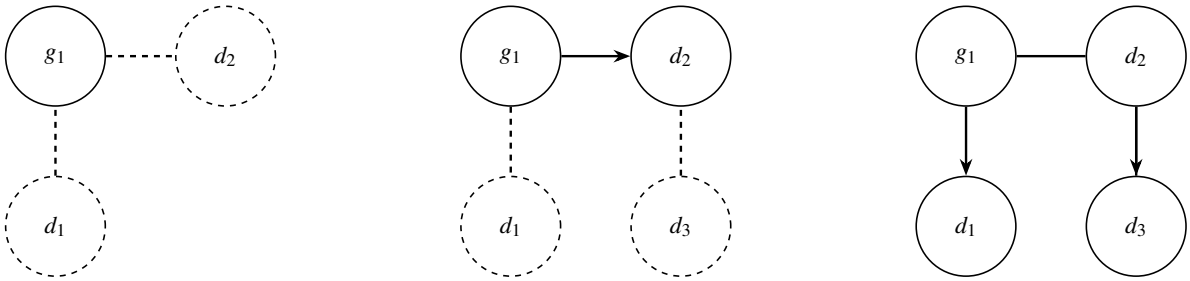


Figure 1.3: Optimal broadcast solution.

1.1 Motivation

The MBT and its generalization, the WEIGHTED MINIMUM BROADCAST TIME (WMBT), have many applications in distributed systems, such as the Internet of Things (IoT) and Wireless Sensor Networks (WSNs) (Shang et al., 2010). These applications rely mainly on large-scale machine communications. Hence, they need data dissemination techniques that combine high reliability with low communication latency.

Besides distributed systems, the MBT has many other applications, including communication among telephone networks (Ivanova, 2019), surveillance and reconnaissance (Dekker, 2002), and Direct Memory Access (DMA) (Lazard, 1992). Next, we present applications in robotics, satellite networks and an industrial with Bluetooth.

In the context of swarm robotics, the MBT can be used for solving the Freeze-Tag Problem (Bucantanschi et al., 2007; Keshavarz et al., 2011). The problem is to devise a schedule to activate all robots in the minimum amount of time. Activation of robots, other than the initial robot, only occurs if an active robot physically moves to the location of an inactive robot.

Chu and Chen (2018) described a practical application on satellite networks. Antennas transmit data over a long distance in a directed way. Each vertex represents a satellite or a base station, and transmissions only occur one at a time. Each satellite has a transmission and setup time, such that the first considers the time to transmit the data and the second considers the time to redirect the antenna.

The original motivation for our work on the MBT was a network problem from an industrial partner, which asked us to optimize their device update process. In their network, devices used Bluetooth for intranet communication. Some of these devices (gateways) had a GPRS board for

external communication. In preliminary experiments, they tested the update process considering that the gateways had a firmware image of 200KB to broadcast. Two broadcasting techniques were tested: peer-to-peer and Bluetooth mesh. In peer-to-peer, the nodes work in a MBT style such that only one transmission occurs at a time per device, whereas in mesh this restriction does not exist and a device can update more than one neighbor at the same time. The experiments have shown that in peer-to-peer the update process took 2 minutes, and in Bluetooth mesh, it took 4.5 hours. The main reason for this difference is that the mesh technique demanded more bandwidth, which in turn resulted in more network congestion and packet loss. This observation was formally demonstrated in Robledo et al. (2020).

1.2 Message-passing systems

According to Tsou et al. (2013), the two most common models for studying message-passing systems are the telephone model (Slater et al., 1981; Su et al., 2010) and the postal model (Bar-Noy and Kipnis, 1994). In both models, the communication is made via calls between adjacent vertices, by transmitting a single message from one sender to one receiver. In the telephone model, each *call* takes one unit of time, and each vertex can only participate in one call at any time. The postal model incorporates a communication setup phase and latency time. It is introduced as the most appropriate for the message-passing system, which employs additional parameters $\alpha \geq 0$ and $\beta \geq 0$, called the setup phase and latency time, respectively.

In this model, each call consists of a setup phase, which takes α units times. Furthermore, a sender can start a new call to another receiver when the transmission phase of its last call finishes. For example, suppose that a vertex u wants to transmit a message first to v_1 and then to v_2 at time 0. In its first call, u sets up the connection to v_1 and after α units time sends the message to v_1. Thus, the call from u to v_1 will be completed after α units of time, but v_1 received message in $\alpha+\beta$ units of time. Since the setup phase of the first call is finishes at time α, the call from u to v_2 must be completed after $2 \cdot \alpha$ units of time, but v_2 receives the message in $2 \cdot \alpha + \beta$ units of time.

Rico-Gallego et al. (2019) present these models generics, considering a setup phase or connection time α and transmission time β. Where α is assumed to be a constant and β varies from edge to edge. Hence, the transmission time between the sender v and receiver u is $\beta_{v,u}$.

The telephone model considers that the device must be set up only once and then can transmit. That is, if the sender v transmits to u_1, u_2, ... and u_n, thus broadcast time is defined as:

$$b(v) = \alpha + \max_{i \in \{1,2,\ldots,n\}} ((\sum_{j=1}^{i} \beta_{v,u_j}) + b(u_i)).$$

On the other hand, the postal model considers that the device must be set up for each connection, and then it can transmit. That is, if the sender v transmits to u_1, u_2, ... and u_n, thus

broadcast time is defined as:

$$b(v) = \max_{i \in \{1,2,\ldots,n\}} (i \cdot \alpha + \beta_{v,u_i} + b(u_i)).$$

Figure 1.4 shows an example instance of the MBT. Figures 1.5.a and 1.5.b show the broadcast scheme for the telephone model and postal model with $\alpha_{g_1} = 2$, $\alpha_{d_1} = \alpha_{d_2} = \alpha_{d_3} = 0$, $\beta_{g_1,v_1} = 1$, $\beta_{g_1,v_2} = 2$, and $\beta_{g_1,v_3} = 3$. The ordering of transmissions is arbitrary. In Figure 1.5.a, the vertex g_1 transmits to v_1, v_2 and v_3. Note that g_1 has only one setup phase before the first sending. The remaining transmissions do not have this phase. On the other hand, in Figure 1.5.b (that illustrated the postal model), the vertex g_1 transmits to v_1, v_3 and v_2. Note that, before each transmission, there is a setup phase.

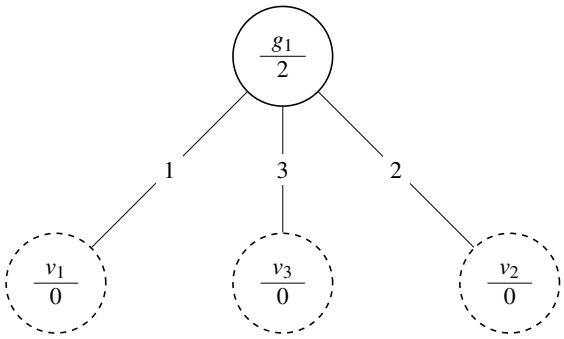
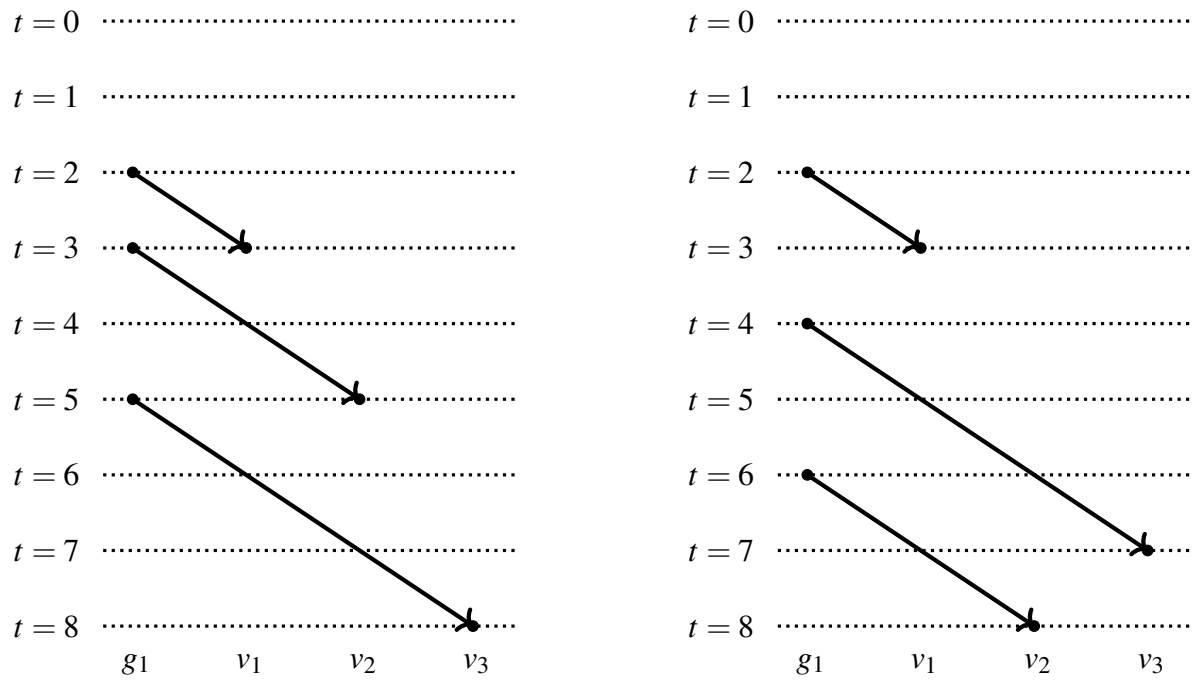


Figure 1.4: Network with connection time and transmission time



(a) Telephone model

(b) Postal model

Figure 1.5: Telephone and postal models

1.3 Optimization

Gomez et al. (2004) define optimization as "doing the most with the least" and Lockhart and Johnson (2000) define optimization as "the process of finding the most effective or favorable value or condition". *Optimization* is the act of finding the input parameters or arguments to a function that result in the minimum or maximum output of the function. Classical optimization models are mathematical programming, combinatorial optimization, constraint satisfaction, and nonanalytic. Mathematical programming can be divided into continuous, integer, and mixed for linear or nonlinear models (Talbi, 2009).

Combinatorial problems can be identified in the most diverse real applications such as planning, production, coordination, investment, transportation, communication, among others. Combinatorial problems can be identified in the most diverse real applications such as planning, production, coordination, investment, transportation, communication, and others. Furthermore, they can be examined, formulated and solved, using techniques from different areas of knowledge such as engineering, biology, economics and computer science. To solve one of these problems, you must maximize/minimize a given function with one or several variables, so that all constraints are satisfied. Those constraints are modeled in the form of equations or inequalities. However, solving some problems might be hard, since they can belong to the $\mathcal{NP}$-hard complexity class. In these cases, their resolution through exact methods is ineffective in instances of high dimensions. They need memory and processor time, so maybe it is not suitable practical to use them. But, approximate or heuristic methods can generate high-quality solutions in acceptable time for practical use, but there is no guarantee of finding a globally optimal solution.

1.3.1 Exact algorithms

Exact methods obtain optimal solutions and guarantee their optimality. For $\mathcal{NP}$-hard problems, we do not know a polynomial-time algorithm to solve them. We include some classical algorithms as dynamic programming, and branch-and-X family of algorithms (branch-and-bound, branch-and-cut, branch-and-price) (Padberg and Rinaldi, 1987; Barnhart et al., 1998).. Both methods perform the search by subdividing it into smaller problems of the same kind. Another exact approach is to transform combinatorial problems into linear programming (LP) and integer linear programming problems and then solve them using a mathematical programming solver.

1.3.2 Approximate algorithms

Approximate methods generate high-quality solutions in a reasonable time for practical use, but there is no guarantee of finding a globally optimal solution. In the class of approximate methods, two subclasses of algorithms may be distinguished: approximation algorithms and heuristic algorithms (Talbi, 2009).

Heuristics find “good” solutions on large-size problem instances. They allow obtaining acceptable performance at acceptable costs in a wide range of problems. In general, heuristics do not have an approximation guarantee on the obtained solutions. Specific heuristics are tailored and designed to solve a specific problem and/or instance. Greedy or constructive algorithms start from scratch (empty solution) and construct a solution by assigning values to one decision variable at a time, until a complete solution is generated (Gendreau and Potvin, 2010).

Approximation algorithms are similar to heuristics algorithms, but they ensure that the bound of the obtained solution is from the global optimum. An k-approximation algorithm generates an approximate solution not less than a factor k times the optimum solution (Vazirani, 2001).

1.3.3 Metaheuristics

Unlike heuristics that are designed to solve a specific problem, metaheuristics are general-purpose algorithms that can be applied to solve almost any optimization problem. They may be viewed as upper-level general methodologies that can be used as a guiding strategy in designing underlying heuristics to solve specific optimization problems. Metaheuristics allow tackling large-size problem instances by delivering satisfactory solutions in a reasonable time. There is no guarantee to find globally optimal solutions or even bounded solutions. Their use in many applications shows their efficiency and effectiveness to solve large and complex problems (Sorensen et al., 2017).

In designing a metaheuristic, two contradictory criteria must be taken into account: (i) the exploration of the search space (diversification) and (ii) the exploitation of the best solutions found (intensification). We can determine if a region is promising by the quality of its solutions. In intensification, a promising area is explored more deeply in the hope of finding better solutions. In diversification, unexplored areas should be visited to make sure that all regions of the search space are explored evenly and that the search is not limited to just a small number of regions.

Talbi (2009) classifies metaheuristics into two groups: single-solution-based and population-based metaheuristics. For single-solution-based metaheuristics, the main idea is to find high-quality solutions interactively. The initial solution is the first solution. It can be entirely random, greedy, or a combination of both. The neighborhood of a solution s is a set of similar solutions, which can be generated from s by applying predefined “small” perturbations. The algorithm can accept a quality improvement or move to worst solution. These features may vary for each metaheuristic.

The population-based metaheuristics approach maintains and improves multiple candidate solutions, often using population characteristics to guide the search. The candidates have a cooperative relationship, where this relationship designs from a kind of metaheuristic. There are such as ACO (ant colonies optimization), GA (genetic algorithms), GRASP (greedy adaptive search procedure), ILS (iterated local search), PSO (particle swarm optimization), SA (simulated annealing), TS (tabu search), and VNS (variable neighborhood search). Now, we will show an

overview of GA and detail the biased random-key genetic algorithm.

Genetic algorithms (GA) is a population-based metaheuristic that relies on the principles of natural selection. It can be viewed as an iterative improvement in a population of solutions. A GA usually has seven steps (Holland, 1975): (i) Initial population, (ii) Evaluation, (iii) Selection, (iv) Recombination, (v) Mutation, (vi) Replacement, and (vii) Repeat (ii)—(vi). First, the population is initialized and evaluated. Then, a new population of solutions is generated using some selection, followed by recombination and mutation procedures. Finally, this new population is integrated into the current one.

In a GA, each individual is referred as a chromosome, which encodes a candidate solution. A chromosome consists of a string of genes whose values are called alleles. An objective function associates a fitness value with each individual indicating its fitting to the problem. In the evolving process, individuals are selected to reproduce, and a crossover operator is used on the selected individuals to produce new offspring that make up the next generation. In addition, mutation operators are involved in this process. Finally, a replacement step is applied to determine which individuals of the population (parents and offspring) will survive.

The proposed GA is not similar to classical Genetic Algorithms (Holland, 1975), where individuals are represented using a binary array and modified by some mutation operator. Our algorithm is based on more recent implementations using random-keys encodings. In random-key genetic algorithms (RKGA), developed by Bean (1994), the chromosomes consist of vectors of real numbers generated randomly in the range $[0, 1)$. Besides, a deterministic algorithm called *decoder* processes the vector to calculate the fitness of the solution. RKGA does not make use of the standard mutation operator, where some alleles are changed with a given probability. Instead, new (mutant) solutions are randomly generated and introduced in the current population in each generation, in the same way as the initial population is created.

A biased random-key genetic algorithm (BRKGA) (Gonçalves and Resende, 2011) differs from an RKGA in the way parents are selected for crossover. As shown in Fig. 1.6, at each generation, the population is partitioned into two parts: elite and non-elite. The elite part contains the best solutions in the total population and is simply copied to the next population generation. The remaining elements are created by crossover or randomly generated (mutants). In the recombination phase, each new individual is generated by combining one individual selected from the elite set and another individual selected from the non-elite population. The elite parent has a higher probability of passing its genes to the offspring generation, i.e., a biased selection via the parameterized uniform crossover operator proposed by Spears and Jong (1991). This process is illustrated in Fig. 1.6, where $\rho_e = 0.7$ (70%) is the probability that the offspring will inherit each of its alleles from the best fit of the two parents. Gonçalves et al. (2014) made an empirical comparison of biased and unbiased random-keys genetic algorithms in four types of covering problems and has shown that the biased variant is faster than the original Bean's algorithm.

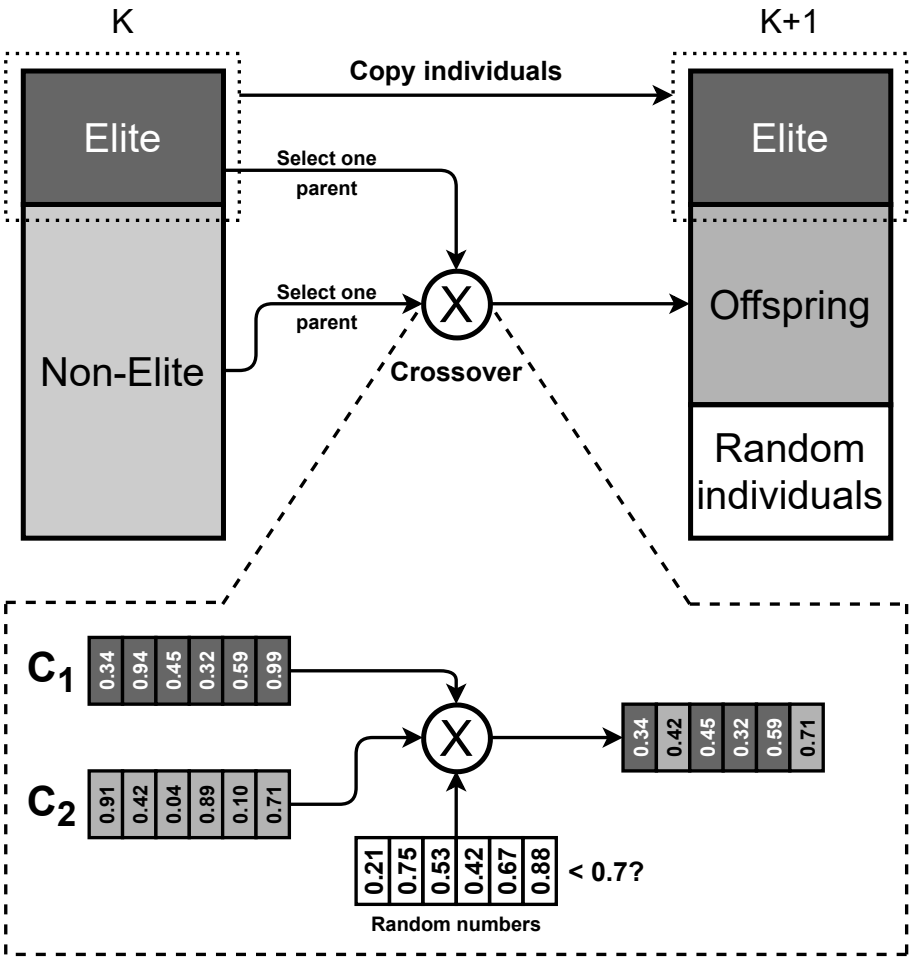


Figure 1.6: BRKGA

1.3.4 Matheuristics

Matheuristics are optimization methods that integrate mathematical programming techniques into a heuristic framework (Boschetti et al., 2009). A matheuristic combines a metaheuristic with exact methods, such as algorithms from mathematical programming and constraint programming. Both components of a hybrid metaheuristic may run concurrently and exchange information to guide the search. Coupling metaheuristics and exact methods is a promising alternative in solving many combinatorial optimization problems. The interest in hybrid approaches has rapidly grown especially due to several inspiring results obtained by the union of these two methods.

1.4 Objectives

Our work is centralized in solving the MBT and the WMBT using a Biased Random-Key Genetic Algorithm (BRKGA). We focus on large and sparse networks since they are the hardest instances to find the optimal solution (Hasson and Sipper, 2004). Finally, our proposal can be summarized as follows (see Figure 1.7):

- For MBT:
  - (i) An algorithm to calculate a lower bound;
  - (ii) A BRKGA metaheuristics;
  - (iii) A BRKGA metaheuristics coupled with a refinement approach;
  - (iv) A hybrid algorithm (BRKGA + ILP);
  - (v) A method to create instances with known optima;
- For WMBT:
  - (i) Three exact models (one for each variant of the WMBT);
  - (ii) An algorithm to calculate a lower bound;
  - (iii) A polynomial algorithm to calculate the WMBT of trees;
  - (iv) Two BRKGA metaheuristics coupled with a refinement approach;
  - (v) A method to create instances with known optima.

1.5 Structure of the Thesis

This work is structured as follows: Chapter 2 is the outcome of an article in revision in the International Transactions in Operational Research (ITOR). It presents the definition, related work of the MBT, and our proposals for the MBT. Chapter 3 introduces the definition, related work of the WMBT, and our contributions to the WMBT. Finally, Chapter 4 includes our final considerations.

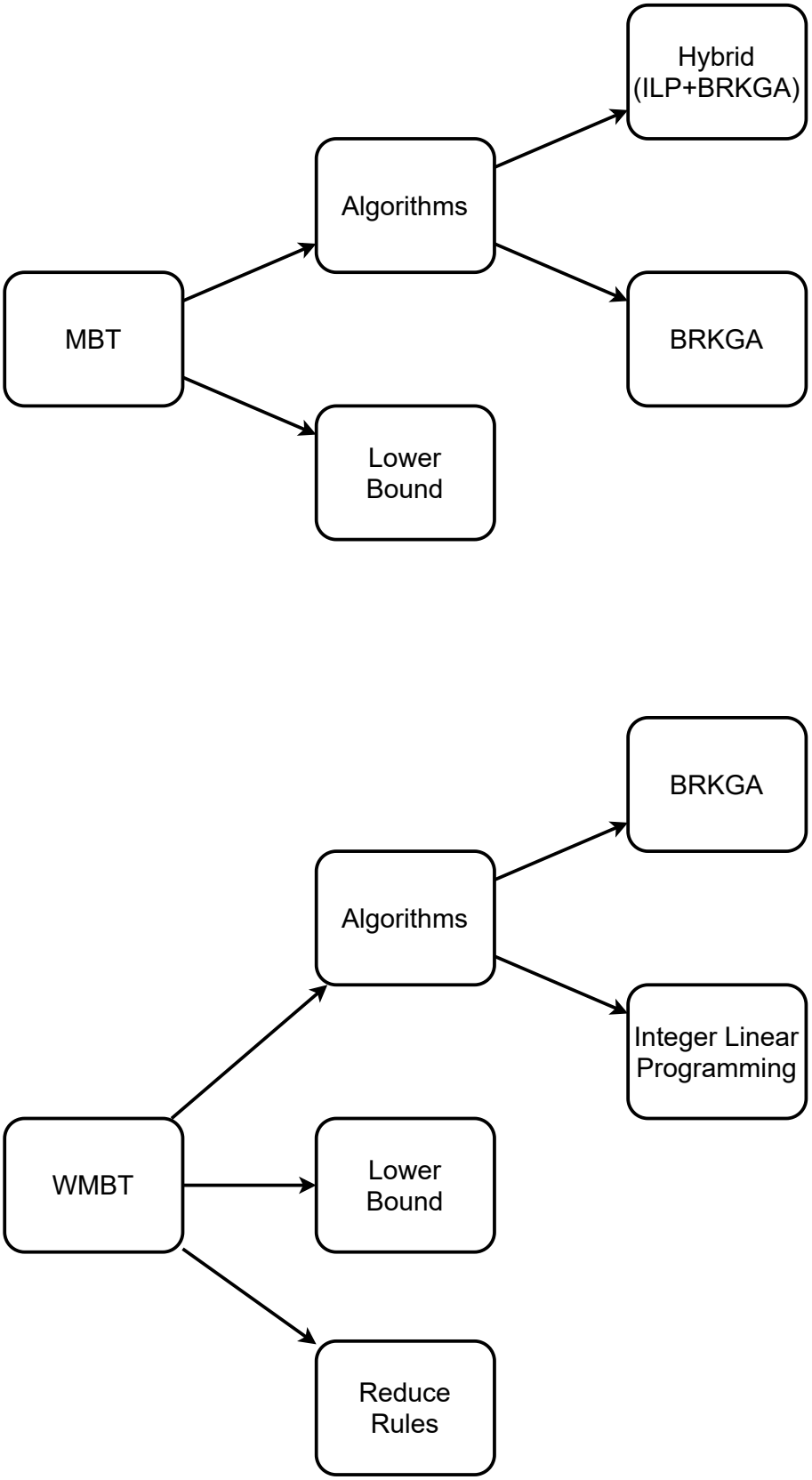


Figure 1.7: Overview of this work.

2

Minimum Broadcast Time

The MINIMUM BROADCAST TIME is a classical $\mathcal{NP}$-hard problem (Garey and Johnson, 1979, problem ND49). The problem can be formally defined as the following. Let $G = (V, E)$ be an undirected connected graph and $V_0 \subseteq V$ a subset of vertices (message originators) which, initially, contain a given message. Let V_t be the set of vertices that receive the message at time t or earlier, with $1 \leq t \leq T$, where T is an upper bound for the broadcast time. From these definitions, the MBT asks to find a sequence $V_0, E_1, V_1, E_2, \ldots, E_k, V_k$ that minimizes k, such that $V_k = V$. In addition, for each $t \in \{1, \ldots, k\}$, the following constraints hold: (i) each edge in E_t has exactly one endpoint in V_{t-1}, (ii) no two edges in E_t share a common endpoint, and (iii) $V_t = V_{t-1} \cup \{v : (u, v) \in E_t\}$.

Here, we have some terminology that will be used throughout this work.

Definition 1. *The broadcast scheme, denoted as B_S, is a sequence of vertices and edges V_0, $E_1, V_1, \ldots, V_T$ that describes a solution.*

Definition 2. *Broadcast tree B_T is a directed spanning tree that describes a solution.*

Definition 3. *Broadcast time of vertex v_0 in G, denoted as $b(G, v_0)$, is minimum number of steps to v_0 broadcast the message for all vertices in V.*

Definition 4. *Broadcast time of graph G is defined by $b(G) = \min_{v_0 \in V}(b(G, v_0))$.*

Definition 5. *Broadcast center of graph G is defined as $BC(G) = \{v_0 \in V |\ b(v_0) = b(G)\}$.*

Definition 6. *A graph G is called broadcast graph if $\max_{v_0 \in V}(b(G, v_0)) = \lceil \log_2 |V| \rceil$.*

Definition 7. *A minimum broadcast graph (MBG) is a broadcast graph with a minimum number of edges.*

2.1 Related Work

This section introduces the related work on the MBT. Given its potential for modeling many real-world applications, the MBT has attracted considerable research interest, and several exact (de Sousa et al., 2018; Ivanova, 2019), approximation (Elkin and Kortsarz, 2003; Kortsarz and Peleg, 1992), and heuristics (Scheuermann and Wu, 1984; Hoelting et al., 1996; de Sousa et al., 2018; Hasson and Sipper, 2004) algorithms have been proposed.

In some special situations, for instance, when $|V_0| = 1$ and the graph is a tree (Koh and Tcha, 1991; Su et al., 2010) or a complete grid (Wojciechowska and Scoy, 1999), the MBT can be optimally solved in polynomial time. For arbitrary graphs, the exact approaches include a dynamic programming algorithm (Scheuermann and Wu, 1984) and the ILP models presented by de Sousa et al. (2018) and Ivanova (2019). Currently, the ILP model suggested by de Sousa et al. (2018) is the best exact method. However, this model is able to solve in reasonable times only instances of up to approximately 50 vertices, which is an insufficient number for real-world industrial applications (this model is presented in Section 2.2.4).

Other works proposed approximation algorithms, constructive-based heuristics and metaheuristics. Approximation algorithms for MBT are studied in Kortsarz and Peleg (1992), where the authors introduce an $O(\sqrt{n})$-additive approximation algorithm for broadcasting in general graphs with n vertices. This work also proposes several approximation algorithms for graph classes with small separators, with an approximation ratio proportional to the separator size times $\log n$. Another algorithm with $O\left(\frac{\log n}{\log \log n}\right)$-approximation ratio is presented in Elkin and Kortsarz (2003).

Scheuermann and Wu (1984) were the first to propose heuristics for the problem, with two constructive-based heuristic algorithms: Least Weight Maximum Matchings (LWMM) and Approximate Matching (AM). The AM presented better results. Next, Hoelting et al. (1996) proposed a genetic algorithm (GA), the first metaheuristic for the MBT. They compared their GA with AM, and the experimental results indicated that the GA produces better broadcast time values. Hasson and Sipper (2004) proposed an ant colony system (ACS) algorithm and compared it with LWMM, AM and GA. The ACS presented the best results in comparison with the other heuristics.

We present a summarized the literature's approaches. Table 2.1 lists the main articles about MBT in the literature.

Table 2.1: List of articles about MBT in the literature.

Article	Approach	Observation
Scheuermann and Wu (1984)	Exact and heuristic algorithms	-
de Sousa et al. (2018)	ILP model and heuristic algorithm	-
Ivanova (2019)	ILP model and lower bound	-
Hoelting et al. (1996)	Metaheuristic algorithm	-
Hasson and Sipper (2004)	Metaheuristic algorithm	-

Continued on next page

Table 2.1 – continued from previous page

Article	Approach	Observation
Slater et al. (1981)	Polynomial time algorithm	For broadcast center problem, when graph is a tree and $\|V_0\| = 1$.
Koh and Tcha (1991)	Polynomial time algorithm	For MBT, when graph is a tree and $\|V_0\| = 1$. With complexity $O(\|E\|\log(\|E\|))$.
Su et al. (2010)	Polynomial time algorithm	For MBT, when graph is a tree and $\|V_0\| = 1$. With complexity $O(\|E\|)$.
Wojciechowska and Scoy (1999)	Polynomial time algorithm	For MBT, when graph is a complete grid and $\|V_0\| = 1$.

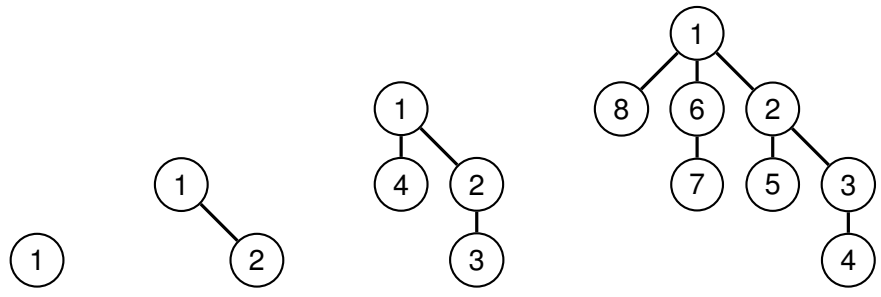
2.2 Algorithmic approaches for the MBT

This section introduces our proposals for the classical MBT. Section 2.2.1 proposes synthetic instances for the MBT. Section 2.2.2 describes a lower bound algorithm for the MBT. Section 2.2.3 presentes some decoders for the MBT. Finally, Section 2.2.4 shows a matheuristic for the MBT.

2.2.1 Synthetic instances for the MBT

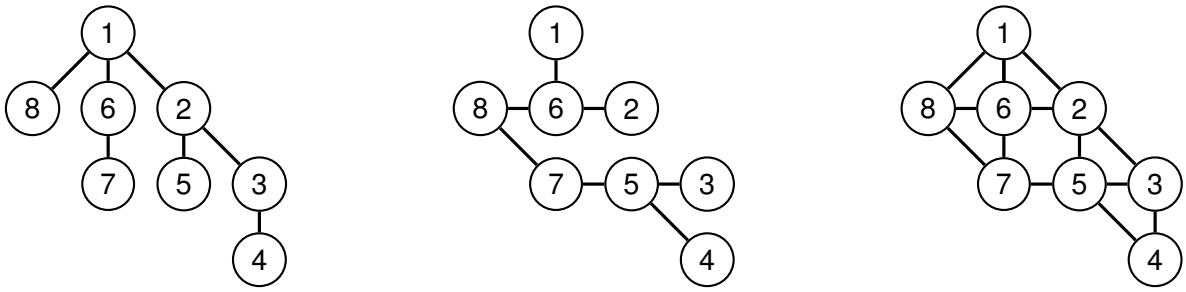
Given that the optimal solution for large MBT instances is often unknown, we generated a new benchmark with known optimal solutions using the following procedure.

Let the instance $G = (V, E)$ be defined by the union of a binomial tree $B_k = (V_B, E_B)$ and a random graph $G_r = (V_r, E_r)$, where $V = V_B = V_r$ and $E = E_B \cup E_r$. The binomial tree B_k is an ordered tree defined recursively as follows: (i) B_0 is a trivial graph, (ii) B_k is constructed from two binomial trees B_{k-1} by attaching one of them as the rightmost (can be leftmost) child of the root of the other. Figure 2.1 shows the binomial trees B_0 through B_3. Note that the MBT of a binomial tree B_k is equal to k if the root of B_k is in V_0. The random graph G_r is based on the $\mathbb{G}(n, p)$ model, also known as the binomial model (Gilbert, 1959). Each graph $G_r = (V, E_r)$ is generated with n vertices and each potential edge in E_r is created with probability p. In Figure 2.2, we apply a union of graphs B_3 and G_r and we obtain a synthetic graph G with 8 vertices.



(a) B_0 (b) B_1 (c) B_2 (d) B_3

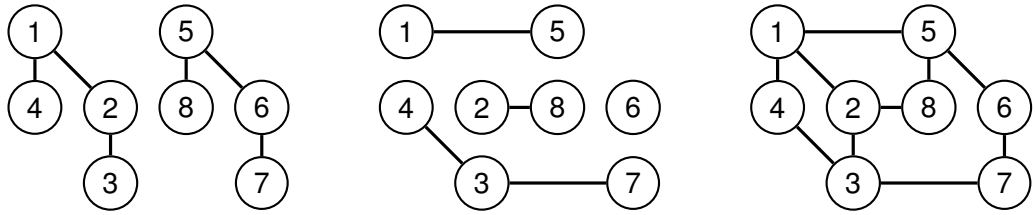
Figure 2.1: Examples of binomial trees.



(a) B_3 (b) G_r (c) $G = B_3 \cup G_r$

Figure 2.2: Building a synthetic instance.

This methodology can be applied to create synthetic instances with multiple sources ($|V_0| > 1$). The idea is simple, we will build $|V_0|$ binomial trees and connect them with a random graph. In Figure 2.3 we created a synthetic instance with $V_0 = \{1, 5\}$.



(a) $B_2 + B_2$ (b) G_r (c) G

Figure 2.3: Building a synthetic instance with 2 sources.

2.2.2 Lower bound algorithm

In many practical applications, bounds on the optimal solutions are sufficient. Bounds can also help solvers to prove the optimally of a solution. Clearly, $|V| - |V_0|$ is an upper bound for the optimal broadcast time. Furthermore, the value

$$TLB(G, V_0) = \left\lceil \log_2 \frac{|V|}{|V_0|} \right\rceil \tag{2.1}$$

defines a theoretical lower bound for the MBT (Ivanova, 2019). It is easy to see that a complete graph needs exactly $\left\lceil \log_2 \frac{|V|}{|V_0|} \right\rceil$ steps to broadcast.

A good estimate of the planning horizon length is very important for the performance of a mathematical formulation. A tighter lower bound can reduce the number of columns in the coefficient matrix and, thus, reduce the computational demand and the total amount of used memory. Moreover, a lower bound can be used by a heuristic algorithm to prove the optimality of a feasible solution, i.e., if a heuristic finds a solution whose value meets the lower bound, then this solution is proved optimal.

Algorithm 1 presents the pseudo-code for the proposed lower bound calculation. The proposed lower bound, called `LBB-BFS`, is based on a multisource breadth-first search (BFS), starting at every vertex of V_0. Its main objective is to find the maximum shortest path between a target vertex and its nearest source. Note a straightforward implementation of `LBB-BFS`, i.e., applying a breadth-first search on each $v \in V_0$, would require $O(|V_0| \cdot |E|)$. This issue can be addressed by labeling every vertex in V_0 as discovered, and placing them at the beginning of the known vertices queue Q_0 (see lines 4–7). Overall, the worst-case running time of the `LBB-BFS` function is $O(|E|)$. A similar approach based on BFS has also been used by de Sousa et al. (2018) to reduce the number of variables in their proposed ILP formulation.

Algorithm 1: Lower bound algorithm for the MBT.

Input : Undirected graph: $G = (V, E)$,
Source set: V_0
Output: Lower bound to broadcast: $lowerBound$

```
1  LBB-BFS(G, V_0)
2      for each v ∈ V do
3          dist[v] ← ∞                                  // Set distance to infinity
4      Q ← InitQueue()                                  // Let Q be an empty queue
5      for each v_0 ∈ V_0 do
6          dist[v_0] ← 0                                // Set distance to zero
7          Q ← Enqueue(Q, v_0)                          // Enqueue v_0 in Q
8      while not isEmpty(Q) do
9          v ← Dequeue(Q)                               // Dequeue v of Q
10         for each u ∈ G.adj[v] do
11             if dist[u] > dist[v] + 1 then            // Check the distance
12                 dist[u] ← dist[v] + 1                // Update distance of u
13                 Q ← Enqueue(Q, u)                    // Enqueue v_0 in Q
14     lowerBound ← max_{v∈V} (dist[v])                 // Get higher distance
15     return lowerBound
```

Theorem 8. *Algorithm 1 returns a lower bound for the optimum of the MBT with input graph $G = (V, E)$ and source set V_0.*

Proof. Let b^* be the optimum of MBT for an instance (G, V_0) and z be the value returned by the `LBB-BFS` algorithm. Let $v \in V_0$ be the closest vertex of V_0 to a vertex $\ell \in V$, such that the distance between them is exactly z. We need at least z steps to reach ℓ from any vertex in V_0. Therefore, $z \leq b^*$, i.e., z is a lower bound for the optimum MBT value. □

The lower bound is given by

$$LB(G, V_0) = \texttt{max}(TLB(G, V_0), LBB\text{-}BFS(G, V_0)), \tag{2.2}$$

where TLB(G,V_0) is defined in Eq. (2.1) and LBB-BFS(G,V_0) is calculated by Algorithm 1. Hence, if the best fitness in the current population is equal to LB(G,V_0), the algorithm proves that it is an optimal solution.

2.2.3 BRKGA decoders for the MBT

The only problem-specific part of BRKGA is the decoder, i.e., the procedure that converts random-keys (chromosomes vector) into a solution for the problem. The following subsections present the two proposed decoders for the MBT.

2.2.3.1 First receive first send decoder

Algorithm 2 describes our first decoder, called FRFS, which is based on the priority decoder of Hoelting et al. (1996). The algorithm receives as input a graph $G(V,E)$, source set V_0, and chromosomes vector Cr. It returns as output the MBT for the encoded candidate solution. In this decoder, the chromosomes vector Cr has size $|V|$. Each allele $Cr[v]$ represents the priority of vertex v to receive a message. The lower the allele value, the higher the priority.

Algorithm 2: FRFS decoder.

Input : Undirected graph: $G=(V,E)$, Source set: V_0, Chromosomes vector: Cr
Output: Total step time to broadcast: $time$

```
1  FRFS(G, V_0, Cr)
2      time ← 0                                                  // Init time
3      Transmitters ← V_0                          // Set the initial Transmitters
4      Ranking ← InitList()                                  // Init Ranking list
5      for each v ∈ Sort(V_0, Cr) do     // Ordering in ascending order of allele value
6          Ranking ← AppendItem(Ranking, v)
7      while Transmitters ≠ V do
8          NewTransmitters ← InitList()              // Set the new transmitters
9          for each v ∈ Ranking do
10             u ← argmin_{u∈N(v)\{Transmitters∪NewTransmitters}} (Cr[u])   // Get the best vertex
11             if u ≠ ∅ then
12                 NewTransmitters ← AppendItem(NewTransmitters, u)   // Add the best
                       vertex in NewTransmitters
13         for each v ∈ NewTransmitters do
14             Transmitters ← Transmitters ∪ {v}                  // Update Transmitters
15             Ranking ← AppendItem(Ranking, v)                     // Update Ranking
16         time ← time + 1                                        // Increment time
17     return time
```

The algorithm starts by setting the variables $time = 0$ and $Transmitters = V_0$ (lines 2–4), where $time$ represents the current time step, set $Transmitters$ the vertices that have the mes-

sage, and list $Ranking$ the order of each transmitter. Next, it appends at list $Ranking$ the vertex v for each vertex $v \in V_0$, in ascending order of allele value (lines 4–6). In the main loop (lines 7–16), the variable $NewTransmitters$ indicates the list of vertices that receive the message in step $time$. This loop is executed until all vertices are in set $Transmitters$. The main loop starts by scanning, in order, each vertex v from list $Ranking$ and selects the neighbor u of v that has not yet received the message, ties are broken according to the allele value $Cr[u]$. The selected vertex u is then added to the list $NewTransmitters$ (line 12). In lines 13–15, the vertices in $NewTransmitters$ are added in $Transmitters$ and appended at the end of the list $Ranking$. Finally, variable $time$ is incremented (line 16). Overall, the worst-case running time of this decoder is $O(|V| \cdot |E|)$.

The graph in Figure 2.4 (with only one source) and chromosomes vector in Table 2.2 show an input example for the decoding procedure. We now describe the `FRFS` decoding procedure for this example. Initially, vertex v_1 is designated as the unique transmitter (Figure 2.5-a). Among the neighbors of v_1, vertex v_3 has the highest priority value (according to its chromosome value), hence vertex v_1 sends the message to v_3 (Figure 2.5-b). In time 2, only vertices v_1 and v_3 have received the message. Because of the ordering in the $Ranking$ list of the `FRFS` algorithm, v_1 will transmit first to vertex v_4 (its second highest priority neighbor), and v_3 transmits to v_8 (Figure 2.5-c). Next, in time 3, v_1 transmits to v_2, and v_4 transmits to v_7 (Figure 2.5-d). Finally, in time 4, the other vertices receive the messages in the following order: $v_4 \rightarrow v_6$ and $v_7 \rightarrow v_5$ (Figure 2.5-e).

Figure 2.4: Input graph

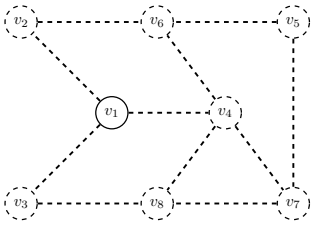


Table 2.2: Chromosomes of input

Vertices	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
Value(*Cr*)	0.5	0.4	0.1	0.2	0.8	0.7	0.6	0.3

2.2.3.2 First Receive First Send with SCHA improvement

Preliminary experiments with the `FRFS` decoder have shown that this decoder has a poor performance in sparse graphs. This motivated us to devise a new decoder, called `FRFS-SCHA`. This new decoder combines `FRFS` with a greedy method, called `SCHA`, that is able to find in polynomial time the optimal MBT on forest graphs. More specifically, in `FRFS-SCHA`, we use an adaptation of the `FRFS` decoder to produce a forest that is then used as input to the `SCHA` algorithm. In the following, we give more details about `FRFS-SCHA`.

Algorithm 3 introduces `SCHA`. This algorithm receives as input a forest graph $F(V,E)$ and source set V_0. It returns as output the optimal MBT for the forest. `SCHA` iteratively calls function `MBT-Tree` (line 5), which is an algorithm proposed by Su et al. (2010); Koh and Tcha (1991) for

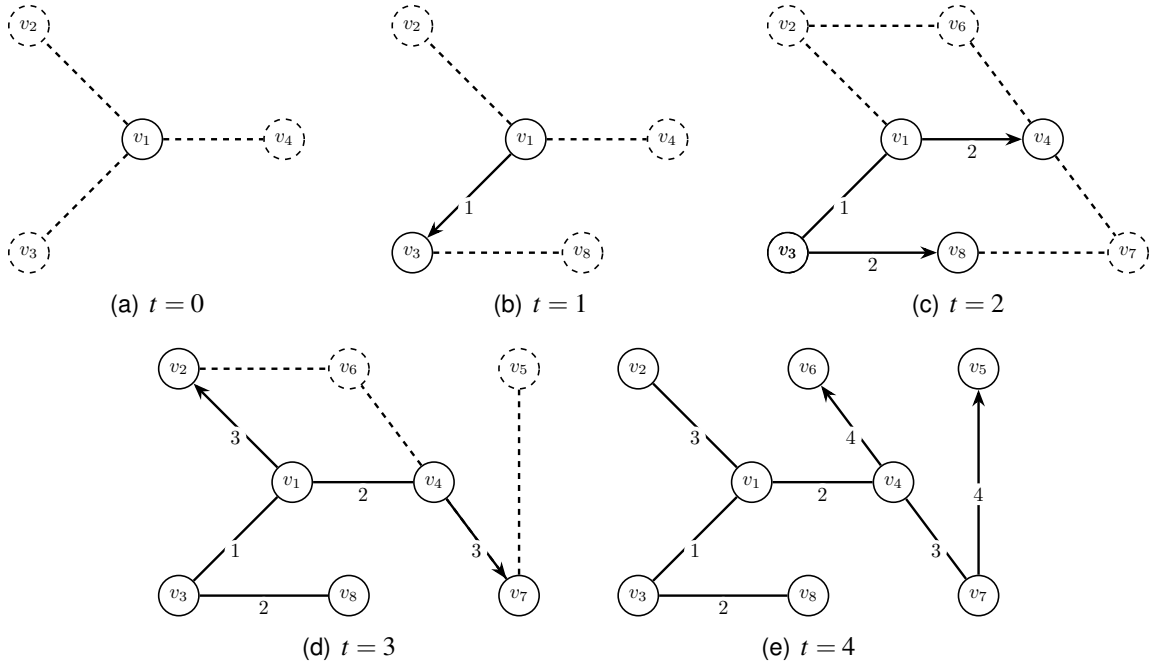


Figure 2.5: FRFS decoding procedure.

finding the MBT on tree graphs. Hence, SCHA finds the optimal MBT by calculating the MBT of each tree in the forest, and then returning the greatest MBT among these trees. Given that the asymptotic complexity of MBT-Tree is $O(|V| \cdot \log_2(|V|))$, SCHA is $O(|V| \cdot \log_2(|V|))$.

Algorithm 3: SCHA.

Input : Forest undirected: $F = (V, E)$, Source set: V_0
Output: Total step time to broadcast: $time$

```
1 SCHA(F, V_0)
2     time ← 0                                              // Init time
3     for each v_0 ∈ V_0 do
4         T ← GetTree(F, v_0)                               // Get the tree with root v_0
5         time ← max(time, MBT-Tree(T, v_0))                // MBT of tree T from source v_0
6     return time
```

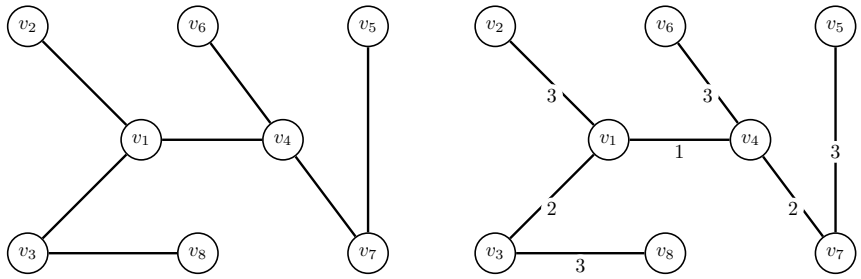
Our new decoder FRFS-SCHA is depicted in Algorithm 4. In lines 2–16, FRFS was adapted to produce a forest. In line 17, this forest is used as input to SCHA, which calculates the resulting MBT. Because of the complexity of FRFS is $O(|V| \cdot |E|)$ and SCHA is $O(|V| \cdot \log_2 |V|)$, the worst-case running time of this decoder is $O(|V| \cdot |E|)$.

Figure 2.6-b shows the resulting forest for the input example in Figure 2.4 and Table 2.2. The solution obtained after the SCHA procedure is applied to this forest is depicted in Fig 2.6-a. Note this solution is better than the solution previously obtained with the original FRFS (Figure 2.5-e).

Algorithm 4: FRFS-SCHA decoder.

Input : Undirected graph: $G = (V, E)$, Source set: V_0, Chromosomes vector: Cr
Output: Total step time to broadcast: $time$

```
1  FRFS-SCHA(G, V_0, Cr)
2      E_F ← ∅                                   // Init the set of edges from the forrest graph
3      Transmitters ← V_0                        // Set the initial Transmitters
4      Ranking ← InitList()                      // Init Ranking list
5      for each v ∈ Sort(V_0, Cr) do             // Ordering in ascending order of allele value
6          Ranking ← AppendItem(Ranking, v)
7      while Transmitters ≠ V do
8          NewTransmitters ← InitList()          // Set the new transmitters
9          for each v ∈ Ranking do
10             u ← argmin_{u∈N(v)\{Transmitters∪NewTransmitters}} (Cr[u])   // Get the best vertex
11             if u ≠ ∅ then
12                 NewTransmitters ← AppendItem(NewTransmitters, u)   // Add the best
                       vertex in NewTransmitters
13                 E_F ← E_F ∪ {(v, u)}          // Add the edge in E_F
14         for each v ∈ NewTransmitters do
15             Transmitters ← Transmitters ∪ {v}  // Update Transmitters
16             Ranking ← AppendItem(Ranking, v)   // Update Ranking
17     return SCHA((V, E_F), V_0)                // Compute MBT of forest with SCHA
```



(a) $t = 0$ (b) $t = 1$

Figure 2.6: FRFS-SCHA decoding procedure.

2.2.4 Proposed matheuristic

This section describes the proposed matheuristic, which combines an extension of the ILP model of de Sousa et al. (2018) and our BRKGA. To the best of our knowledge, this is the first work to propose a matheuristic for the MBT.

This is a small extension for the ILP model of de Sousa et al. (2018). Different from the original model, ILP extension considers graphs with multi-source (i.e., $|V_0| > 1$). Let $V = \{0, 1, \cdots, n-1\}$ be the set of vertices, V_0 the set of sources, E the set of connections between two vertices, and $N(i)$ the set of neighboring vertices of vertex i. In the model, K_i is a binary constant indicating whether or not vertex i is in V_0, and T_{max} a constant that represents

an upper limit on the time in which a vertex receives the message (e.g., $T_{max} = |V| - |V_0|$ is the trivial limit). Finally, define T as a decision variable that represents the minimum broadcast time, and x_{ij}^t as a binary variable that has value 1 if the vertex i sends the message to the vertex j in time t and 0, otherwise. Based on these elements, the ILP model is as follows:

$$\min \quad T \tag{2.3}$$

$$\text{s. t} \quad K_i + \sum_{j \in N(i)} \sum_{t=1}^{T_{max}} x_{ji}^t = 1 \qquad \forall i \in V \tag{2.4}$$

$$\sum_{j \in N(i)} x_{ij}^t \leq 1 \qquad \forall i \in V, \forall t \in [1, T_{max}] \tag{2.5}$$

$$x_{ij}^t \leq K_i + \sum_{\tau=1}^{t-1} \sum_{k \in N(i) \setminus \{j\}} x_{ki}^\tau \qquad \forall (i,j) \in E, \forall t \in [1, T_{max}] \tag{2.6}$$

$$\sum_{t=1}^{T_{max}} t \cdot x_{ij}^t \leq T \qquad \forall (i,j) \in E \tag{2.7}$$

$$T \in \mathbb{N} \tag{2.8}$$

$$x_{ij}^t \in \mathbb{B} \qquad \forall (i,j) \in E, \forall t \in [1, T_{max}] \tag{2.9}$$

Equation (2.3) is the objective function. Constraints (2.4) limit each vertex to receive only one message at a time or to start with it. Constraints (2.5) require that each vertex sends at most one message to a neighbor in each t. Constraints (2.6) establish that each vertex can only transmit if it has already received the message. Constraints (2.7) state that the value of T must be greater than or equal to the time of any transmission. Finally, Constraints (2.8) and (2.9) define the domain of the decision variables.

Figure 2.7 illustrates the main ideas of the proposed matheuristic. First, given a problem instance, the matheuristic uses `BRKGA` to generate and maintain a pool of candidate solutions. After some `BRKGA` iterations without improvement in this pool, the matheuristic combines the solutions in the pool through a merging process. The merging process generates a subgraph that contains the original instance vertices and some edges from the solutions in the pool. Next, the matheuristic uses the ILP model to solve the problem instance induced by this subgraph. Note that solving the induced problem instance is far easier than solving the original instance. The solution obtained through the ILP model is then added to the pool of solutions, and some other solution is removed from the pool. The process in Figure 2.7 is then repeated until a stop criterion is met.

Algorithm 5 details the proposed matheuristic. It starts with an initial pool of candidate solutions P generated through `BRKGA` (line 3). In the main loop (lines 5–28), the algorithm uses `BRKGA` to evolve the current pool of solutions, and then it checks if this pool has improved. If

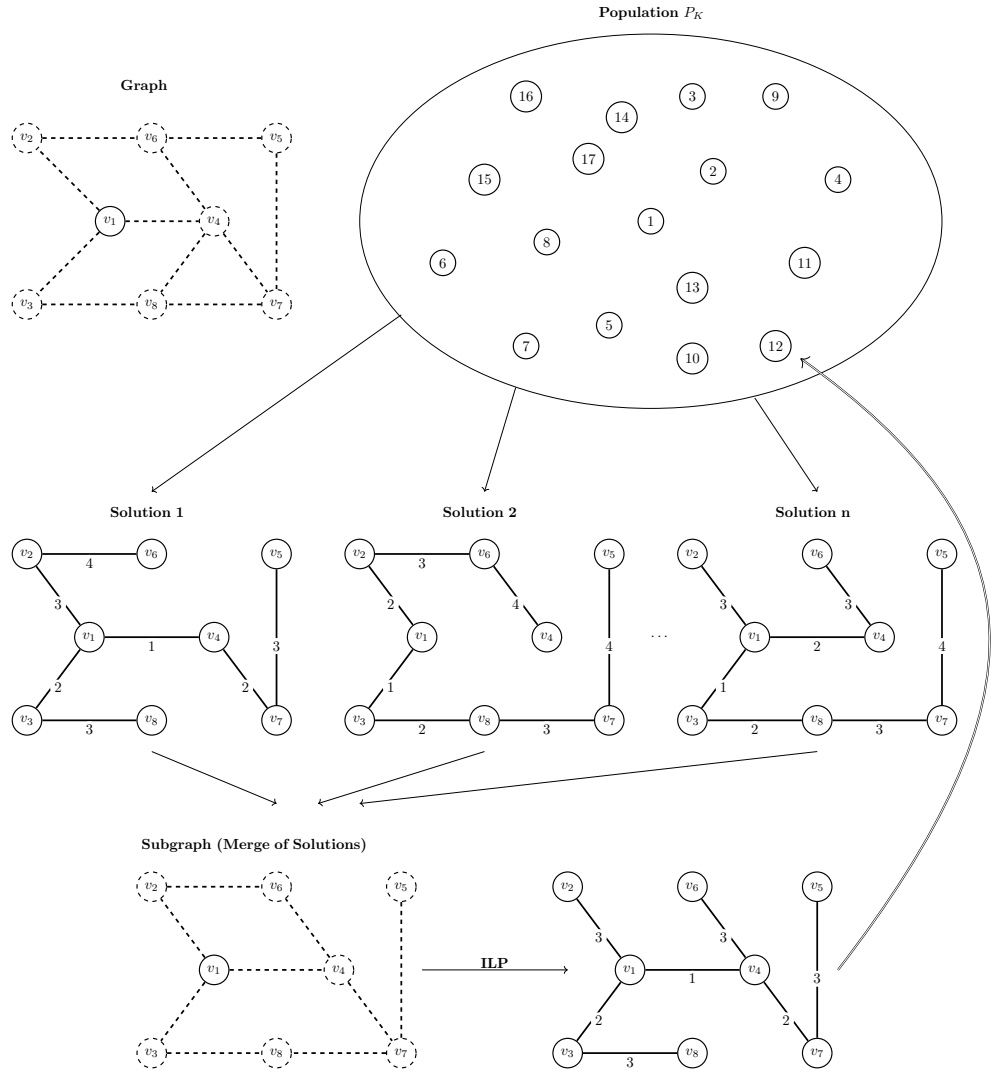


Figure 2.7: Proposed matheristic: pool of solutions merging and solving process.

no improvement is achieved after $maxIt$ iterations, the merging process begins. In the merging process, the subgraph $G_S(V, E_S)$ is created by adding the edges from the best solutions in the pool until the edge density of the subgraph is greater than or equal to d (lines 15–22). After the subgraph is created, the algorithm uses the lower bound given in Eq. (2.2) to check if a better solution than the best current one S^* can be found. Note this checking is useful to avoid unnecessary ILP model solving, which can be quite costly. If the subgraph passes this filter, the algorithm solves the ILP model using a standard model solver. The model solver is invoked with S^* as the initial solution and a time limit of t_{ILP}. If the ILP solution returned by the model solver S_{ILP} is better than S^*, then we remove S^* from P, add S_{ILP} in P, and update S^* with S_{ILP} (lines 25–28).

Algorithm 5: Proposed matheuristic for the MBT.

Input : Undirected graph: $G = (V,E)$, Source set: V_0, Time limit for each running of ILP: t_{ILP},
Minimal density of subgraph: d,
Maximum number of generations without improvement: $maxIt$

Output: Schedule Broadcast: S

```
1  BRKGA+ILP(G, V_0, t_ILP, d, maxIt)
2      P ← BRKGA_Init(G, V_0)                    // Inicialize the first BRKGA population
3      S* ← GetBestIndividual(P)                 // Get the best individual of the population
4      noImprovement ← 0
5      while stopping criterion not met do
6          P ← BRKGA_Evolve(P)                   // Evolve population
7          S ← GetBestIndividual(P)              // Get the best individual of the
           population
8          if Fitness(S) < Fitness(S*) then      // Check if the population has improved
9              S* ← S                            // Update S*
10             noImprovement ← 0
11         else
12             noImprovement ← noImprovement + 1
13         if noImprovement = maxIt then         // If no improvement after I generations
14             noImprovement ← 0
15             E_S ← GetEdges(S*)
16             E_P ← InitList()                  // Let E_P a empty list
17             for each S in Sort(P \ S*) do     // For each solution S in P (in fitness
               order)
18                 E_P ←AppendItems(GetEdges(S))  // Append the edges of S in E_P
19             for each e in E_P do              // For each edge e in E_P
20                 if Density((V, E_S)) ≥ d then
21                     break
22                 E_S ← E_S ∪ {e}               // Add the edge e in E_S
23             if LB((V, E_S), V_0) < Fitness(S*) then  // Check if a better solution can
               be found
24                 S_ILP ← ILP-MBT((V, E_S), V_0, S*, t_ILP)   // Run ILP
25                 if Fitness(S_ILP) < Fitness(S*) then  // Check if S_ILP is better
26                     P ← P \ {S*}              // Remove solution S* on P
27                     P ← P ∪ {S_ILP}           // Add solution S_ILP in P
28                     S* ← S_ILP                // Update S*
29     return S*
```

2.3 Computacional results of the MBT

This section presents the computational experiments conducted to evaluate the effectiveness of our proposed BRKGA and matheuristic. The proposed algorithms are compared with the following state-of-the-art approaches: (i) the Ant Colony metaheuristic (ACS) from Hasson and Sipper (2004); (ii) the ILP model from de Sousa et al. (2018); and (iii) the constructive heuristic TreeBlock from de Sousa et al. (2018).

Since we could not obtain the source code of ACS, we implemented our own version of this algorithm based on its original definitions. We did not implement the GA approach of (Hoelting et al., 1996) because the results reported in Hasson and Sipper (2004) are enough to conclude that ACS outperforms it.

To test the effectiveness of the SCHA decoding approach, we developed two versions of our BRKGA: one without SCHA (Algorithm 2), called BRKGA_FRFS, and one with SCHA (Algorithm 4), called BRKGA. The effectiveness of SCHA was also tested on the proposed matheuristic, hence two versions of the matheuristic were devised: one without SCHA, called BRKGA_FRFS+ILP, and one with SCHA, called BRKGA+ILP.

All experiments in this section were conducted on an Intel Core i7-6700 with 3.40 GHz, 32 GB of RAM, running Ubuntu 18.04.5. The heuristic algorithms were coded in C++ and compiled with g++ 7.5 and '-O3' flag. The BRKGA C++ framework developed by Toso and Resende (2015) has been used to implement our BRKGA. Moreover, IBM Cplex 12.9 has been adopted to solve the ILP models.

2.3.1 Instances

The algorithms were tested on a total of 142 instances, which include:

- **Harary graphs** (16 instances): These are the same instances used in the work of de Sousa et al. (2018). A Harary graph, denoted by $H_{k,n}$, is a k-connected graph with n vertices having the smallest possible number of edges.

- **Others graphs** (25 instances): We also included the instances we were able to reproduce from the works of Harutyunyan and Wang (2010); Harutyunyan and Jimborean (2014). These instances are hypercube (6 instances), shuffle exchange (7 instances), cube-connected cycles graphs (5 instances), and deBruijn (7 instances).

- **Network Data Repository**[1] (59 instances): Because the MBT instances used in previous works (Scheuermann and Wu, 1984; Hoelting et al., 1996; Hasson and Sipper, 2004) are no longer available, we have chosen some instances from well-established benchmarks of complex network problems. To cover different industry scenarios, we consider instances based on connected small-world networks with 100 or 1000 vertices (Freitas et al., 2019).

[1] http://www.networkrepository.com

We have chosen the small-world model because it is commonly used to represent communication networks in industrial scenarios, as suggested in (Guidoni et al., 2010; Cabral et al., 2013).

- **Large synthetic instances based on Binomial Tree** (42 instances): Given that the optimal solution for large MBT instances is often unknown, we generated a new benchmark with known optimal solutions using the following procedure. Let the instance $G = (V, E)$ be defined by the union of a binomial tree $B_k = (V_B, E_B)$ and a random graph $G_r = (V_r, E_r)$, where $V = V_B = V_r$ and $E = E_B \cup E_r$. The binomial tree B_k is an ordered tree defined recursively as follows: (i) B_0 is a trivial graph, (ii) B_k is constructed from two binomial trees B_{k-1} by attaching one of them as the rightmost (can be leftmost) child of the root of the other. Figure 2.1 shows the binomial trees B_0 through B_3. Note that the MBT of a binomial tree B_k is equal to k if the root of B_k is in V_0. The random graph G_r is based on the $\mathbb{G}(n, p)$ model, also known as binomial model (Gilbert, 1959). Each graph $G_r = (V, E_r)$ is generated with n vertices and each potential edge in E_r is created with probability p. In Figure 2.2-b, we apply an union of graphs B_3 and G_r, obtaining a synthetic instance G with 8 vertices.

For much more detail of instances, see Appendix A.1.

2.3.2 Parameter settings and experimental protocol

In our experiments with the ACS algorithm, we adopted the same parameter settings indicated by its authors. Moreover, we have used the irace tunning tool (López-Ibáñez et al., 2016) to configure the parameters of our algorithms and their variations. The best parameter settings identified by the tuning experiment are reported in Table 2.3. In this table, BRKGA parameters p, p_e, p_m, ρ_e, and K represent, respectively, number of individuals in each population, percentage of elite individuals into each population, percentage of mutants introduced at each generation into the population, probability that an offspring inherits the allele of its elite parent, and the number of independent populations. Parameters d and t_{ILP} are used only in the matheuristic and represent, respectively, density of subgraph $G_S(V, E_S)$ and time limit for the ILP model solver. The matheuristic has an additional parameter $maxIt$ (maximum number of generations without improvement), which was experimentally set to a linear function that depends on $|V|$: $maxIt = 550 - \left\lceil \frac{|V|}{10} \right\rceil$, where $20 \leq maxIt \leq 500$.

Table 2.3: Range considered by IRACE and best parameter settings obtained.

Parameter	Value ranges	BRKGA_FRFS	BRKGA	BRKGA_FRFS+ILP	BRKGA+ILP
p	-	$\|V\|$	$\|V\|$	$\|V\|$	$\|V\|$
p_e	0.10, 0.11, ..., 0.25	0.16	0.11	0.16	0.11
p_m	0.10, 0.11, ..., 0.30	0.11	0.19	0.11	0.19

Continued on next page

Table 2.3 – continued from previous page

Parameter	Value ranges	BRKGA_FRFS	BRKGA	BRKGA_FRFS+ILP	BRKGA+ILP
ρ_e	0.50, 0.51, ..., 0.80	0.69	0.53	0.69	0.53
K	-	1	1	1	1
d	0.10, 0.11, ..., 1.00	-	-	0.85	0.30
t_{ILP}	5.00, 5.01, ..., 25.00	-	-	13.31	18.01

We have set a time limit of 3600 s (1 h) to Cplex for solving the ILP models from de Sousa et al. (2018). To assess the average performance of the heuristic algorithms (BRGKAs and ACS), we have performed 10 runs in each benchmark instance with different random seeds for each run. A time limit of 60 s has been used for these runs.

2.3.3 Experiments on Harary Graphs

Table 2.4 compares BRKGA_FRFS with the heuristic approaches ACS and TreeBlock in Harary instances. The methods are compared using the following criteria: the best, the average, the worst solution obtained (columns 'Best', 'Avg.' and 'Worst', respectively), as well as the average CPU time to find the best solution (column 't (s)'). An asterisk means that the method has been able to prove the optimality of a given result. If in a run the heuristic fails to attain the best solution, its CPU time to find the best in this run is considered to be the cutoff time of 60 seconds. The bottom of Table 2.4 shows a summary that includes: number of instances in which the method found the best broadcast time, number of instances in which the algorithm determined the best average broadcast time, and the average of the average CPU time values to find the best solution. The results of TreeBlock are reproduced from its original paper.

Table 2.4: Comparative results of ACS, Treeblock and BRKGA_FRFS.

Instance	ACS				Treeblock	BRKGA_FRFS			
	Best	Avg.	Worst	t (s)	Best	Best	Avg.	Worst	t (s)
$H_{10,30}$	**5***	**5.00**	5	0.10	6	**5***	**5.00**	5	< 0.01
$H_{11,50}$	**6***	**6.00**	6	< 0.01	7	**6***	**6.00**	6	< 0.01
$H_{20,50}$	**6***	**6.00**	6	< 0.01	8	**6***	**6.00**	6	< 0.01
$H_{21,50}$	**6***	**6.00**	6	< 0.01	7	**6***	**6.00**	6	< 0.01
$H_{2,100}$	**50***	**50.00**	50	< 0.01	**50**	**50***	**50.00**	50	< 0.01
$H_{2,17}$	**9***	**9.00**	9	< 0.01	**9**	**9***	**9.00**	9	< 0.01
$H_{2,30}$	**15***	**15.00**	15	< 0.01	**15**	**15***	**15.00**	15	< 0.01
$H_{2,50}$	**25***	**25.00**	25	< 0.01	**25**	**25***	**25.00**	25	< 0.01
$H_{3,17}$	**5***	**5.00**	5	< 0.01	**5**	**5***	**5.00**	5	< 0.01
$H_{3,30}$	**9***	**9.00**	9	< 0.01	**9**	**9***	**9.00**	9	< 0.01
$H_{3,50}$	**14***	**14.00**	14	< 0.01	**14**	**14***	**14.00**	14	< 0.01
$H_{5,17}$	**5***	**5.00**	5	< 0.01	**5**	**5***	**5.00**	5	< 0.01
$H_{6,17}$	**5***	**5.00**	5	< 0.01	**5**	**5***	**5.00**	5	< 0.01
$H_{7,17}$	**5***	**5.00**	5	< 0.01	**5**	**5***	**5.00**	5	< 0.01
$H_{8,30}$	6	6.00	6	60.00	6	**5***	**5.00**	5	0.02
$H_{9,30}$	**5***	**5.00**	5	< 0.01	6	**5***	**5.00**	5	< 0.01
# Best		15			10		**16**		

Continued on next page

Table 2.4 – continued from previous page

Instance	ACS				Treeblock	BRKGA_FRFS			
	Best	Avg.	Worst	t (s)	Best	Best	Avg.	Worst	t (s)
# Best Avg.			15		-			**16**	
Avg. t (s)			3.76		-			**0.01**	

The results in Table 2.4 show that our BRKGA version without SCHA outperforms both ACS and TreeBlock. In particular, BRKGA_FRFS was able to prove the optimal solution for all Harary graph instances in milliseconds.

Next, we compare in Table 2.5 our BRKGA_FRFS with the ILP from de Sousa et al. (2018). Five variants of this model were considered: (i) the original model without bounds, (ii) the model with the lower bound defined in Eq. (2.1), (iii) the model with the lower bound defined in Eq. (2.2), (iv) the model with an upper bound determined by our BRKGA_FRFS, and (v) the model with upper and lower bounds.

It is possible to observe in the results of Table 2.5 that the utilization of bounds in the ILP model effectively reduced the computational time. Without the upper bounds provided by our BRKGA_FRFS, instances $H_{11,50}$, $H_{20,50}$, and $H_{21,50}$ could not be optimally solved by the ILP model. In instances $H_{2,100}$, $H_{2,30}$ and $H_{3,50}$, the ILP model with the proposed lower bound outperformed the model with the theoretical lower bound defined in Eq. (2.1). Indeed, for these three instances, $LBB\text{–}BFS(G,V_0) > TLB(G,V_0)$ (Table A.1). Additionally, BRKGA_FRFS used less CPU time to prove the optimal solution than all ILP model variants.

Table 2.5: Comparative results of BRKGA_FRFS and ILPs.

Instance	de Sousa et al's ILP										BRKGA_FRFS		
	No Bounds		Lower Bound - Eq. (2.1)		Lower Bound - Eq. (2.2)		Upper Bound		Both Bounds				
	Best	t (s)	Best	t (s)	Best	t (s)	Best	t (s)	Best	t (s)	Best	Avg.	t(s)
$H_{10,30}$	**5***	8.91	**5***	44.01	**5***	43.63	**5***	1.71	**5***	0.11	**5***	**5.00**	< 0.01
$H_{11,50}$	**6**	3600	**6***	2431.80	**6***	2472.72	**6**	3600	**6***	0.17	**6***	**6.00**	< 0.01
$H_{20,50}$	**6***	890.20	**6***	0.07	**6***	0.07	**6**	3600	**6***	0.01	**6***	**6.00**	< 0.01
$H_{21,50}$	**6**	3600	**6***	0.07	**6***	0.07	**6**	3600	**6***	0.01	**6***	**6.00**	< 0.01
$H_{2,100}$	**50***	9.71	**50***	7.71	**50***	0.13	**50***	0.76	**50***	0.76	**50***	**50.00**	< 0.01
$H_{2,17}$	**9***	0.02	**9***	0.02	**9***	0.02	**9***	0.01	**9***	0.01	**9***	**9.00**	< 0.01
$H_{2,30}$	**15***	0.08	**15***	0.08	**15***	< 0.01	**15***	0.02	**15***	0.02	**15***	**15.00**	< 0.01
$H_{2,50}$	**25***	1.06	**25***	0.82	**25***	0.01	**25***	0.09	**25***	0.09	**25***	**25.00**	< 0.01
$H_{3,17}$	**5***	0.09	**5***	0.03	**5***	0.03	**5***	0.01	**5***	< 0.01	**5***	**5.00**	< 0.01
$H_{3,30}$	**9***	0.72	**9***	0.61	**9***	0.36	**9***	0.03	**9***	0.03	**9***	**9.00**	< 0.01
$H_{3,50}$	**14***	3.23	**14***	2.33	**14***	0.64	**14***	0.09	**14***	0.09	**14***	**14.00**	< 0.01
$H_{5,17}$	**5***	0.61	**5***	< 0.01	**5***	< 0.01	**5***	0.52	**5***	< 0.01	**5***	**5.00**	< 0.01
$H_{6,17}$	**5***	0.61	**5***	< 0.01	**5***	< 0.01	**5***	0.36	**5***	< 0.01	**5***	**5.00**	< 0.01
$H_{7,17}$	**5***	61.49	**5***	< 0.01	**5***	< 0.01	**5***	134.06	**5***	< 0.01	**5***	**5.00**	< 0.01
$H_{8,30}$	**5***	3.78	**5***	1.42	**5***	1.42	**5***	0.20	**5***	0.10	**5***	**5.00**	0.02
$H_{9,30}$	**5***	14.72	**5***	29.79	**5***	29.73	**5***	8.43	**5***	0.08	**5***	**5.00**	< 0.01
# Best	16		16		16		16		16		16		
Avg. t (s)	512.19		157.42		159.30		684.12		0.09		0.01		

2.3.4 Experiments on others instances from literature

Table 2.6 compares the methods on the instances we were able to reproduce from the works of Harutyunyan and Wang (2010); Harutyunyan and Jimborean (2014). In the experiments shown in this table, we use the BRKGA version with SCHA decoding. The results for NTBA and NEWH were reproduced from their original papers (Harutyunyan and Wang, 2010; Harutyunyan and Jimborean, 2014). A hyphen in Table 2.6 indicates we do not have the method result for an instance or the method could not produce a feasible solution. Once again, the results show that BRKGA is competitive in terms of both solution quality and computational efficiency relative the best performing methods in the literature.

Table 2.6: Comparative results of NEWH, NTBA, ILP, ACS and BRKGA

Instance	Treeblock	NTBA	NEWH	ILP (Both bounds)		ACS			BRKGA		
	Best	Best	Best	Best (GAP)	t (s)	Best	Avg.	t (s)	Best	Avg.	t (s)
HC_5	**5**	**5**	**5**	**5***	< 0.01	**5***	**5.00**	< 0.01	**5***	**5.00**	< 0.01
HC_6	**6**	7	**6**	**6***	< 0.01	**6***	**6.00**	< 0.01	**6***	**6.00**	< 0.01
HC_7	**7**	9	**7**	**7***	0.02	**7***	**7.00**	0.01	**7***	**7.00**	0.16
HC_8	**8**	11	**8**	**8***	0.07	**8***	**8.00**	0.01	9	9.00	60.00
HC_9	**9**	14	**9**	**9***	0.39	**9***	**9.00**	0.02	10	10.00	60.00
HC_{10}	**10**	15	**10**	**10***	2.02	**10***	**10.00**	0.23	11	11.00	60.00
CCC_3	-	**6**	7	**6***	0.04	**6***	**6.00**	0.04	**6***	**6.00**	< 0.01
CCC_4	-	**9**	**9**	**9***	0.14	**9***	**9.00**	0.01	**9***	**9.00**	< 0.01
CCC_5	-	**11**	12	**11***	0.57	12	12.00	60.00	**11***	**11.00**	< 0.01
CCC_6	-	14	14	**13***	4.82	14	14.90	60.00	**13***	13.90	55.83
CCC_7	-	**16**	17	-	3600.00	17	17.80	60.00	**16**	**16.00**	1.37
DB_4	**4**	5	5	-	3600.00	5	5.00	60.00	5	5.00	60.00
DB_5	7	7	7	-	3600.00	**6***	**6.00**	1.27	**6***	**6.00**	< 0.01
DB_6	**8**	**8**	**8**	-	3600.00	**8**	**8.00**	0.03	**8**	**8.00**	< 0.01
DB_7	12	10	10	-	3600.00	10	10.00	60.00	**9**	**9.00**	0.60
DB_8	12	12	12	-	3600.00	12	12.00	60.00	**11**	**11.00**	< 0.01
DB_9	14	**13**	**13**	-	3600.00	14	14.00	60.00	**13**	**13.00**	< 0.01
DB_{10}	15	15	15	-	3600.00	16	16.00	60.00	**14**	**14.20**	21.48
SE_4	-	**7**	**7**	**7***	0.02	**7**	**7.00**	< 0.01	**7**	**7.00**	< 0.01
SE_5	-	**9**	**9**	**9***	0.02	**9***	**9.00**	0.01	**9***	**9.00**	< 0.01
SE_6	-	**11**	**11**	**11***	0.05	**11***	**11.00**	0.01	**11***	**11.00**	< 0.01
SE_7	-	**13**	**13**	**13***	0.25	**13***	**13.00**	0.01	**13***	**13.00**	< 0.01
SE_8	-	**15**	**15**	**15***	1.21	**15***	**15.00**	0.04	**15***	**15.00**	0.34
SE_9	-	18	18	**17***	12.65	**17***	**17.00**	0.04	18	18.00	60.00
SE_{10}	-	20	20	-	3600.00	**19***	**19.00**	0.07	20	20.00	60.00
# Best	8/13	13/25	15/25	16/25		17/25			19/25		
# Best Avg.	8/13	13/25	15/25	16/25		17/25			18/25		
Avg. t (s)	-	-	-	1296.89		19.27			**17.59**		

2.3.5 Experiments on Small-World and synthetic instances

Table 2.7 compares BRKGA_FRFS with ACS and the ILP model with both bounds in the Small-World and synthetic instances. In this table, we have added a GAP information for instances in which the ILP model did not prove the optimal solution.

Table 2.7: Comparative results of ACS, ILP and BRKGA_FRFS

Instance	ACS			ILP (Both bounds)		BRKGA_FRFS		
	Best	Avg.	t (s)	Best (GAP)	t (s)	Best	Avg.	t (s)
B_4	**4***	**4.00**	< 0.01	**4***	< 0.01	**4***	**4.00**	< 0.01
B_5	**5***	**5.00**	< 0.01	**5***	< 0.01	**5***	**5.00**	2.03
B_6	**6***	**6.00**	< 0.01	**6***	< 0.01	**6***	6.80	55.04
B_7	**7***	**7.00**	< 0.01	**7***	0.01	8	8.80	60.00
B_8	**8***	**8.00**	0.01	**8***	0.06	10	10.60	60.00
B_9	**9***	**9.00**	0.08	**9***	0.35	12	12.30	60.00
$B_5 \cup \mathbb{G}(32, 5\%)$	6	6.00	60.00	**5***	0.01	**5***	5.90	58.45
$B_5 \cup \mathbb{G}(32, 7.5\%)$	6	6.00	60.00	**5***	0.06	**5***	5.10	15.21
$B_5 \cup \mathbb{G}(32, 10\%)$	**5***	**5.00**	2.28	**5***	0.02	**5***	**5.00**	0.15
$B_5 \cup \mathbb{G}(32, 15\%)$	**5***	**5.00**	2.25	**5***	0.02	**5***	**5.00**	0.02
$B_5 \cup \mathbb{G}(32, 20\%)$	**5***	**5.00**	0.03	**5***	0.11	**5***	**5.00**	< 0.01
$B_5 \cup \mathbb{G}(32, 25\%)$	**5***	**5.00**	0.05	**5***	0.15	**5***	**5.00**	< 0.01
$B_6 \cup \mathbb{G}(64, 5\%)$	7	7.00	60.00	**6***	1.02	7	7.00	60.00
$B_6 \cup \mathbb{G}(64, 7.5\%)$	7	7.00	60.00	**6***	2.34	7	7.00	60.00
$B_6 \cup \mathbb{G}(64, 10\%)$	7	7.00	60.00	**6***	0.49	**6***	**6.00**	3.65
$B_6 \cup \mathbb{G}(64, 15\%)$	**6***	**6.00**	5.57	**6***	1.17	**6***	**6.00**	0.02
$B_6 \cup \mathbb{G}(64, 20\%)$	**6***	**6.00**	0.68	**6***	1.78	**6***	**6.00**	< 0.01
$B_6 \cup \mathbb{G}(64, 25\%)$	**6***	**6.00**	0.32	**6***	2.16	**6***	**6.00**	< 0.01
$B_7 \cup \mathbb{G}(128, 5\%)$	8	8.00	60.00	**7***	65.50	8	8.00	60.00
$B_7 \cup \mathbb{G}(128, 7.5\%)$	8	8.00	60.00	**7***	81.61	8	8.00	60.00
$B_7 \cup \mathbb{G}(128, 10\%)$	8	8.00	60.00	**7***	45.78	**7***	**7.00**	2.06
$B_7 \cup \mathbb{G}(128, 15\%)$	8	8.00	60.00	**7***	1643.97	**7***	**7.00**	0.04
$B_7 \cup \mathbb{G}(128, 20\%)$	**7***	**7.00**	2.09	**7***	204.40	**7***	**7.00**	0.01
$B_7 \cup \mathbb{G}(128, 25\%)$	**7***	**7.00**	1.41	**7***	239.89	**7***	**7.00**	< 0.01
$B_8 \cup \mathbb{G}(256, 5\%)$	**9**	**9.00**	< 0.01	**9**(11.11%)	3600	**9**	**9.00**	< 0.01
$B_8 \cup \mathbb{G}(256, 7.5\%)$	9	9.00	60.00	-	3600	**8***	**8.90**	56.79
$B_8 \cup \mathbb{G}(256, 10\%)$	9	9.00	60.00	-	3600	**8***	**8.00**	16.23
$B_8 \cup \mathbb{G}(256, 15\%)$	9	9.00	60.00	-	3600	**8***	**8.00**	0.07
$B_8 \cup \mathbb{G}(256, 20\%)$	9	9.00	60.00	-	3600	**8***	**8.00**	0.03
$B_8 \cup \mathbb{G}(256, 25\%)$	**8***	**8.00**	2.23	-	3600	**8***	**8.00**	0.01
$B_9 \cup \mathbb{G}(512, 5\%)$	**10**	**10.00**	0.01	**10**(10.00%)	3600	**10**	**10.00**	< 0.01
$B_9 \cup \mathbb{G}(512, 7.5\%)$	**10**	**10.00**	0.03	**10**(10.00%)	3600	**10**	**10.00**	< 0.01
$B_9 \cup \mathbb{G}(512, 10\%)$	10	10.00	60.00	-	3600	**9***	**9.30**	35.79
$B_9 \cup \mathbb{G}(512, 15\%)$	10	10.00	60.00	-	3600	**9***	**9.00**	0.53
$B_9 \cup \mathbb{G}(512, 20\%)$	10	10.00	60.00	-	3600	**9***	**9.00**	0.07
$B_9 \cup \mathbb{G}(512, 25\%)$	10	10.00	60.00	-	3600	**9***	**9.00**	0.02
$B_{10} \cup \mathbb{G}(1024, 5\%)$	**11**	**11.00**	0.11	-	3600	**11**	**11.00**	0.01
$B_{10} \cup \mathbb{G}(1024, 7.5\%)$	**11**	**11.00**	0.22	-	3600	**11**	**11.00**	0.01
$B_{10} \cup \mathbb{G}(1024, 10\%)$	11	11.00	60.00	-	3600	**10***	**10.50**	41.54
$B_{10} \cup \mathbb{G}(1024, 15\%)$	11	11.00	60.00	-	3600	**10***	**10.00**	3.64
$B_{10} \cup \mathbb{G}(1024, 20\%)$	**10***	**10.00**	2.58	-	3600	**10***	**10.00**	0.27
$B_{10} \cup \mathbb{G}(1024, 25\%)$	11	11.00	60.00	-	3600	**10***	**10.00**	0.08
SW-100-3-0d1-trial1	**61***	**61.00**	0.09	**61***	0.70	**61***	**61.00**	0.01
SW-100-3-0d2-trial1	**31***	**31.00**	0.02	**31***	0.33	**31***	**31.00**	< 0.01
SW-100-3-0d2-trial3	**31***	**31.00**	0.04	**31***	0.33	**31***	**31.00**	< 0.01
SW-100-4-0d1-trial1	10	10.00	60.00	**9***	0.84	**9***	**9.00**	0.24
SW-100-4-0d1-trial2	9	9.00	60.00	**8***	0.79	**8***	8.70	45.89
SW-100-4-0d1-trial3	11	11.00	60.00	**10***	0.77	**10***	**10.00**	0.07
SW-100-4-0d2-trial1	9	9.00	60.00	**8***	1.28	**8***	8.90	57.91
SW-100-4-0d2-trial2	9	9.00	60.00	**8***	3.78	9	9.00	60.00
SW-100-4-0d2-trial3	**9***	**9.00**	1.41	**9***	1.88	**9***	**9.00**	< 0.01
SW-100-4-0d3-trial1	9	9.00	60.00	**8***	1.20	**8***	**8.00**	1.43

Continued on next page

Table 2.7 – continued from previous page

Instance	ACS			ILP (Both bounds)		BRKGA_FRFS		
	Best	Avg.	t (s)	Best (GAP)	t (s)	Best	Avg.	t (s)
SW-100-4-0d3-trial2	**8***	**8.00**	0.90	**8***	16.23	**8***	**8.00**	0.01
SW-100-4-0d3-trial3	9	9.00	60.00	**8***	0.56	**8***	**8.00**	10.86
SW-100-5-0d1-trial1	10	10.00	60.00	**9***	0.56	**9***	**9.00**	5.00
SW-100-5-0d1-trial2	11	11.00	60.00	**10***	0.73	**10***	**10.00**	0.05
SW-100-5-0d1-trial3	13	13.00	60.00	**12***	0.32	**12***	**12.00**	0.05
SW-100-5-0d2-trial1	10	10.00	60.00	**9***	1.59	10	10.00	60.00
SW-100-5-0d2-trial2	10	10.00	60.00	**9***	0.42	10	10.00	60.00
SW-100-5-0d2-trial3	9	9.00	60.00	**8***	2.62	9	9.00	60.00
SW-100-5-0d3-trial1	9	9.00	60.00	**8***	3.22	**8***	**8.00**	0.12
SW-100-5-0d3-trial2	**8**	**8.00**	0.36	**8**(12.50%)	3600	**8**	**8.00**	< 0.01
SW-100-5-0d3-trial3	**8***	**8.00**	4.29	**8***	11.25	**8***	**8.00**	< 0.01
SW-100-6-0d1-trial1	8	8.00	60.00	**7***	4.35	8	8.00	60.00
SW-100-6-0d1-trial2	9	9.00	60.00	**8***	870.41	**8***	**8.00**	0.02
SW-100-6-0d1-trial3	9	9.00	60.00	**7***	2.94	8	8.00	60.00
SW-100-6-0d2-trial1	8	8.00	60.00	**7***	1.64	**7***	7.10	17.88
SW-100-6-0d2-trial2	8	8.00	60.00	**7***	0.11	**7***	**7.00**	0.81
SW-100-6-0d2-trial3	8	8.00	60.00	**7***	1.65	**7***	**7.00**	6.58
SW-100-6-0d3-trial1	8	8.00	60.00	**7***	1.53	**7***	**7.00**	1.31
SW-100-6-0d3-trial2	8	8.00	60.00	**7***	1.28	**7***	**7.00**	0.56
SW-100-6-0d3-trial3	8	8.00	60.00	**7***	0.86	**7***	**7.00**	0.11
SW-1000-3-0d2-trial1	96	96.00	60.00	-	3600	**89***	**89.00**	14.42
SW-1000-3-0d2-trial2	94	94.00	60.00	-	3600	**88***	**88.00**	9.26
SW-1000-3-0d3-trial2	96	96.00	60.00	-	3600	**87***	**87.00**	9.39
SW-1000-4-0d1-trial1	19	19.00	60.00	-	3600	**17**	**17.00**	5.68
SW-1000-4-0d1-trial2	20	20.00	60.00	-	3600	**17**	**17.90**	55.04
SW-1000-4-0d1-trial3	20	20.00	60.00	-	3600	**18**	**18.00**	5.07
SW-1000-4-0d2-trial1	15	15.00	60.00	-	3600	**14**	**14.30**	41.90
SW-1000-4-0d2-trial2	16	16.00	60.00	-	3600	**14**	**14.80**	49.96
SW-1000-4-0d2-trial3	16	16.00	60.00	-	3600	**15**	**15.20**	28.21
SW-1000-4-0d3-trial1	14	14.00	60.00	**12**(16.67%)	3600	13	13.00	60.00
SW-1000-4-0d3-trial3	14	14.00	60.00	**13**(23.08%)	3600	**13**	**13.00**	0.37
SW-1000-5-0d1-trial1	20	20.00	60.00	-	3600	**17**	**17.10**	15.22
SW-1000-5-0d1-trial2	20	20.00	60.00	-	3600	**17**	**17.00**	13.70
SW-1000-5-0d1-trial3	18	18.00	60.00	**15**(24.38%)	3600	16	16.00	60.00
SW-1000-5-0d2-trial1	16	16.00	60.00	-	3600	**15**	**15.00**	0.10
SW-1000-5-0d2-trial2	16	16.00	60.00	-	3600	**14**	**14.90**	54.75
SW-1000-5-0d2-trial3	15	15.00	60.00	-	3600	**14**	**14.00**	1.25
SW-1000-5-0d3-trial1	14	14.00	60.00	**12**(16.67%)	3600	13	13.90	60.00
SW-1000-5-0d3-trial2	14	14.00	60.00	-	3600	**13**	**13.20**	26.52
SW-1000-5-0d3-trial3	**14**	**14.00**	17.22	-	3600	**14**	**14.00**	0.04
SW-1000-6-0d1-trial1	16	16.00	60.00	-	3600	**14**	**14.00**	16.05
SW-1000-6-0d1-trial2	15	15.00	60.00	-	3600	**13**	**13.90**	57.65
SW-1000-6-0d1-trial3	14	14.00	60.00	-	3600	**13**	**13.00**	6.31
SW-1000-6-0d2-trial1	13	13.00	60.00	-	3600	**12**	**12.00**	0.13
SW-1000-6-0d2-trial2	14	14.00	60.00	-	3600	**13**	**13.00**	0.01
SW-1000-6-0d2-trial3	13	13.00	60.00	-	3600	**12**	**12.00**	0.24
SW-1000-6-0d3-trial1	**12**	**12.00**	0.12	-	3600	**12**	**12.00**	0.01
SW-1000-6-0d3-trial2	**12**	**12.00**	5.33	-	3600	**12**	**12.00**	0.01
SW-1000-6-0d3-trial3	**12**	**12.00**	3.99	-	3600	**12**	**12.00**	0.01
Avg. t (s)		40.93		1742.87			**17.94**	
# Best		33		61			**85**	
# Best Avg.		33		61			**79**	

The results in Table 2.7 confirm the robustness of BRKGA_FRFS and its superiority over ACS and the ILP model. Even with an upper bound set by BRKGA_FRFS, the ILP model could not produce any feasible results for instances with 256 vertices. BRKGA_FRFS proved the optimal solution in 56 instances, whereas this number was 53 for the ILP model. Moreover, our algorithm outperformed ACS in terms of both solution quality and CPU time. It attained the best solution in 85 out of the 101 instances. It is worthwhile mentioning that the synthetic instances proposed in this paper are indeed very challenging, as the ILP model could only solve a small portion of them. Hence, we think the proposed benchmark with known optima is a very good one to evaluate future new exact methods.

2.3.6 Experiments with SCHA decoding and the proposed matheuristic

As shown in the previous results, our basic BRKGA_FRFS algorithm outperforms the best-performing algorithms in the literature. In this subsection, we show that BRKGA_FRFS results can be improved using SCHA decoding and our matheuristic. Table 2.8 compares: (i) BRKGA without SCHA (BRKGA_FRFS), (ii) BRKGA with SCHA, (iii) proposed matheuristic without SCHA (BRKGA_FRFS+ILP), and (iv) proposed matheuristic with SCHA (BRKGA+ILP).

Table 2.8: Comparative results of BRKGAs and hybrids

Instance	BRKGA_FRFS			BRKGA			BRKGA_FRFS+ILP			BRKGA+ILP		
	Best	Avg.	t (s)	Best	Avg.	t (s)	Best	Avg.	t (s)	Best	Avg.	t (s)
B_4	**4***	**4.00**	< 0.01	**4***	**4.00**	< 0.01	**4***	**4.00**	< 0.01	**4***	**4.00**	< 0.01
B_5	**5***	**5.00**	2.03	**5***	**5.00***	< 0.01	**5***	**5.00**	0.05	**5***	**5.00**	< 0.01
B_6	**6***	6.80	55.04	**6***	**6.00**	< 0.01	**6***	6.10	6.25	**6***	**6.00**	< 0.01
B_7	8	8.80	60.00	**7***	**7.00**	< 0.01	**7***	7.20	7.15	**7***	**7.00**	< 0.01
B_8	10	10.60	60.00	**8***	**8.00**	< 0.01	**8***	**8.00**	6.02	**8***	**8.00**	< 0.01
B_9	12	12.30	60.00	**9***	**9.00**	< 0.01	**9***	9.30	40.91	**9***	**9.00**	< 0.01
$B_5 \cup \mathbb{G}(32, 5\%)$	**5***	5.90	58.45	**5***	5.10	19.41	**5***	**5.00**	0.12	**5***	**5.00**	0.17
$B_5 \cup \mathbb{G}(32, 7.5\%)$	**5***	5.10	15.21	**5***	**5.00**	3.45	**5***	**5.00**	0.25	**5***	**5.00**	0.22
$B_5 \cup \mathbb{G}(32, 10\%)$	**5***	**5.00**	0.15	**5***	**5.00**	0.04	**5***	**5.00**	0.14	**5***	**5.00**	0.04
$B_5 \cup \mathbb{G}(32, 15\%)$	**5***	**5.00**	0.02	**5***	**5.00**	0.06	**5***	**5.00**	0.05	**5***	**5.00**	0.09
$B_5 \cup \mathbb{G}(32, 20\%)$	**5***	**5.00**	< 0.01	**5***	**5.00**	< 0.01	**5***	**5.00**	< 0.01	**5***	**5.00**	< 0.01
$B_5 \cup \mathbb{G}(32, 25\%)$	**5***	**5.00**	< 0.01	**5***	**5.00**	< 0.01	**5***	**5.00**	< 0.01	**5***	**5.00**	< 0.01
$B_6 \cup \mathbb{G}(64, 5\%)$	7	7.00	60.00	7	7.00	60.00	**6***	6.10	9.96	**6***	**6.00**	5.47
$B_6 \cup \mathbb{G}(64, 7.5\%)$	7	7.00	60.00	7	7.00	60.00	**6***	**6.00**	6.77	**6***	**6.00**	11.97
$B_6 \cup \mathbb{G}(64, 10\%)$	**6***	**6.00**	3.65	**6***	**6.00**	4.58	**6***	**6.00**	3.87	**6***	**6.00**	3.69
$B_6 \cup \mathbb{G}(64, 15\%)$	**6***	**6.00**	0.02	**6***	**6.00**	0.03	**6***	**6.00**	0.02	**6***	**6.00**	0.03
$B_6 \cup \mathbb{G}(64, 20\%)$	**6***	**6.00**	< 0.01	**6***	**6.00**	< 0.01	**6***	**6.00**	< 0.01	**6***	**6.00**	< 0.01
$B_6 \cup \mathbb{G}(64, 25\%)$	**6***	**6.00**	< 0.01	**6***	**6.00**	< 0.01	**6***	**6.00**	< 0.01	**6***	**6.00**	< 0.01
$B_7 \cup \mathbb{G}(128, 5\%)$	**8**	**8.00**	< 0.01	**8**	**8.00**	< 0.01	**8**	**8.00**	< 0.01	**8**	**8.00**	< 0.01
$B_7 \cup \mathbb{G}(128, 7.5\%)$	8	8.00	60.00	**7***	**7.80**	52.93	8	8.00	60.00	8	8.00	60.00
$B_7 \cup \mathbb{G}(128, 10\%)$	**7***	**7.00**	2.06	**7***	**7.00**	6.25	**7***	**7.00**	23.98	**7***	7.40	33.13
$B_7 \cup \mathbb{G}(128, 15\%)$	**7***	**7.00**	0.04	**7***	**7.00**	0.07	**7***	**7.00**	0.05	**7***	**7.00**	0.08
$B_7 \cup \mathbb{G}(128, 20\%)$	**7***	**7.00**	0.01	**7***	**7.00**	0.01	**7***	**7.00**	0.01	**7***	**7.00**	0.01
$B_7 \cup \mathbb{G}(128, 25\%)$	**7***	**7.00**	< 0.01	**7***	**7.00**	< 0.01	**7***	**7.00**	< 0.01	**7***	**7.00**	< 0.01
$B_8 \cup \mathbb{G}(256, 5\%)$	**9**	**9.00**	< 0.01	**9**	**9.00**	< 0.01	**9**	**9.00**	< 0.01	**9**	**9.00**	< 0.01
$B_8 \cup \mathbb{G}(256, 7.5\%)$	**8***	**8.90**	56.79	**8***	**8.90**	55.27	9	9.00	60.00	**8***	**8.90**	57.20

Continued on next page

Table 2.8 – continued from previous page

Instance	BRKGA_FRFS			BRKGA			BRKGA_FRFS+ILP			BRKGA+ILP		
	Best	Avg.	t (s)	Best	Avg.	t (s)	Best	Avg.	t (s)	Best	Avg.	t (s)
$B_8 \cup \mathbb{G}(256, 10\%)$	**8***	**8.00**	16.23	**8***	8.30	34.42	**8***	8.40	38.81	**8***	8.60	42.42
$B_8 \cup \mathbb{G}(256, 15\%)$	**8***	**8.00**	0.07	**8***	**8.00**	0.29	**8***	**8.00**	0.08	**8***	**8.00**	0.30
$B_8 \cup \mathbb{G}(256, 20\%)$	**8***	**8.00**	0.03	**8***	**8.00**	0.04	**8***	**8.00**	0.04	**8***	**8.00**	0.05
$B_8 \cup \mathbb{G}(256, 25\%)$	**8***	**8.00**	0.01	**8***	**8.00**	0.01	**8***	**8.00**	0.01	**8***	**8.00**	0.01
$B_9 \cup \mathbb{G}(512, 5\%)$	**10**	**10.00**	< 0.01	**10**	**10.00**	< 0.01	**10**	**10.00**	< 0.01	**10**	**10.00**	< 0.01
$B_9 \cup \mathbb{G}(512, 7.5\%)$	**10**	**10.00**	< 0.01	**10**	**10.00**	< 0.01	**10**	**10.00**	< 0.01	**10**	**10.00**	< 0.01
$B_9 \cup \mathbb{G}(512, 10\%)$	**9***	**9.30**	35.79	**9***	9.70	50.77	**9***	9.50	38.45	**9***	9.70	51.38
$B_9 \cup \mathbb{G}(512, 15\%)$	**9***	**9.00**	0.53	**9***	**9.00**	1.05	**9***	**9.00**	0.62	**9**	**9.00**	1.18
$B_9 \cup \mathbb{G}(512, 20\%)$	**9***	**9.00**	0.07	**9***	**9.00**	0.09	**9***	**9.00**	0.08	**9***	**9.00**	0.10
$B_9 \cup \mathbb{G}(512, 25\%)$	**9***	**9.00**	0.02	**9***	**9.00**	0.03	**9***	**9.00**	0.03	**9***	**9.00**	0.03
$B_{10} \cup \mathbb{G}(1024, 5\%)$	**11**	**11.00**	0.01	**11**	**11.00**	0.01	**11**	**11.00**	0.01	**11**	**11.00**	0.02
$B_{10} \cup \mathbb{G}(1024, 7.5\%)$	**11**	**11.00**	0.01	**11**	**11.00**	0.01	**11**	**11.00**	0.02	**11**	**11.00**	0.01
$B_{10} \cup \mathbb{G}(1024, 10\%)$	**10***	**10.50**	41.54	11	11.00	60.00	**10***	**10.50**	42.99	11	11.00	60.00
$B_{10} \cup \mathbb{G}(1024, 15\%)$	**10***	**10.00**	3.64	**10***	**10.00**	4.58	**10***	**10.00**	4.14	**10***	**10.00**	5.09
$B_{10} \cup \mathbb{G}(1024, 20\%)$	**10***	**10.00**	0.27	**10***	**10.00**	0.31	**10***	**10.00**	0.31	**10***	**10.00**	0.36
$B_{10} \cup \mathbb{G}(1024, 25\%)$	**10***	**10.00**	0.08	**10***	**10.00**	0.09	**10***	**10.00**	0.09	**10***	**10.00**	0.10
SW-100-3-0d1-trial1	**61***	**61.00**	0.01	**61***	**61.00**	< 0.01	**61***	**61.00**	0.01	**61***	**61.00**	< 0.01
SW-100-3-0d2-trial1	**31***	**31.00**	< 0.01	**31***	**31.00**	< 0.01	**31***	**31.00**	< 0.01	**31***	**31.00**	< 0.01
SW-100-3-0d2-trial3	**31***	**31.00**	< 0.01	**31***	**31.00**	< 0.01	**31**	**31.00**	< 0.01	**31***	**31.00**	< 0.01
SW-100-4-0d1-trial1	**9***	**9.00**	0.24	**9***	**9.00**	< 0.01	**9***	**9.00**	0.28	**9***	**9.00**	0.01
SW-100-4-0d1-trial2	**8***	8.70	45.89	**8***	**8.00**	3.10	**8***	**8.00**	1.44	**8***	**8.00**	1.42
SW-100-4-0d1-trial3	**10***	**10.00**	0.07	**10***	**10.00**	0.01	**10***	**10.00**	0.07	**10***	**10.00**	0.02
SW-100-4-0d2-trial1	**8***	8.90	57.91	**8***	8.50	43.80	**8***	**8.00**	1.62	**8***	**8.00**	3.18
SW-100-4-0d2-trial2	9	9.00	60.00	9	9.00	60.00	**8***	**8.00**	6.14	**8***	**8.00**	7.48
SW-100-4-0d2-trial3	**9***	**9.00**	< 0.01	**9***	**9.00**	< 0.01	**9***	**9.00**	< 0.01	**9***	**9.00**	< 0.01
SW-100-4-0d3-trial1	**8***	**8.00**	1.43	**8***	**8.00**	1.00	**8***	**8.00**	1.01	**8***	**8.00**	0.76
SW-100-4-0d3-trial2	**8***	**8.00**	0.01	**8***	**8.00**	< 0.01	**8***	**8.00**	0.01	**8***	**8.00**	< 0.01
SW-100-4-0d3-trial3	**8***	**8.00**	10.86	**8***	**8.00**	1.44	**8***	**8.00**	1.33	**8***	**8.00**	0.76
SW-100-5-0d1-trial1	**9***	**9.00**	5.00	**9***	**9.00**	0.08	**9***	**9.00**	0.83	**9***	**9.00**	0.10
SW-100-5-0d1-trial2	**10***	**10.00**	0.05	**10***	**10.00**	< 0.01	**10***	**10.00**	0.05	**10***	**10.00**	< 0.01
SW-100-5-0d1-trial3	**12***	**12.00**	0.05	**12***	**12.00**	< 0.01	**12***	**12.00**	0.05	**12***	**12.00**	< 0.01
SW-100-5-0d2-trial1	10	10.00	60.00	10	10.00	60.00	**9***	**9.00**	2.08	**9***	**9.00**	3.34
SW-100-5-0d2-trial2	10	10.00	60.00	10	10.00	60.00	**9***	**9.00**	1.50	**9***	**9.00**	2.45
SW-100-5-0d2-trial3	9	9.00	60.00	9	9.00	60.00	**8***	**8.00**	3.72	**8***	**8.00**	6.96
SW-100-5-0d3-trial1	**8***	**8.00**	0.12	**8***	**8.00**	0.03	**8***	**8.00**	0.13	**8***	**8.00**	0.02
SW-100-5-0d3-trial2	**8**	**8.00**	< 0.01	**8**	**8.00**	< 0.01	**8**	**8.00**	< 0.01	**8**	**8.00**	< 0.01
SW-100-5-0d3-trial3	**8***	**8.00**	< 0.01	**8***	**8.00**	< 0.01	**8***	**8.00**	< 0.01	**8***	**8.00**	< 0.01
SW-100-6-0d1-trial1	8	8.00	60.00	8	8.00	60.00	**7***	**7.50**	48.87	**7***	**7.50**	44.86
SW-100-6-0d1-trial2	**8***	**8.00**	0.02	**8***	**8.00**	< 0.01	**8***	**8.00**	0.03	**8***	**8.00**	< 0.01
SW-100-6-0d1-trial3	8	8.00	60.00	8	8.00	60.00	**7***	**7.00**	13.55	**7***	**7.00**	15.28
SW-100-6-0d2-trial1	**7***	7.10	17.88	**7***	7.20	36.35	**7***	**7.00**	7.39	**7***	**7.00**	8.02
SW-100-6-0d2-trial2	**7***	**7.00**	0.81	**7***	**7.00**	0.35	**7***	**7.00**	5.87	**7***	**7.00**	0.44
SW-100-6-0d2-trial3	**7***	**7.00**	6.58	**7***	**7.00**	8.53	**7***	**7.00**	10.50	**7***	**7.00**	6.92
SW-100-6-0d3-trial1	**7***	**7.00**	1.31	**7***	**7.00**	0.89	**7***	**7.00**	9.88	**7***	**7.00**	5.60
SW-100-6-0d3-trial2	**7***	**7.00**	0.56	**7***	**7.00**	0.40	**7***	**7.00**	4.91	**7***	**7.00**	0.45
SW-100-6-0d3-trial3	**7***	**7.00**	0.11	**7***	**7.00**	0.06	**7***	**7.00**	0.76	**7***	**7.00**	0.08
SW-1000-3-0d2-trial1	**89***	**89.00**	14.42	**89***	**89.00**	0.01	**89***	**89.00**	14.81	**89***	**89.00**	0.02
SW-1000-3-0d2-trial2	**88***	**88.00**	9.26	**88***	**88.00**	0.01	**88***	**88.00**	9.34	**88***	**88.00**	0.01
SW-1000-3-0d3-trial2	**87***	**87.00**	9.39	**87***	**87.00**	0.01	**87***	**87.00**	9.43	**87***	**87.00**	0.02
SW-1000-4-0d1-trial1	17	17.00	60.00	**16**	**16.60**	46.05	17	17.00	60.00	**16**	**16.60**	44.75
SW-1000-4-0d1-trial2	**17**	17.90	55.04	**17**	**17.00**	0.42	**17**	17.90	55.05	**17**	**17.00**	0.67
SW-1000-4-0d1-trial3	18	18.00	60.00	**17**	17.80	53.42	18	18.00	60.00	**17**	**17.70**	53.47
SW-1000-4-0d2-trial1	**14**	14.30	41.90	**14**	**14.00**	0.02	**14**	14.30	42.26	**14**	**14.00**	0.02

Continued on next page

Table 2.8 – continued from previous page

Instance	BRKGA_FRFS			BRKGA			BRKGA_FRFS+ILP			BRKGA+ILP		
	Best	Avg.	t (s)	Best	Avg.	t (s)	Best	Avg.	t (s)	Best	Avg.	t (s)
SW-1000-4-0d2-trial2	**14**	14.80	49.96	**14**	**14.00**	0.17	**14**	14.80	49.98	**14**	**14.00**	0.17
SW-1000-4-0d2-trial3	**15**	15.20	28.21	**15**	**15.00**	0.01	**15**	15.20	28.29	**15**	**15.00**	0.02
SW-1000-4-0d3-trial1	**13**	**13.00**	0.60	**13**	**13.00**	0.02	**13**	**13.00**	0.60	**13**	**13.00**	0.01
SW-1000-4-0d3-trial3	**13**	**13.00**	0.37	**13**	**13.00**	0.02	**13**	**13.00**	0.37	**13**	**13.00**	0.02
SW-1000-5-0d1-trial1	**17**	17.10	15.22	**17**	**17.00**	0.05	**17**	17.10	15.36	**17**	**17.00**	0.06
SW-1000-5-0d1-trial2	**17**	**17.00**	13.70	**17**	**17.00**	0.14	**17**	**17.00**	14.96	**17**	**17.00**	0.15
SW-1000-5-0d1-trial3	16	16.00	60.00	**15**	**15.00**	0.65	16	16.00	60.00	**15**	**15.00**	1.09
SW-1000-5-0d2-trial1	15	15.00	60.00	**14**	**14.00**	2.39	15	15.00	60.00	**14**	**14.00**	2.58
SW-1000-5-0d2-trial2	**14**	14.90	54.75	**14**	**14.00**	0.07	**14**	14.90	54.77	**14**	**14.00**	0.07
SW-1000-5-0d2-trial3	**14**	**14.00**	1.25	**14**	**14.00**	0.01	**14**	**14.00**	1.26	**14**	**14.00**	0.02
SW-1000-5-0d3-trial1	**13**	13.90	55.65	**13**	**13.00**	0.05	**13**	13.90	55.67	**13**	**13.00**	0.05
SW-1000-5-0d3-trial2	**13**	13.20	26.52	**13**	**13.00**	0.03	**13**	13.20	26.64	**13**	**13.00**	0.03
SW-1000-5-0d3-trial3	14	14.00	60.00	**13**	**13.00**	5.81	14	14.00	60.00	**13**	**13.00**	3.22
SW-1000-6-0d1-trial1	**14**	**14.00**	16.05	**14**	**14.00**	0.39	**14**	14.10	18.35	**14**	**14.00**	0.61
SW-1000-6-0d1-trial2	**13**	13.90	57.65	**13**	13.20	28.66	**13**	13.90	58.20	**13**	**13.00**	26.01
SW-1000-6-0d1-trial3	**13**	**13.00**	6.31	**13**	**13.00**	0.08	**13**	**13.00**	7.24	**13**	**13.00**	0.08
SW-1000-6-0d2-trial1	**12**	**12.00**	0.13	**12**	**12.00**	0.02	**12**	**12.00**	0.14	**12**	**12.00**	0.02
SW-1000-6-0d2-trial2	**13**	**13.00**	0.01	**13**	**13.00**	0.01	**13**	**13.00**	0.01	**13**	**13.00**	0.01
SW-1000-6-0d2-trial3	**12**	**12.00**	0.24	**12**	**12.00**	0.03	**12**	**12.00**	0.24	**12**	**12.00**	0.03
SW-1000-6-0d3-trial1	**12**	**12.00**	0.01	**12**	**12.00**	0.01	**12**	**12.00**	0.01	**12**	**12.00**	0.01
SW-1000-6-0d3-trial2	**12**	**12.00**	0.01	**12**	**12.00**	0.01	**12**	**12.00**	0.01	**12**	**12.00**	0.01
SW-1000-6-0d3-trial3	**12**	**12.00**	0.01	**12**	**12.00**	0.01	**12**	**12.00**	0.01	**12**	**12.00**	0.01
# Best		84			92			94			**99**	
# Best Avg.		69			85			78			**96**	
Avg. t (s)		18.98			9.98			12.24			**5.69**	

The results in Table 2.8 show that the SCHA decoding positively impacts the search since the algorithms with SCHA performed better than their counterparts without SCHA. The SCHA refinement gives more accuracy to the BRKGA decoding. Moreover, the results reveal that BRKGA+ILP compares favorably with BRKGA in both features, solution quality and CPU time. Note that in the cases BRKGA+ILP missed the best solution, the solution determined by BRKGA+ILP is at most one unit time longer than the best one. Adding ILP features to the BRKGA has shown a very feasible approach.

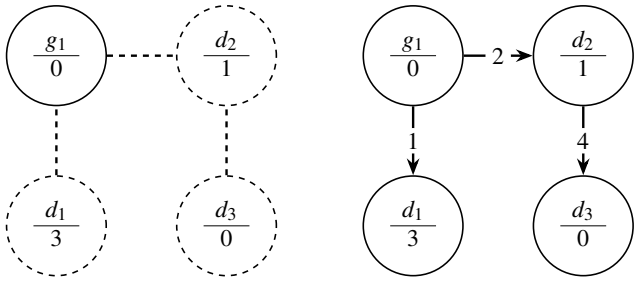
3

Weighted Minimum Broadcast Time

The Weighted Minimum Broadcast Time (WMBT) is a generalization of the MBT with weight in vertices or/and edges. The WMBT model is more realistic than the MBT. Because the WMBT can represent transmission delay rates as explained in Section 1.2. We remark that our work focuses on the telephone model.

There are four variations of the WMBT: (i) classic model ($\alpha = 0$ and $\beta = 1$ for all edges), (ii) weighted-vertex model (Harutyunyan and Kamali, 2008) ($\alpha \geq 0$ and $\beta = 1$ for all edges), (iii) weighted-edge model (Su et al., 2010) ($\alpha = 0$ and $\beta \geq 1$ for all edges), and (iv) weighted-vertex-and-edge (general) model ($\alpha \geq 0$ and $\beta \geq 1$ for all edges). The α and β were defined in Section 1.2. The first model was presented in Section 2.2. In this chapter, we will propose solutions for the remaining models.

Figures 3.1–3.3 illustrate an example for each variant of the WMBT. Figure 3.1-a shows a network with weighted-vertex, such that each vertex has a label and setup phase. Figure 3.1-b shows an optimal solution of example of the previous example. The values in edges represent the time that vertex received information.



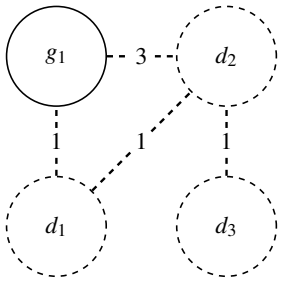
(a) Example network with weighted-vertex

(b) Optimal broadcast solution.

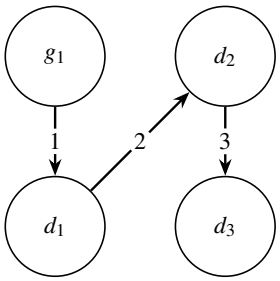
Figure 3.1: Example of weighted-vertex model.

Figure 3.2-a shows a network with a weighted-edge graph. Each edge is associated with a value representing the transmission time of connection. Similarly, Figure 3.1-b shows optimal

solution of the previous example. The values in edges represent the time that each vertex received information.



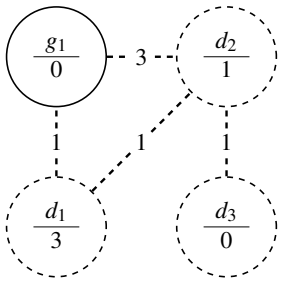
(a) Example network with weighted-edge



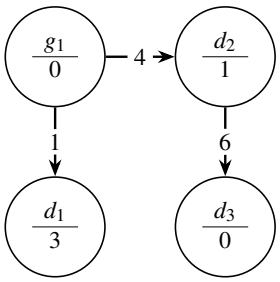
(b) Optimal broadcast solution.

Figure 3.2: Example of weighted-edge model.

Figure 3.3-a shows an example of a weighted-edge-and-vertex variant. Edges are associated with a transmission time, whereas vertices are associated with a setup time. Figure 3.1-b shows optimal solution of the previous example.



(a) Example network with weighted-edge-and-vertex



(b) Optimal broadcast solution.

Figure 3.3: Example of weighted-edge-and-vertex model.

Theorem 9. *The weighted-edge, weighted-vertex and weighted-edge-and-vertex Minimum Broadcast Problems are $\mathcal{NP}$-Complete.*

Proof. Restrict each WMTB variant to classical MBT by allowing only instances having $w_e = 1$ for all $e \in E(G)$ and $w_v = 0$ for all $v \in V(G)$. □

We remark that the classical MBT can be viewed as telephone or postal models when $\alpha = 0$ and $\beta = 1$ for all edges. Harutyunyan and Kamali (2008) introduced a generalization of the telephone model with the weighted-vertex model, where each vertex has a different α value.

In both, telephone and postal models, only one transmission can be performed at a time. Because of this, these models are labeled as 1-broadcast or 1-port. In addition, a model in which a maximum of k transmission can be performed at a time, it called k-broadcast or k-port

(König and Lazard, 1994). This work aims to study the telephone model as a message-passing system for the MBT.

We remark that Definitions 1–7 are related to the MBT when $|V_0| = 1$. However, similar considerations can be applied, *mutatis mutandis*, for the WMBT or/and when $|V_0| > 1$.

3.1 Related Work

Harutyunyan and Kamali (2008); Koh and Tcha (1991) introduced generalizations of the Telephone model. Koh and Tcha (1991) show a polynomial algorithm that optimally solves the weighted-edge model for the tree graphs. Harutyunyan and Kamali (2008) presented a greedy algorithm and an evolutionary algorithm for the general graphs with setup phase (weighted-vertex). The greedy algorithm outperformed the evolutionary algorithm. Averbuch et al. (2000) propose polynomial algorithms for weighted-vertex and weighted-edge trees, but the cost function is computed differently than Garey and Johnson (1979); Harutyunyan and Kamali (2008); Koh and Tcha (1991). To the best of the authors' knowledge, there are no exact algorithms for any generalization of the WMBT. Also, there are no heuristic or metaheuristic algorithms for general graphs with transmission time and setup phase (weighted-vertex-and-edge).

Harutyunyan and Kamali (2008) introduce a greedy algorithm for the weighted-vertex model, which we will show in Algorithm 6. This algorithm is an adaptation of Dijkstra's algorithm for Weighted-Vertex MBT. It starts setting the cost of all vertices to infinity, creates sets to informed and uninformed vertices, and sets the cost of source v_0 (lines 2-6). Next, the procedure may be repeated if there is any uninformed vertex, choosing an edge with the least weight that does not cycle in the solution (lines 7-12). Finally, *broadcastTime* gets the biggest cost of vertices (line 13). Overall, the worst-case running time of this algorithm is $O(|V| \cdot |E|)$.

Harutyunyan and Kamali (2008) described that even if it finds the optimal broadcast tree, the algorithm does not guarantee that the WMBT schedule will be the optimum. In Section 3.2.4, we describe an exact algorithm for the WMBT for trees.

We present a summarized comparison of the literature's approaches. Table 3.1 lists the main articles about WMBT in the literature.

Table 3.1: List of articles about WMBT in the literature.

Article	Approach	Observation
Bar-Noy et al. (2000)	Description of message-passing systems	-
Harutyunyan and Kamali (2008)	Heuristic and metaheuristic algorithm	For weighted-vertex.
Su et al. (2016)	Polynomial time algorithm	For edge-vertex, when graph is a tree and $\|V_0\| = 1$.

Algorithm 6: Greedy algorithm for Weighted-Vertex-MBT

Input : Weighted Vertex Undirected graph: $G = (V, E, W_v)$,
Source vertex: v_0
Output: Broadcast time: $broadcastTime$

```
1  GreedyAlgorithm(G, v_0)
2      for v ∈ V do
3          cost[v] ← ∞
4      I ← {v_0}                                          // let I be a set
5      U ← V \ {v_0}                                      // let U be a set
6      cost[v_0] ← W_v[v_0]
7      while U ≠ ∅ do
8          v_α, v_β ← argmin_{i∈I, j∈U, (i,j)∈E} (c(v_i) + w(v_j))
9          cost(v_β) ← cost(v_α) + w(v_β) + 1
10         cost(v_α) ← cost(v_α) + 1
11         I ← I ∪ {v_β}
12         U ← U \ {v_β}
13     broadcastTime ← max_{v∈V} (cost[v])                // Get higher cost
14
15     return broadcastTime
```

3.2 Algorithmic approaches for the WMBT

In this Section, we will show our contributions. First, we will explain about our mathematical models for WMBTs in Section 3.2.1. Section 3.2.2 describes a lower bound algorithm for the WMBT. Section 3.2.3 shows an exact greedy algorithm for the WMBT for forests. Section 3.2.4 presents a greedy algorithm for the WMBT for general instances. Section 3.2.5 describes a reduce rule for the WBMT. Finally, Section 3.2.6 some decoders for `BRKGA` for the WMBT.

3.2.1 Mathematical models for WMBT

To the best of the author's knowledge, there is no mathematical model for WMBTs. In this section, we proposed a mathematical model for each variant of the WMBT: (i) weighted-vertex model, (ii) weighted-edge model, and (iii) general model. These mathematical models are extensions of the model proposed by de Sousa et al. (2018) for the MBT.

3.2.1.1 Weighted-vertex model

Let $G = (V, E)$ be a simple graph, $N(i)$ the set of neighboring vertices of vertex $i \in V$, and W_v the list of weights for each vertex of G. In our model, K_i is a binary constant indicating whether or not vertex i is in V_0, and T_{max} a constant that represents an upper bound on the time in which a vertex receives the message (e.g., $T_{max} = (\sum_{v \in V} W_v(v)) + |V| - |V_0|$ is the trivial bound). Finally,

define T as a decision variable that represents the minimum broadcast time, and x_{ij}^t as a binary variable that has value 1 if the vertex i starts transmission of the message to the vertex j in time t and 0, otherwise. Based on these elements, our ILP model is defined as follows:

$$\min \quad T \tag{3.1}$$

$$\text{s. t} \quad K_i + \sum_{j \in N(i)} \sum_{t=0}^{T_{max}} x_{ji}^t = 1 \qquad \forall i \in V \tag{3.2}$$

$$\sum_{j \in N(i)} x_{ij}^t = 0 \qquad \forall i \in V_0, \forall t \in [0, W_v(i)) \tag{3.3}$$

$$\sum_{j \in N(i)} x_{ij}^t \leq 1 \qquad \forall i \in V, \forall t \in [0, T_{max}] \tag{3.4}$$

$$x_{ij}^t \leq K_i + \sum_{k \in N(i) \setminus \{j\}} \sum_{\tau=0}^{t-1-W_v(i)} x_{ki}^\tau \qquad \forall (i,j) \in E, \forall t \in [0, T_{max}] \tag{3.5}$$

$$\sum_{t=0}^{T_{max}} (t + 1 + W_v(j)) \cdot x_{ij}^t \leq T \qquad \forall (i,j) \in E \tag{3.6}$$

$$T \in \mathbb{N} \tag{3.7}$$

$$x_{ij}^t \in \mathbb{B} \qquad \forall (i,j) \in E, \forall t \in [0, T_{max}] \tag{3.8}$$

The objective function (3.1) minimizes the weighted broadcast time. Constraints (3.2) limit each vertex to receive only one message or start with its. Constraints (3.3) imply that each source sends the message after its setup phase. Constraints (3.4) require that each vertex sends at most one message to a neighbor at each time t. Constraints (3.5) establish that each vertex can only transmit if it has already received the message. Constraints (3.6) state that the value of T must be greater than or equal to the time of any transmission. Finally, Constraints (3.7) and (3.8) define the domain of the decision variables.

3.2.1.2 Weighted-edge model

In the weighted-edge variation, there is a list W_e with weights for each edge of the graph. The ILP model for this variation is as follows:

$$\min \quad T \tag{3.9}$$

$$\text{s. t} \quad K_i + \sum_{j \in N(i)} \sum_{t=1}^{T_{max}} x_{ji}^t = 1 \qquad \forall i \in V \tag{3.10}$$

$$\sum_{j \in N(i)} x_{ij}^t \leq 1 \qquad \forall i \in V, \forall t \in [0, T_{max}] \tag{3.11}$$

$$x_{ij}^{t} \leq K_i + \sum_{k \in N(i) \setminus \{j\}} \sum_{\tau=0}^{t-W_e((k,i))} x_{ki}^{\tau} \qquad \forall (i,j) \in E, \forall t \in [0, T_{max}] \quad (3.12)$$

$$\sum_{t=0}^{T_{max}} (t + W_e((i,j))) \cdot x_{ij}^{t} \leq T \qquad \forall (i,j) \in E \quad (3.13)$$

$$x_{ik}^{\tau} \leq 1 - x_{ij}^{t} \qquad \forall (i,j) \in E, \forall t \in [0, T_{max}], \forall k \in N(i), \forall \tau \in [t+1, t+W_e(i,j)] \quad (3.14)$$

$$T \in \mathbb{N} \quad (3.15)$$

$$x_{ij}^{t} \in \mathbb{B} \qquad \forall (i,j) \in E, \forall t \in [0, T_{max}] \quad (3.16)$$

The objective function (3.9) minimizes the weighted broadcast time. Constraints (3.10) limit each vertex to receive only one message or start with its, where $T_{max} = \sum_{e \in E} W_e(e)$ is the trivial upper bound. Constraints (3.11) require that each vertex sends at most one message to a neighbor at each time t. Constraints (3.12) establish that each vertex can only transmit if it has already received the message. Constraints (3.13) state that the value of T must be greater than or equal to the time of any transmission. Constraints (3.14) mean each vertex wait $W_e(i,j)$ units of time for a new transmission. Finally, Constraints (3.15) and (3.16) define the domain of the decision variables.

3.2.1.3 General model

The general model is based on the previous ones and combines them as follows:

$$\min \quad T \quad (3.17)$$

$$\text{s. t} \quad K_i + \sum_{j \in N(i)} \sum_{t=1}^{T_{max}} x_{ji}^{t} = 1 \qquad \forall i \in V \quad (3.18)$$

$$\sum_{j \in N(i)} x_{ij}^{t} = 0 \qquad \forall i \in V_0, \forall t \in [0, W_v(i)) \quad (3.19)$$

$$\sum_{j \in N(i)} x_{ij}^{t} \leq 1 \qquad \forall i \in V, \forall t \in [0, T_{max}] \quad (3.20)$$

$$x_{ij}^{t} \leq K_i + \sum_{k \in N(i) \setminus \{j\}} \sum_{\tau=0}^{t-W_e((k,i))-W_v(i)} x_{ki}^{\tau} \qquad \forall (i,j) \in E, \forall t \in [0, T_{max}] \quad (3.21)$$

$$\sum_{t=0}^{T_{max}} (t + W_e((i,j)) + W_v((j))) \cdot x_{ij}^{t} \leq T \qquad \forall (i,j) \in E \quad (3.22)$$

$$x_{ik}^{\tau} \leq 1 - x_{ij}^{t} \qquad \forall (i,j) \in E, \forall t \in [0, T_{max}], \forall k \in N(i), \forall \tau \in [t+1, t+W_e((i,j))] \quad (3.23)$$

$$T \in \mathbb{N} \quad (3.24)$$

$$x_{ij}^{t} \in \mathbb{B} \qquad \forall (i,j) \in E, \forall t \in [0, T_{max}] \quad (3.25)$$

The objective function (3.17) minimizes the weighted broadcast time. Constraints (3.18) limit each vertex to receive only one message or start with its, where $T_{max} = \sum_{e \in E} W_e(e) + \sum_{v \in V} W_v(v)$ is the trivial bound. Constraints (3.19) imply that each source sends the message after its setup phase. Constraints (3.20) require that each vertex sends at most one message to a neighbor at each time t. Constraints (3.21) establish that each vertex can only transmit if it has already received the message. Constraints (3.22) state that the value of T must be greater than or equal to the time of any transmission. Constraints (3.23) mean each vertex wait $W_e(i,j)$ units of time for a new transmission. Finally, Constraints (3.24) and (3.25) define the domain of the decision variables.

For the remainder of this work, we will consider only the general model as the WMBT.

3.2.2 Lower bound algorithm for the WMBT

We have already commented on the importance of bounds in Section 2.2.2. Harutyunyan and Kamali (2008) proposed a lower bound given by $d + (d+1) \cdot w_{min}$, where d is the maximum distance of sources to other vertices and w_{min} is $\min_{v \in V}(W_v(v))$. To the best of the author's knowledge, there are no works about bounds for weighted-edge and weighted-edge-and-vertex models.

Clearly, $\sum_{e \in E} W_e(e) + \sum_{v \in V} W_v(v)$ is an upper bound for the optimal broadcast time. However, this upper bound value is not tight, because, this sum can be a very high value. Consequently, for exact models, a large upper bound makes the solver require a lot of memory to solve the problem exactly.

Algorithm 7 calculates an upper bound for the WMBT. Its main objective is to find the maximum shortest path between a target vertex and its nearest source. Note that a straightforward implementation of `LBB-Dijkstra`, i.e., applying a Dijkstra's algorithm on each $v \in V_0$, would require $O(|V_0| \cdot (|V| + |E| \cdot \log(|V|)))$. But, the worst-case running time of `LBB-Dijkstra` is $O(|V| + |E| \cdot \log(|V|)$.

Theorem 10. *Algorithm 7 returns a lower bound for the optimum of the MBT with input graph $G = (V,E)$, positive edge weights W_e, non-negative vertex weights W_v and source set V_0.*

Proof. Let b^* be the optimum an instance $G(V,E,W_e,W_v)$, V_0 and z be the value returned by the `LBB-Dijkstra` algorithm. Consider that $v \in V_0$ is the closest vertex of V_0 to a vertex $\ell \in V$, such that the distance between them is exactly z. We need at least z distance to reach ℓ from any vertex in V_0. Therefore, $z \leq b^*$, i.e., z is a lower bound for the optimum MBT value. □

3.2.3 Exact greedy algorithm for the WMBT for forest

In this section, we extend the algorithm proposed by Su et al. (2016). The original algorithm solves the weighted-vertex MBT for trees. In our extension, both vertices and edges have

Algorithm 7: Lower bound algorithm for WMBT.

Input : Undirected graph: $G = (V, E)$,
Positive edge weights: W_e,
Non-negative vertex weights: W_v,
Source set: V_0
Output: Lower bound to broadcast: $lowerBound$

```
1  LBB-Dijkstra(G, W_v, W_e, V_0)
2      for v ∈ V do
3          dist[v] ← ∞
4      S ← ∅                                              // let S be an set
5      for v ∈ V_0 do
6          dist[v] ← [0] + W_v[v]
7          S ← S ∪ {v}
8      while S ≠ ∅ do
9          v ← argmin_{u∈S} (dist[u])                     // Get the vertex with lower distance
10         S ← S \ {v}
11         for all (v,u) ∈ E do
12             if dist[u] > dist[v] + [W_e[(v,u)]] + W_v[u] then
13                 dist[u] ← dist[v] + [W_e[(v,u)]] + W_v[u]
14                 S ← S ∪ {u}
15     lowerBound ← max_{v∈V} (dist[v])                   // Get higher distance
16     return lowerBound
```

weights. Our contribution is illustrated in Algorithm 8 with an underline. For the remainder of this work, we will label Algorithm 8 as `Weighted-SCHA`.

3.2.4 Greedy algorithm for the WMBT for general instances

In this section, we propose an extension for the algorithm of Harutyunyan and Kamali (2008) (Algorithm 6). We improve this algorithm by considering multi-source graphs and also by allowing weights on both vertices and edges. The differences from the original algorithm are highlighted with rectangles in Algorithm 9. Note that our algorithm refines the greedy solution by applying the `Weighted-SCHA` algorithm (line 17). Moreover, our algorithm ensures the optimal solution if the input graph is a tree. Overall, the worst-case running time of the algorithm remains $O(|E| \cdot |V|)$.

3.2.5 Reducing rules

In this section, we propose a reducing rule to improve the performance of heuristics for WMBT. The idea is to reduce the search space by removing edges that do not belong to any optimal solution. Consequently, the probability of finding an optimal solution increases. The proposed

Algorithm 8: WMBT for forest instances.

Input : Weighted forest: $F = (V, E, S, W_v, W_e)$
Output: Total step time to broadcast: $broadcastTime$

```
1  WMBT-Forest(F)
2      broadcastTime ← 0
3      for each v ∈ V do
4          vstd[v] ← false
5      for each r ∈ R do
6          T_r ← GetTree(F,r)
7          broadcastTime ← max(broadcastTime, WMBT-Tree(T_r, r, vstd))
8      return broadcastTime
9  WMBT-Tree(T, v, vstd)
10     vstd[v] ← true
11     broadcastTime ← 0
12     time ← 0
13     PQ ← InitPQ()                                  // let PQ be an priority queue
14     for each u ∈ N(v) do
15         if vstd[u] = false then
16             PQ ←PushPQ(PQ,(WMBT-Tree(T,u,vstd) + W_e((v,u)), W_e((v,u)), u))
17     while not IsEmpty(PQ) do
18         p, w, u ← PopPQ(PQ)
19         broadcastTime ← max(broadcastTime, p + time)
20         time ← time + w
21     broadcastTime ← broadcastTime + W_v(v)
22     return broadcastTime
```

reduce algorithm uses Algorithm 7 to compute WMBT lowers bounds for each vertex. The main idea of the algorithm is to check if the lower bound of a vertex plus its transmission cost is greater than the best WMBT found. If so, we can delete some of this edges.

Definition 11. *Given an instance $G(V, E, S, W_e, W_v)$, for a vertex $v \in V$, there is a lower bound on the time required to v receive any message from S, denoted by $lb[v]$, which is computed by the* `LBB-Dijkstra` *algorithm.*

Definition 12. *Let $e = (v, u) \in E$ an edge with weight $w_e = W_e(e)$ for an instance $G(V, E, S, W_e, W_v)$. The cost of edge $cost(e)$ is computed as* $\min(lb[v] + W_v[u], lb[u] + W_v[v]) + w_e$.

Theorem 13. *Let b be the WMBT of a primal solution for an instance $G(V, E, S, W_e, W_v)$ and $e \in E$ an edge of G. If $cost(e) > b$, then not exists an optimal solution B, which has a lower WMBT than the b such that e belong to B.*

Proof. Let B' be a solution of $G(V, E, S, W_e, W_v)$ with WMBT $b' \leq b$ such that $e \in S'$. The edge e can be used in the broadcast to transmit from u to v or from v to u. The vertex v can receive the message (sent by u) at time $lb[u] + W_v[v] + w_e$. Similarly, the vertex u can receive the message

Algorithm 9: Exact greedy algorithm for WMBT

Input : Weighted Undirected graph: $G = (V, E, W_v, W_e)$,
Source set: V_0
Output: Broadcast time: $broadcastTime$

```
1  ImprovedGreedyAlgorithm(G, V_0)
2      for v ∈ V do
3          cost[v] ← ∞
4      I ← ∅                                              // let I be a set
5      for v_0 ∈ V_0 do
6          cost[v_0] ← W_v[v_0]
7          I ← I ∪ {v_0}
8      E_b ← ∅                                            // let E_b be a set
9      U ← V \ {v_0}                                      // let U be a set
10     while U ≠ ∅ do
11         v_α, v_β ← argmin_{i∈I, j∈U, (i,j)∈E} (cost[v_i] + W_v[v_j] + W_e[(v_i, v_j)])
12         cost[v_β] ← cost[v_α] + W_v[v_β] + W_e[(v_i, v_j)]
13         cost[v_α] ← cost[v_α] + W_e[(v_i, v_j)]
14         I ← I ∪ {v_β}
15         U ← U \ {v_β}
16         E_b ← E_b ∪ {(v_α, v_β)}
17     broadcastTime ← WSCHA((V, E_b, V_0, W_v, W_e))
18     return broadcastTime
```

Algorithm 10: Algorithm to remove edges of graph

Input : Weighted graph: $G = (V, E, S, W_v, W_e)$, Lower bounds: lbs,
Broadcast time: $broadcastTime$
Output: Weighted graph: G_c

```
1  ReduceRules(G, lbs, broadcastTime)
2      E_c ← ∅
3      W_{e_c} ← ∅
4      for each e ∈ E do
5          cost ← min(lbs[e_v] + W_v[e_u], lbs[e_u] + W_v[e_v]) + e_w
6          if cost ≤ broadcastTime then
7              E_c ← E_c ∪ {e}
8              W_{e_c}[e] ← W_e[e]
9      G_c = (V, E_c, S, W_v, W_e)
10     return G_c
```

(sent by v) at time $lb[v] + W_v[u] + w_e$. Therefore, S' (and any solution that uses the edge e) must have a WMBT of at least $\min(lb[v] + W_v[u], lb[u] + W_v[v]) + w_e = cost(e)$, contradicting our assumption that $b' \leq b$.

□

Rules 14. *Given instance $G(V,E,S,W_e,W_v)$, a solution S of G with b an edge e such that $cost(e) > b$, then delete e from G.*

3.2.6 BRKGA for the WMBT

In this section, we propose two `BRKGA` decoders for the WMBT: (i) Dijkstra-based with Weighted-SCHA, (ii) Minimum Spanning Forest-based with `Weighted-SCHA` refinement. Both decoders start by generating a forest to be used as input for Weighted-SCHA. We do not apply the same ideas in Chapter 2 since the decoders of MBT can not represent all solutions of WMBT in the search space. Figure 3.2 shows an example of an instance such that the FRFS decoder can not construct the optimal solution. In this example, any subgraph generated by the FRFS decoder will have the edge (g_1,d_2), but this edge does not belong to the optima.

In preliminary tests, we realize that an initial population with a single greedy solution produced good results. Thus, Algorithm 11 starts by computing the greedy solution using Algorithm 9 and the lower bound for each vertex in V. After that, in the multi-start loop (lines 2–3), it is applied the ReduceRule and population are initialized (including the greedy solution). In the main loop (lines 4–15), we evolve the population and evaluate it using a decoder method (line 10), which will be presented in the following sections. Finally, we finish the main loop after `maxIt` attempts without improvement, and then we reset the population (lines 11–14).

Algorithm 11: BRKGA for the WMBT.

Input : Weighted graph: $G = (V,E,W_v,W_e)$,
Source set: V_0, Maximum number of generations without improvement: $maxIt$
Output: Schedule Broadcast: S

```
1  BRKGA(G, V_0, maxIt)
2      S* ← ImprovedGreedyAlgorithm((V, E, V_0, W_v, W_e))
3      Lb ← LB(G, V_0)
4      while stopping criterion not met do
5          G ← {ReduceRules(G, Lb, Fitness(S*))}
6          P ← {EncodeSolution(S*)}
7          P ← P ∪ InitPopulation(G, V_0)  // Inicialize the first BRKGA population
8          noImprovement ← 0
9          while stopping criterion not met or does not improve the solution do
10             P ← BRKGA_Evolve(P)                          // Evolve population
11             noImprovement ← noImprovement + 1
12             if noImprovement = maxIt then          // If no improvement after maxIt
                generations
13                 noImprovement ← 0
14                 P ← ResetPopulation(P)              // Reset the BRKGA population
15         S* ← GetBestIndividual(P)                 // Get the best individual of the
            population
16     return S*
```

3.2.6.1 Dijkstra-based decoder (DJ)

The Dijkstra-based decoder describes a solution by a set of $|E|$ random keys (allele). Each random key represents the priority of an edge to be added on the broadcast forest, i.e., the edge with the lowest allele has the highest priority, and so on. We use the forest of Dijkstra as input to `Weighted-SCHA` for computing the WMBT. Overall, the worst-case running time of the decoder function is $O(|E| \cdot \log |V|)$.

Algorithm 12: DJ decoder

Input : Weighted graph: $G = (V, E, S, W_v, W_e)$,
Chromosomes vector: Cr
Output: Total step time to broadcast: $broadcastTime$

```
1  DJ(G, Cr)
2      for v ∈ V do
3          cost[v] ← ∞
4      I ← S                                          // let I be a set
5      E_b ← ∅                                        // let E_b be a set
6      for v ∈ V do
7          cost[v] ← ∞
8      U ← V \ S                                      // let U be a set
9      cost[v_0] ← 0
10     while U ≠ ∅ do
11         v_α, v_β ← argmin_{i∈I, j∈U, (i,j)∈E} (cost[v_i] + Cr[(v_i, v_j)])
12         I ← I ∪ {v_β}
13         U ← U \ {v_β}
14         E_b ← E_b ∪ {(v_α, v_β)}
15     broadcastTime ← WSCHA((V, E_b, S, W_v, W_e))
16     return broadcastTime
```

3.2.6.2 Minimum Spanning Forest-based decoder (MSF)

In this section, we describe a Minimum Spanning Forest-based decoder (MSF). In this approach, each random key is associated with an edge. Each allele indicates the weight of this edge. Next, we compute the minimum spanning forest using Kruskal Algorithm (Kruskal, 1956), . Finally, we compute the WMBT over this forest using the `Weighted-SCHA`. Overall, the worst-case running time of the decoder function is $O(|E| \cdot \log |E|)$.

3.3 Computacional results of the WMBT

This section presents the computational experiments conducted to evaluate the effectiveness of our proposed ILP and BRKGAs. The proposed algorithms are compared with the following state-of-the-art approaches: (i) an adaptation of Ant Colony metaheuristic (ACS) from Hasson and

Algorithm 13: MSF decoder

Input : Weighted graph: $G = (V, E, S, W_v, W_e)$,
Chromosomes vector: Cr
Output: Total step time to broadcast: $broadcastTime$

```
1  Root(v, parent)
2      if parent[v] < 0 then
3          return v
4      parent[v] ← Root(parent[v], parent)
5      return parent[v]
6  Merge(u, v, parent)
7      u ← Root(u, parent)
8      v ← Root(v, parent)
9      parent[v] ← parent[u]
10 MSF(G, Cr)
11     E_b ← ∅
12     for each v ∈ V do
13         parent[v] ← −1
14     r ← ∀v_0 ∈ S
15     for each v_0 ∈ V_0 do
16         Merge(r, v_0)
17     Ranking ← ∅
18     for each e ∈ E do
19         Ranking ← Ranking ∪ {(Cr[e], e_v, e_u)}
20     while |E_b| + |S| ≠ |V| do
21         e ←ExtractMin(Ranking)
22         Ranking ← Ranking \ {e}
23         if Root(e_v, parent) ≠ Root(e_u, parent) then
24             E_b ← E_b ∪ {(e_v, e_u)}
25             Merge(e_v, e_u)
26     broadcastTime ← WSCHA((V, E_b, S, W_v, W_e))
27     return broadcastTime
```

Sipper (2004); and (ii) an adaption of constructive heuristic from Harutyunyan and Kamali (2008). For the ACS algorithm, we use the Weighted-SCHA algorithm to refine the partial solutions. We adapt the constructive heuristic from Harutyunyan and Kamali (2008) by adding weights in the edges.

All experiments in this section were conducted on an Intel Core i7-6700 with 3.40 GHz, 32 GB of RAM, running Ubuntu 18.04.5. The heuristic algorithms were coded in C++ and compiled with g++ 7.5 and '-O3' flag. The BRKGA C++ framework developed by Toso and Resende (2015) has been used to implement our BRKGA. Moreover, IBM Cplex 12.9 has been adopted to solve the ILP models.

3.3.1 Instances

We tested our algorithms on a total of 40 instances, which include:

- **Random graph** (20 instances): The random graph G_r is based on the $\mathbb{G}(n,p)$ model, also known as binomial model (Gilbert, 1959). Each graph $G_r = (V, E_r)$ is generated with n vertices and each potential edge in E_r is created with probability p. The vertices and edges weights have been created using a uniform distribution.

- **Large synthetic instances based on random tree** (20 instances): Given that the optimal solution for large the WMBT instances is often unknown, we generated a new benchmark with known optimal solutions using the following procedure.

Algorithm 14 creates a random instance $G = (V, E, W_v, W_e)$ by applying the union of a weighted-vertex-and-edge random tree $T = (V_T, E_T, W_{v_T}, W_{e_T})$ and a weighted-edge random graph $G_r = (V_r, E_r, W_r)$, where $V = V_T = V_r$, $E = E_T \cup E_r$, $W_v = W_{v_T}$ and $W_e = W_{e_T} \cup W_r$. First, we calculate the optimal broadcast time of T by using the Algorithm 8 (WMBT-TREE). After that, we created a random graph G_r to merge with T into G. Finally, for each new edge in G_r, we assigned a weight value preserving the optimal solution found previously.

Appendix A.2 gives more information regarding the adopted instances, such as the number of vertices and edges.

3.3.2 Parameter settings and experimental protocol

In our experiments with the ACS algorithm, we adopted the same parameter settings indicated by its authors. Moreover, we have used the irace tunning tool (López-Ibáñez et al., 2016) to configure the parameters of our algorithms and their variations. The best parameter settings identified by the tuning experiment are reported in Table 3.2. In this table, BRKGA parameters p, p_e, p_m, ρ_e, and K represent, respectively, the number of individuals in each population, percentage of elite individuals into each population, percentage of mutants introduced at each generation into the population, the probability that an offspring inherits the allele of its elite parent, and the number of independent populations.

Table 3.2: Range considered by IRACE and best parameter settings obtained.

Parameter	Value ranges	BRKGA-DJ	BRKGA-MSF
p	-	$\|V\|$	$\|V\|$
p_e	0.10, 0.11, ..., 0.25	0.24	0.25
p_m	0.10, 0.11, ..., 0.30	0.23	0.22
ρ_e	0.50, 0.51, ..., 0.80	0.59	0.52
K	-	1	1

We have set a time limit of 3600 s (1 h) to Cplex for solving our ILP model. To assess the

Algorithm 14: Create a random weighted graph with optima known.

Input : Number of vertices: n,
density: d
Output: Weighted-vertex-and-edge graph: $G(V_G, E_G, S, W_v, W_e)$,
minimum broadcast time of graph G: $wmbt$

```
1  GenerateWeightedGraph(n,d)
2      T ← RandomTree(n)
3      root ← Random(0,n)
4      for each v ∈ V(T) do
5          W_vT[v] ← Random(0,∞)
6      for each e ∈ E(T) do
7          W_eT[e] ← Random(1,∞)
8      S ← WMBT-Tree((V(T),V(E),W_vT,W_eT), root)
9      times ← GetTimes(S)
10     G_r ← RandomGraph(n,d)
11     V_G ← V(T)
12     W_v ← W_vT
13     E_G ← E(T)
14     W_e ← W_eT
15     for each e ∈ E(G_r) \ E(T) do
16         E_G ← E_G ∪ {e}
17         differentTime ← abs(times[e.v] − times[e.u])
18         W_e[e] ← differentTime + Random(1,∞)
19     wmbt ← max_{v∈V} (times[v])
20     G ← (V_G, E_G, {root}, W_v, W_e)
21     return G, wmbt
```

average performance of the heuristic algorithms (BRGKAs and ACS), we have performed 10 runs in each benchmark instance with different random seeds for each run. A time limit of 300 s has been used for these runs.

3.3.3 Comparing the deterministic algorithms

Table 3.3 compares the algorithm of Harutyunyan and Kamali (2008) (Algorithm 6), our improvement in this algorithm (Algorithm 9), and the ILP model in all instances. The methods are compared using only the following criteria: the WMBT, and the CPU time to find the best solution (column 't (s)'). An asterisk means that the method has been able to prove the optimality of a given result. Since the algorithms are deterministic, we run them only one time.

Note that `Harutyunyan-WSCHA` attains or improves the quality of the solutions obtained by `Harutyunyan` on all instances. In the ILP model, we set the lower bound using Algorithm 7. We also used the WMBT value of `Harutyunyan-WSCHA` as the upper bound to reduce the computational demand and the total amount of used memory. The ILP model does not produce feasible solutions for instances with more than 512 vertices. But in graphs with up to 256 vertices, the

model showed a significant result.

Table 3.3: Comparative results of deterministic algorithms.

Instance	Lower Bound	HARUTYUNYAN		HARUTYUNYAN-WSCHA		ILP	
		WMBT	t (s)	WMBT	t (s)	WMBT	t (s)
R1024-G1-D10-1	13	20	0.19	**19**	0.19	-	3600.00
R1024-G1-D15-1	15	22	0.30	**21**	0.30	-	3600.00
R1024-G1-D20-1	9	17	0.40	**16**	0.39	-	3600.00
R1024-G1-D25-1	9	17	0.48	**16**	0.47	-	3600.00
R128-G1-D10-1	20	29	< 0.01	27	< 0.01	**23**	3575.57
R128-G1-D15-1	20	28	< 0.01	27	< 0.01	**23**	3603.59
R128-G1-D20-1	18	26	< 0.01	25	< 0.01	**24**	3605.44
R128-G1-D25-1	15	22	< 0.01	20	< 0.01	**18**	3606.34
R256-G1-D10-1	15	24	< 0.01	**23**	< 0.01	-	3600.00
R256-G1-D15-1	16	22	< 0.01	**21**	< 0.01	-	3600.00
R256-G1-D20-1	10	18	0.01	**16**	0.01	-	3600.00
R256-G1-D25-1	12	17	0.01	**16**	0.01	-	3600.00
R512-G1-D10-1	13	**21**	0.03	**21**	0.02	-	3600.00
R512-G1-D15-1	16	23	0.04	**22**	0.04	-	3600.00
R512-G1-D20-1	10	18	0.05	**17**	0.05	-	3600.00
R512-G1-D25-1	13	19	0.06	**18**	0.06	-	3600.00
R64-G1-D10-1	22	28	< 0.01	27	< 0.01	**23**	36.45
R64-G1-D15-1	20	25	< 0.01	24	< 0.01	**22**	42.99
R64-G1-D20-1	20	30	< 0.01	29	< 0.01	**25**	3600.64
R64-G1-D25-1	16	21	< 0.01	21	< 0.01	**18**	51.16
RO1024-G1-D10-1	440	492	0.19	**490**	0.19	-	3600.00
RO1024-G1-D15-1	550	609	0.30	**603**	0.30	-	3600.00
RO1024-G1-D20-1	479	536	0.40	**532**	0.39	-	3600.00
RO1024-G1-D25-1	353	**393**	0.48	**393**	0.47	-	3600.00
RO128-G1-D10-1	117	139	< 0.01	**124**	< 0.01	-	3600.00
RO128-G1-D15-1	186	214	< 0.01	**186**	< 0.01	-	3600.00
RO128-G1-D20-1	167	202	< 0.01	**197**	< 0.01	-	3600.00
RO128-G1-D25-1	122	**149**	< 0.01	**149**	< 0.01	-	3600.00
RO256-G1-D10-1	250	282	< 0.01	**279**	< 0.01	-	3600.00
RO256-G1-D15-1	208	239	< 0.01	**237**	< 0.01	-	3600.00
RO256-G1-D20-1	169	202	< 0.01	**197**	0.01	-	3600.00
RO256-G1-D25-1	211	243	0.01	**241**	0.01	-	3600.00
RO512-G1-D10-1	288	335	0.03	**331**	0.02	-	3600.00
RO512-G1-D15-1	409	479	0.04	**476**	0.04	-	3600.00
RO512-G1-D20-1	297	333	0.05	**332**	0.05	-	3600.00
RO512-G1-D25-1	320	362	0.06	**356**	0.06	-	3600.00
RO64-G1-D10-1	74	95	< 0.01	**74***	< 0.01	**74***	15.23
RO64-G1-D15-1	108	128	< 0.01	128	< 0.01	**109**	52.02
RO64-G1-D20-1	87	99	< 0.01	**88**	< 0.01	**88**	57.86
RO64-G1-D25-1	140	**168**	< 0.01	**168**	< 0.01	-	3600.00
# Best		4		31		11	
Avg. t (s)		0.08		0.08		3066.18	

3.3.4 Validating the reducing rules

This section shows the benefits of using the proposed reducing rules (RR). Tables 3.4, 3.5 and 3.6 compare the decoder with and without the RR. We considered instances with a density of

25%, a time limit of 600 seconds, and 5 runs with different random seeds for each run. Moreover, we behold that the RR can reduce the density by 5–95%. Consequently, the BRKGA gets faster.

Table 3.4: Comparative results of BRKGA-DJs with and without RR

Method	BRKGA-DJ without RR	BRKGA-DJ with RR
# Best	9	**10**
# Best Avg.	8	**10**
Avg. t (s)	129.66	**55.68**

Table 3.5: Comparative results of BRKGA-MSFs with and without RR

Method	BRKGA-MSF without RR	BRKGA-MSF with RR
# Best	8	**10**
# Best Avg.	8	**9**
Avg. t (s)	140.93	**113.39**

Table 3.6: Comparative results of BRKGAs with and without RR

Method	BRKGA-DJ without RR	BRKGA-MSF without RR	BRKGA-DJ with RR	BRKGA-MSF with RR
# Best	8	8	9	**10**
# Best Avg.	7	**8**	**8**	**8**
Avg. t (s)	189.65	140.93	115.68	**113.39**

3.3.5 Comparing the metaheuristics

Table 3.7 compares the heuristics `ACS-WSCHA`, `Harutyunyan-WSCHA` and `BRKGAs` in all instances for the WMBT. The results show that `BRKGAs` outperform both heuristics from the literature in solution quality and CPU time. The `BRKGA-MSF` found 38 best solutions, whereas this number was 36 for the `BRKGA-DJ`. Both decodes have similar computational times.

The BRKGAs initialize the initial population with an element based on a solution of `Harutyunyan-WSCHA`, so it is prospective that the BRKGA would be better than its.

Table 3.7: Comparative results of ACS, BRKGAs and HARUTYUNYAN-WSCHA

Instance	BRKGA-DJ			BRKGA-MSF			HARUTYUNYAN-WSCHA		ACS-WSCHA		
	Best	Avg.	t (s)	Best	Avg.	t (s)	Best	t (s)	Best	Avg.	t (s)
R1024-G1-D10-1	**19**	**19.00**	0.24	**19**	**19.00**	0.24	**19**	0.24	391	391.00	300.00
R1024-G1-D15-1	**21**	**21.00**	0.36	**21**	**21.00**	0.36	**21**	0.38	512	512.00	300.00
R1024-G1-D20-1	**16**	**16.00**	0.46	**16**	**16.00**	0.46	**16**	0.49	735	735.00	300.00
R1024-G1-D25-1	**16**	**16.00**	0.57	**16**	**16.00**	0.57	**16**	0.59	843	843.00	300.00
R128-G1-D10-1	**26**	**26.80**	256.73	**26**	**26.80**	273.74	27	< 0.01	58	58.00	300.00

Continued on next page

Table 3.7 – continued from previous page

Instance	BRKGA-DJ			BRKGA-MSF			HARUTYUNYAN-WSCHA		ACS-WSCHA		
	Best	Avg.	t (s)	Best	Avg.	t (s)	Best	t (s)	Best	Avg.	t (s)
R128-G1-D15-1	**26**	26.80	275.70	**26**	**26.70**	257.74	27	< 0.01	70	70.00	300.00
R128-G1-D20-1	**24**	**24.90**	299.05	25	25.00	300.00	25	< 0.01	93	93.00	300.00
R128-G1-D25-1	**20**	**20.00**	< 0.01	**20**	**20.00**	< 0.01	**20**	< 0.01	130	130.00	300.00
R256-G1-D10-1	**23**	**23.00**	< 0.01	**23**	**23.00**	< 0.01	**23**	< 0.01	97	97.00	300.00
R256-G1-D15-1	**21**	**21.00**	0.01	**21**	**21.00**	0.01	**21**	0.01	135	135.00	300.00
R256-G1-D20-1	**16**	**16.00**	0.01	**16**	**16.00**	0.01	**16**	0.01	158	158.00	300.00
R256-G1-D25-1	**16**	**16.00**	0.01	**16**	**16.00**	0.01	**16**	0.01	197	197.00	300.00
R512-G1-D10-1	**21**	**21.00**	0.03	**21**	**21.00**	0.03	**21**	0.03	194	194.00	300.00
R512-G1-D15-1	**22**	**22.00**	0.04	**22**	**22.00**	0.04	**22**	0.05	297	297.00	300.00
R512-G1-D20-1	**17**	**17.00**	0.06	**17**	**17.00**	0.06	**17**	0.06	340	340.00	300.00
R512-G1-D25-1	**18**	**18.00**	0.07	**18**	**18.00**	0.07	**18**	0.08	407	407.00	300.00
R64-G1-D10-1	**25**	**25.30**	146.31	**25**	25.60	201.67	27	< 0.01	38	38.00	300.00
R64-G1-D15-1	**23**	23.80	277.72	**23**	**23.70**	226.22	24	< 0.01	52	52.00	300.00
R64-G1-D20-1	28	28.00	300.00	**27**	**27.90**	278.18	29	< 0.01	72	72.00	300.00
R64-G1-D25-1	**19**	**19.40**	142.70	**19**	19.50	195.32	21	< 0.01	56	56.00	300.00
RO1024-G1-D10-1	**490**	**490.00**	0.23	**490**	**490.00**	0.23	**490**	0.24	43309	43309.00	300.00
RO1024-G1-D15-1	**603**	**603.00**	0.35	**603**	**603.00**	0.35	**603**	0.37	73373	73373.00	300.00
RO1024-G1-D20-1	**532**	**532.00**	0.45	**532**	**532.00**	0.45	**532**	0.48	106708	106708.00	300.00
RO1024-G1-D25-1	**393**	**393.00**	0.56	**393**	**393.00**	0.56	**393**	0.59	91319	91319.00	300.00
RO128-G1-D10-1	**124**	**124.00**	< 0.01	**124**	**124.00**	< 0.01	**124**	< 0.01	1567	1568.10	300.00
RO128-G1-D15-1	**186**	**186.00**	< 0.01	**186**	**186.00**	< 0.01	**186**	< 0.01	3655	3655.00	300.00
RO128-G1-D20-1	182	192.90	300.00	**167**	**191.40**	287.96	197	< 0.01	3829	3829.00	300.00
RO128-G1-D25-1	**122**	140.90	245.69	**122**	**138.70**	268.81	149	< 0.01	3720	3720.00	300.00
RO256-G1-D10-1	**277**	**278.80**	271.21	**277**	**278.80**	285.83	279	< 0.01	5878	5882.30	300.00
RO256-G1-D15-1	232	235.50	300.00	**209**	**234.10**	281.18	237	0.01	8603	8603.00	300.00
RO256-G1-D20-1	197	197.00	300.00	**189**	**195.10**	277.43	197	0.01	8056	8056.00	300.00
RO256-G1-D25-1	**238**	**240.30**	249.51	241	241.00	300.00	241	0.01	10637	11548.90	300.00
RO512-G1-D10-1	**331**	**331.00**	0.03	**331**	**331.00**	0.03	**331**	0.03	14470	14640.60	300.00
RO512-G1-D15-1	**476**	**476.00**	0.04	**476**	**476.00**	0.04	**476**	0.05	31088	31088.00	300.00
RO512-G1-D20-1	**332**	**332.00**	0.06	**332**	**332.00**	0.06	**332**	0.06	30598	30598.00	300.00
RO512-G1-D25-1	**356**	**356.00**	0.07	**356**	**356.00**	0.07	**356**	0.08	36776	36776.00	300.00
RO64-G1-D10-1	**74**	**74.00**	< 0.01	**74**	**74.00**	< 0.01	**74**	< 0.01	485	485.00	300.00
RO64-G1-D15-1	**109**	**109.00**	5.81	**109**	**109.00**	65.92	128	< 0.01	914	916.00	300.00
RO64-G1-D20-1	**88**	**88.00**	< 0.01	**88**	**88.00**	< 0.01	**88**	< 0.01	1044	1044.00	300.00
RO64-G1-D25-1	**140**	142.80	71.03	**140**	**140.00**	37.46	168	< 0.01	2636	2636.00	300.00
Avg. t (s)		**86.13**			88.53			0.1		300	
# Best		36			**38**			25		0	
# Best Avg.		32			**36**			25		0	

4

Final Considerations

In this work, we proposed some algorithms for the MINIMUM BROADCAST TIME problem and its weighted version (WMBT). For the MBT, we have described a new lower bound procedure for the problem based on vertex distances in the input graph. This procedure can reduce computational resources and heuristic algorithms used to prove the optimality. The experimental results reveal that our approach increased several lower bounds and helped the Integer Linear Programming (ILP) model and our BRKGA prove several previously unknown optima.

We proposed two BRKGA metaheuristics and a matheuristic using BRKGA and ILP, our approaches outperformed ACS (Hasson and Sipper, 2004) and ILP (de Sousa et al., 2018) in both solution quality and CPU time. In our best approaches, we used the idea described in Koh and Tcha (1991); Su et al. (2010) to improve solution quality. For all instances for the MBT with known optima value, our approaches either attained the optimal value or missed it by at most one broadcast step. Moreover, we described a method to create instances with known optima. To the best of our knowledge, this is the first time that major efforts were made to study, generate, and solve hard instances for the MBT.

The WMBT is even harder to handle than the MBT because the broadcast scheme does not follow a standard pattern. There are a smaller amount of articles. To the best of our knowledge, we proposed the first mathematical model for the WMBT. Thus, we formulated three ILP models (one for each variant of the WMBT). Moreover, we adapted our lower bound of MBT algorithm to WMBT. We adopted a similar refinement of MBT to improve solution quality, which the experimental computational showed a significant improvement. We proposed two BRKGA metaheuristics, which outperformed the algorithms in the literature. Finally, we proposed a reducing rule.

4.1 Future works

We also plan to extend the proposed algorithm to solve related problems, such as the following MBT or WMBT:

(i) with at most k transmissions (k-broadcast) (Lazard, 1992; Harutyunyan and Liestman, 2001),

(ii) proposal an algorithm distributed for problems, and

(iii) apply our algorithms for solving real problems, such as swarm robotics (Al-Sarawi et al., 2017).

4.2 Scientific production

This master's degree resulted in the following scientific production:

(i) An article published in the *Simpósio Brasileiro de Pesquisa Operacional* (SBPO) (Lima et al., 2020) composed by part of Chapter 2;

(ii) An article in revision in the International Transactions in Operational Research (ITOR) composed by Chapter 2.

(iii) An article published in IEEE Systems, Man, and Cybernetics Society (SMC) (Lima et al., 2021), in cooperation with others researches;

Finally, we are writing a new article composed by the results of Chapter 3.

References

S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi. Internet of things (iot) communication protocols: Review. In *2017 8th International Conference on Information Technology (ICIT)*, pages 685–690, May 2017. **DOI** 10.1109/ICITECH.2017.8079928.

A Averbuch, Y Roditty, and B Shoham. Computation of broadcasting multiple messages in a positive weighted tree. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 35:161–184, 2000.

Amotz Bar-Noy and Shlomo Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. *Mathematical systems theory*, 27(5):431–452, 1994.

Amotz Bar-Noy, Sudipto Guha, Joseph Naor, and Baruch Schieber. Message multicasting in heterogeneous networks. *SIAM Journal on Computing*, 30(2):347–358, 2000.

Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998. **DOI** 10.1287/opre.46.3.316.

Alex Bavelas. Communication patterns in task-oriented groups. *The journal of the acoustical society of America*, 22(6):725–730, 1950.

James C. Bean. Genetic algorithms and random keys for sequencing and optimization. *INFORMS Journal on Computing*, 6(2):154–160, 1994. URL https://EconPapers.repec.org/RePEc:inm:orijoc:v:6:y:1994:i:2:p:154-160.

Marco A. Boschetti, Vittorio Maniezzo, Matteo Roffilli, and Antonio Bolufé Röhler. Matheuristics: Optimization, simulation and control. In María J. Blesa, Christian Blum, Luca Di Gaspero, Andrea Roli, Michael Sampels, and Andrea Schaerf, editors, *Hybrid Metaheuristics*, pages 171–177, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04918-7.

Dan Bucantanschi, Blaine Hoffmann, Kevin R Hutson, and R Matthew Kretchmar. A neighborhood search technique for the freeze tag problem. In *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, pages 97–113. Springer, 2007.

Raquel Silva Cabral, Andre Aquino, Alejandro Frery, Osvaldo Rosso, and Jaime Ramírez. Structural changes in data communication in wireless sensor networks. *Open Physics*, 11 (12):1645–1652, 2013.

Xiaogeng Chu and Yuning Chen. Time division inter-satellite link topology generation problem: Modeling and solution. *International Journal of Satellite Communications and Networking*, 36 (2):194–206, 2018.

Amaro de Sousa, Gabriela Gallo, Santiago Gutierrez, Franco Robledo, Pablo Rodríguez-Bocca, and Pablo Romero. Heuristics for the minimum broadcast time. *Electronic Notes in Discrete Mathematics*, 69:165–172, aug 2018. **DOI** 10.1016/j.endm.2018.07.022. URL https://doi.org/10.1016%2Fj.endm.2018.07.022.

Anthony Dekker. Applying social network analysis concepts to military c4isr architectures. *Connections*, 24(3):93–103, 2002.

Michael Elkin and Guy Kortsarz. Sublogarithmic approximation for telephone multicast: Path out of jungle. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 76–85, 01 2003. **DOI** 10.1145/644108.644120.

Arthur Farley, Stephen Hedetniemi, Sandra Mitchell, and Andrzej Proskurowski. Minimum broadcast graphs. *Discrete Mathematics*, 25(2):189 – 193, 1979. ISSN 0012-365X. **DOI** https://doi.org/10.1016/0012-365X(79)90022-0. URL http://www.sciencedirect.com/science/article/pii/0012365X79900220.

Cristopher G. S. Freitas, Andre L. L. Aquino, Heitor S. Ramos, Alejandro C. Frery, and Osvaldo A. Rosso. A detailed characterization of complex networks using information theory. *Scientific Reports*, 9(1):16689, 2019.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979. ISBN 0716710447.

Michel Gendreau and Jean-Yves Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010.

E. N. Gilbert. Random graphs. *Ann. Math. Statist.*, 30(4):1141–1144, 12 1959. **DOI** 10.1214/aoms/1177706098. URL https://doi.org/10.1214/aoms/1177706098.

Alan G Gomez, William C Oakes, and Les L Leone. *Engineering your future: A project-based introduction to engineering*. Great Lakes Press, 2004.

José Fernando Gonçalves, Mauricio GC Resende, and Rodrigo F Toso. An experimental comparison of biased and unbiased random-key genetic algorithms. *Pesquisa Operacional*, 34(2):143–164, 2014.

José Gonçalves and Mauricio Resende. Biased random-key genetic algorithms for combinatorial optimization. *J. Heuristics*, 17:487–525, 10 2011. **DOI** 10.1007/s10732-010-9143-1.

Daniel L Guidoni, Raquel AF Mini, and Antonio AF Loureiro. On the design of resilient heterogeneous wireless sensor networks based on small world concepts. *Computer Networks*, 54(8):1266–1281, 2010.

H. A. Harutyunyan and S. Kamali. Efficient broadcasting in networks with weighted nodes. In *2008 14th IEEE International Conference on Parallel and Distributed Systems*, pages 879–884, 2008.

Hovhannes A Harutyunyan and Cosmin Jimborean. New heuristic for message broadcasting in networks. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 517–524. IEEE, 2014.

Hovhannes A. Harutyunyan and Arthur L. Liestman. Improved upper and lower bounds for k-broadcasting. *Networks*, 37(2):94–101, 2001. **DOI** 10.1002/1097-0037(200103)37:2<94::AID-NET4>3.0.CO;2-6. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/1097-0037%28200103%2937%3A2%3C94%3A%3AAID-NET4%3E3.0.CO%3B2-6.

Hovhannes A. Harutyunyan and Wei Wang. Broadcasting algorithm via shortest paths. In *2010 IEEE 16th International Conference on Parallel and Distributed Systems*, pages 299–305, 2010. **DOI** 10.1109/ICPADS.2010.110.

Yehudit Hasson and Moshe Sipper. A novel ant algorithm for solving the minimum broadcast time problem. In Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 501–510, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30217-9.

Sandra M. Hedetniemi, Stephen T. Hedetniemi, and Arthur L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988. **DOI** 10.1002/net.3230180406. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230180406.

Cory J. Hoelting, Dale A. Schoenefeld, and Roger L. Wainwright. A genetic algorithm for the minimum broadcast time problem using a global precedence vector. In *Proceedings of the 1996 ACM Symposium on Applied Computing*, SAC '96, page 258–262, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897918207. **DOI** 10.1145/331119.331187. URL https://doi.org/10.1145/331119.331187.

John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI, 1975.

Marika Ivanova. Optimization problems in communication networks and multi-agent path finding. *Bergen Open Research Archive*, 2019.

H. Keshavarz, A. Bagheri, K. Layeghi, and Seyed Iman Mahdavi. A simulated annealing approach for the freeze-tag problem. In *2011 International Conference on Recent Trends in Information Systems*, pages 94–98, 2011. **DOI** 10.1109/ReTIS.2011.6146847.

J. Koh and D. Tcha. Information dissemination in trees with nonuniform edge transmission times. *IEEE Transactions on Computers*, 40(10):1174–1177, 1991. **DOI** 10.1109/12.93751.

J-C König and Emmanuel Lazard. Minimum k-broadcast graphs. *Discrete Applied Mathematics*, 53(1-3):199–209, 1994.

Guy Kortsarz and David Peleg. Approximation algorithms for minimum time broadcast. In D. Dolev, Z. Galil, and M. Rodeh, editors, *Theory of Computing and Systems*, pages 67–78, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. ISBN 978-3-540-47214-8.

J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. the American Mathematical Soc.*, 7(1):48–48, 1956. **DOI** 10.1090/s0002-9939-1956-0078686-7.

E. Lazard. Broadcasting in dma-bound bounded degree graphs. *Discrete Applied Mathematics*, 37-38:387 – 400, 1992. ISSN 0166-218X. **DOI** https://doi.org/10.1016/0166-218X(92)90147-3. URL http://www.sciencedirect.com/science/article/pii/0166218X92901473.

A. Lima, M. Santos, A. Lima, B. Nogueira, and R. G. S. Pinheiro. A multi-population brkga for the automatic clustering problem. In *2021 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 1–6. IEEE, 2021.

Alfredo Lima, Rian Pinheiro, Bruno Nogueira, and Rodrigo Peixoto. Algoritmo genético de chaves aleatórias viciadas para o problema do tempo de transmissão mínimo, Oct 2020. URL https://proceedings.science/sbpo-2020/papers/algoritmo-genetico-de-chaves-aleatorias-viciadas-para-o-problema-do-tempo-de-tran

Shawna D Lockhart and Cindy M Johnson. *Engineering design communication: conveying design through graphics*. Addison-Wesley, 2000.

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration.

Operations Research Perspectives, 3:43–58, 2016. ISSN 2214-7160. **DOI** https://doi.org/10.1016/j.orp.2016.09.002. URL http://www.sciencedirect.com/science/article/pii/S2214716015300270.

M. Padberg and G. Rinaldi. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operations Research Letters*, 6(1):1–7, 1987. ISSN 0167-6377. **DOI** 10.1016/0167-6377(87)90002-2. URL http://www.sciencedirect.com/science/article/pii/0167637787900022.

Juan A Rico-Gallego, Juan C Díaz-Martín, Ravi Reddy Manumachu, and Alexey L Lastovetsky. A survey of communication performance models for high-performance computing. *ACM Computing Surveys (CSUR)*, 51(6):1–36, 2019.

Franco Robledo, Pablo Rodríguez-Bocca, and Pablo Romero. Optimal broadcast strategy in homogeneous point-to-point networks. In Giuseppe Nicosia, Varun Ojha, Emanuele La Malfa, Giorgio Jansen, Vincenzo Sciacca, Panos Pardalos, Giovanni Giuffrida, and Renato Umeton, editors, *Machine Learning, Optimization, and Data Science*, pages 448–457, Cham, 2020. Springer International Publishing. ISBN 978-3-030-64583-0.

Scheuermann and Wu. Heuristic algorithms for broadcasting in point-to-point computer networks. *IEEE Transactions on Computers*, C-33(9):804–811, Sep. 1984. ISSN 2326-3814. **DOI** 10.1109/TC.1984.1676496.

Weiping Shang, Pengjun Wan, and Xiaodong Hu. Approximation algorithms for minimum broadcast schedule problem in wireless sensor networks. *Frontiers of Mathematics in China*, 5(1):75–87, 2010.

P. J. Slater, E. J. Cockayne, and S. T. Hedetniemi. Information dissemination in trees. *SIAM Journal on Computing*, 10(4):692–701, 1981. **DOI** 10.1137/0210052. URL https://doi.org/10.1137/0210052.

Kenneth Sorensen, Marc Sevaux, and Fred Glover. A history of metaheuristics. *arXiv preprint arXiv:1704.00853*, 2017.

Villiam M. Spears and Kenneth A. De Jong. On the virtues of parameterized uniform crossover. In *In Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, 1991.

Yu-Hsuan Su, Ching-Chi Lin, and DT Lee. Broadcasting in heterogeneous tree networks. In *International Computing and Combinatorics Conference*, pages 368–377. Springer, 2010.

Yu-Hsuan Su, Ching-Chi Lin, and D.T. Lee. Broadcasting in weighted trees under the postal model. *Theoretical Computer Science*, 621:73 – 81, 2016. ISSN 0304-3975.

DOI https://doi.org/10.1016/j.tcs.2016.01.031. URL http://www.sciencedirect.com/science/article/pii/S0304397516000554.

El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009. ISBN 0470278587.

Rodrigo F Toso and Mauricio GC Resende. A c++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 30(1):81–93, 2015.

Cheng-Hsiao Tsou, Gen-Huey Chen, Hung-I Yu, and Ching-Chi Lin. The broadcast median problem in heterogeneous postal model. *Journal of Combinatorial Optimization*, 25(4): 602–616, 2013.

Vijay V Vazirani. *Approximation algorithms*, volume 1. Springer, 2001.

Iwona Wojciechowska and Frances L. Scoy. *Broadcasting in Grid Graphs*. PhD thesis, West Virginia University, USA, 1999. AAI9967246.

Appendix A

Detailed instances

A.1 Instances for the MBT.

For each instance, Table A.1 gives the name (column 'Instance'), the number of vertices (column '$|V|$'), the number of edges (column '$|E|$'), the initial source vertex (column 'V_0'), the theoretical lower bound (column 'TLB', see Eq. (2.1)), the lower bound found by the proposed Algorithm 1 (column 'LBB'), and the edge density (column 'density').

Table A.1: Description of the test instances.

Instance	$\|V\|$	$\|E\|$	V_0	TLB	LBB	density	Instance	$\|V\|$	$\|E\|$	V_0	TLB	LBB	density
Harary Graphs													
$H_{10,30}$	30	150	{30}	5	3	0.3448	$H_{11,50}$	50	275	{50}	6	3	0.2245
$H_{20,50}$	50	500	{50}	6	3	0.4082	$H_{21,50}$	50	525	{50}	6	2	0.4286
$H_{2,100}$	100	100	{100}	7	50	0.0202	$H_{2,17}$	17	17	{17}	4	8	0.125
$H_{2,30}$	30	30	{30}	5	15	0.069	$H_{2,50}$	50	50	{50}	6	25	0.0408
$H_{3,17}$	17	26	{6}	4	4	0.1912	$H_{3,30}$	30	45	{30}	5	8	0.1034
$H_{3,50}$	50	75	{50}	6	13	0.0612	$H_{5,17}$	17	43	{17}	4	3	0.3162
$H_{6,17}$	17	51	{17}	4	3	0.375	$H_{7,17}$	17	60	{17}	4	2	0.4412
$H_{8,30}$	30	120	{30}	5	4	0.2759	$H_{9,30}$	30	135	{30}	5	3	0.3103
Hypercube Graphs													
HC_5	32	80	{1}	5	5	0.1613	HC_6	64	192	{1}	6	6	0.0952
HC_7	128	448	{1}	7	7	0.0551	HC_8	256	1024	{1}	8	8	0.0314
HC_9	512	2304	{1}	9	9	0.0176	HC_{10}	1024	5120	{1}	10	10	0.0098
Cube-Connected Cycles Graphs													
CCC_3	24	36	{1}	5	6	0.1304	CCC_4	64	96	{1}	6	8	0.0476
CCC_5	160	240	{1}	8	10	0.0189	CCC_6	384	576	{1}	9	13	0.0078
CCC_7	896	1344	{1}	10	15	0.0034							
deBruijn Graphs													
DB_4	16	32	{1}	4	4	0.2583	DB_5	32	64	{1}	5	5	0.1270
DB_6	64	128	{1}	6	6	0.0630	DB_7	128	256	{1}	7	7	0.0314
DB_8	256	512	{1}	8	8	0.0157	DB_9	512	1024	{1}	9	9	0.0078
DB_{10}	1024	2048	{1}	10	10	0.0039							

Continued on next page

Table A.1 – continued from previous page

Instance	$\|V\|$	$\|E\|$	V_0	TLB	LBB	density	Instance	$\|V\|$	$\|E\|$	V_0	TLB	LBB	density
Shuffle Exchange Graphs													
SE_4	16	21	{1}	4	7	0.1750	SE_5	32	46	{1}	5	9	0.0927
SE_6	64	93	{1}	6	11	0.0461	SE_7	128	190	{1}	7	13	0.0234
SE_8	256	381	{1}	8	15	0.0117	SE_9	512	766	{1}	9	17	0.0059
SE_{10}	1024	1533	{1}	10	19	0.0029							
Network Repository													
SW-100-3-0d1-trial1	100	100	{1}	7	61	0.0202	SW-100-3-0d2-trial1	100	100	{1}	7	31	0.0202
SW-100-3-0d2-trial3	100	100	{1}	7	31	0.0202	SW-100-4-0d1-trial1	100	200	{1}	7	7	0.0404
SW-100-4-0d1-trial2	100	200	{1}	7	7	0.0404	SW-100-4-0d1-trial3	100	200	{1}	7	9	0.0404
SW-100-4-0d2-trial1	100	200	{1}	7	7	0.0404	SW-100-4-0d2-trial2	100	200	{1}	7	7	0.0404
SW-100-4-0d2-trial3	100	200	{1}	7	7	0.0404	SW-100-4-0d3-trial1	100	200	{1}	7	6	0.0404
SW-100-4-0d3-trial2	100	200	{1}	7	6	0.0404	SW-100-4-0d3-trial3	100	200	{1}	7	7	0.0404
SW-100-5-0d1-trial1	100	200	{1}	7	8	0.0404	SW-100-5-0d1-trial2	100	200	{1}	7	9	0.0404
SW-100-5-0d1-trial3	100	200	{1}	7	11	0.0404	SW-100-5-0d2-trial1	100	200	{1}	7	8	0.0404
SW-100-5-0d2-trial2	100	200	{1}	7	9	0.0404	SW-100-5-0d2-trial3	100	200	{1}	7	7	0.0404
SW-100-5-0d3-trial1	100	200	{1}	7	6	0.0404	SW-100-5-0d3-trial2	100	200	{1}	7	6	0.0404
SW-100-5-0d3-trial3	100	200	{1}	7	6	0.0404	SW-100-6-0d1-trial1	100	300	{1}	7	5	0.0606
SW-100-6-0d1-trial2	100	300	{1}	7	6	0.0606	SW-100-6-0d1-trial3	100	300	{1}	7	6	0.0606
SW-100-6-0d2-trial1	100	300	{1}	7	6	0.0606	SW-100-6-0d2-trial2	100	300	{1}	7	4	0.0606
SW-100-6-0d2-trial3	100	300	{1}	7	4	0.0606	SW-100-6-0d3-trial1	100	300	{1}	7	4	0.0606
SW-100-6-0d3-trial2	100	300	{1}	7	5	0.0606	SW-100-6-0d3-trial3	100	300	{1}	7	5	0.0606
SW-1000-3-0d2-trial1	1000	1000	{1}	10	89	0.002	SW-1000-3-0d2-trial2	1000	1000	{1}	10	88	0.002
SW-1000-3-0d3-trial2	1000	1000	{1}	10	87	0.002	SW-1000-4-0d1-trial1	1000	2000	{1}	10	14	0.004
SW-1000-4-0d1-trial2	1000	2000	{1}	10	15	0.004	SW-1000-4-0d1-trial3	1000	2000	{1}	10	15	0.004
SW-1000-4-0d2-trial1	1000	2000	{1}	10	10	0.004	SW-1000-4-0d2-trial2	1000	2000	{1}	10	10	0.004
SW-1000-4-0d2-trial3	1000	2000	{1}	10	11	0.004	SW-1000-4-0d3-trial1	1000	2000	{1}	10	9	0.004
SW-1000-4-0d3-trial3	1000	2000	{1}	10	8	0.004	SW-1000-5-0d1-trial1	1000	2000	{1}	10	14	0.004
SW-1000-5-0d1-trial2	1000	2000	{1}	10	15	0.004	SW-1000-5-0d1-trial3	1000	2000	{1}	10	12	0.004
SW-1000-5-0d2-trial1	1000	2000	{1}	10	11	0.004	SW-1000-5-0d2-trial2	1000	2000	{1}	10	10	0.004
SW-1000-5-0d2-trial3	1000	2000	{1}	10	10	0.004	SW-1000-5-0d3-trial1	1000	2000	{1}	10	9	0.004
SW-1000-5-0d3-trial2	1000	2000	{1}	10	9	0.004	SW-1000-5-0d3-trial3	1000	2000	{1}	10	10	0.004
SW-1000-6-0d1-trial1	1000	3000	{1}	10	10	0.006	SW-1000-6-0d1-trial2	1000	3000	{1}	10	9	0.006
SW-1000-6-0d1-trial3	1000	3000	{1}	10	8	0.006	SW-1000-6-0d2-trial1	1000	3000	{1}	10	8	0.006
SW-1000-6-0d2-trial2	1000	3000	{1}	10	8	0.006	SW-1000-6-0d2-trial3	1000	3000	{1}	10	7	0.006
SW-1000-6-0d3-trial1	1000	3000	{1}	10	6	0.006	SW-1000-6-0d3-trial2	1000	3000	{1}	10	6	0.006
SW-1000-6-0d3-trial3	1000	3000	{1}	10	7	0.006							
Synthetic instances													
$B_5 \cup RG_{32,0.05}$	32	48	{1}	5	5	0.0968	$B_5 \cup RG_{32,0.075}$	32	64	{1}	5	4	0.129
$B_5 \cup RG_{32,0.1}$	32	83	{1}	5	3	0.1673	$B_5 \cup RG_{32,0.15}$	32	89	{1}	5	3	0.1794
$B_5 \cup RG_{32,0.2}$	32	142	{1}	5	2	0.2863	$B_5 \cup RG_{32,0.25}$	32	156	{1}	5	2	0.3145
$B_6 \cup RG_{64,0.05}$	64	159	{1}	6	3	0.0789	$B_6 \cup RG_{64,0.075}$	64	184	{1}	6	3	0.0913
$B_6 \cup RG_{64,0.1}$	64	243	{1}	6	3	0.1205	$B_6 \cup RG_{64,0.15}$	64	349	{1}	6	2	0.1731
$B_6 \cup RG_{64,0.2}$	64	461	{1}	6	2	0.2287	$B_6 \cup RG_{64,0.25}$	64	558	{1}	6	2	0.2768
$B_7 \cup RG_{128,0.05}$	128	560	{1}	7	3	0.0689	$B_7 \cup RG_{128,0.075}$	128	716	{1}	7	3	0.0881
$B_7 \cup RG_{128,0.1}$	128	923	{1}	7	3	0.1136	$B_7 \cup RG_{128,0.15}$	128	1313	{1}	7	3	0.1615
$B_7 \cup RG_{128,0.2}$	128	1742	{1}	7	2	0.2143	$B_7 \cup RG_{128,0.25}$	128	2140	{1}	7	2	0.2633
$B_8 \cup RG_{256,0.05}$	256	1863	{1}	8	3	0.0571	$B_8 \cup RG_{256,0.075}$	256	2657	{1}	8	3	0.0814
$B_8 \cup RG_{256,0.1}$	256	3450	{1}	8	2	0.1057	$B_8 \cup RG_{256,0.15}$	256	5168	{1}	8	2	0.1583
$B_8 \cup RG_{256,0.2}$	256	6691	{1}	8	2	0.205	$B_8 \cup RG_{256,0.25}$	256	8307	{1}	8	2	0.2545
$B_9 \cup RG_{512,0.05}$	512	6881	{1}	9	3	0.0526	$B_9 \cup RG_{512,0.075}$	512	10304	{1}	9	3	0.0788
$B_9 \cup RG_{512,0.1}$	512	13444	{1}	9	2	0.1028	$B_9 \cup RG_{512,0.15}$	512	20009	{1}	9	2	0.153
$B_9 \cup RG_{512,0.2}$	512	27012	{1}	9	2	0.2065	$B_9 \cup RG_{512,0.25}$	512	33313	{1}	9	2	0.2547

Continued on next page

Table A.1 – continued from previous page

Instance	$\|V\|$	$\|E\|$	V_0	TLB	LBB	density	Instance	$\|V\|$	$\|E\|$	V_0	TLB	LBB	density
$B_{10} \cup RG_{1024,0.05}$	1024	27259	{1}	10	3	0.052	$B_{10} \cup RG_{1024,0.075}$	1024	40222	{1}	10	3	0.0768
$B_{10} \cup RG_{1024,0.1}$	1024	53480	{1}	10	2	0.1021	$B_{10} \cup RG_{1024,0.15}$	1024	79574	{1}	10	2	0.1519
$B_{10} \cup RG_{1024,0.2}$	1024	105448	{1}	10	2	0.2013	$B_{10} \cup RG_{1024,0.25}$	1024	131643	{1}	10	2	0.2513

A.2 Instances for the WMBT.

For each instance, Table A.2 gives the name (column 'Instance'), the number of vertices (column '$|V|$'), the number of edges (column '$|E|$'), the initial source vertex (column 'V_0'), the lower bound found by the proposed Algorithm 7 (column 'LBB'), and the edge density (column 'density').

Table A.2: Description of the test instances.

Instance	$\|V\|$	$\|E\|$	V_0	TLB	LBB	density	Instance	$\|V\|$	$\|E\|$	V_0	TLB	LBB	density
Random graph													
$RG_{64,10,1}$	64	274	{7}	6	3	0.1359	$RG_{64,15,1}$	64	352	{34}	6	3	0.1746
$RG_{64,20,1}$	64	456	{31}	6	3	0.2262	$RG_{64,25,1}$	64	549	{30}	6	2	0.2723
$RG_{128,10,1}$	128	917	{82}	7	20	0.1128	$RG_{128,15,1}$	128	1313	{104}	7	20	0.1615
$RG_{128,20,1}$	128	1753	{5}	7	18	0.2157	$RG_{128,25,1}$	128	2209	{57}	7	15	0.2718
$RG_{256,10,1}$	256	3493	{20}	8	15	0.107	$RG_{256,15,1}$	256	5069	{235}	8	16	0.1553
$RG_{256,20,1}$	256	6743	{190}	8	10	0.2066	$RG_{256,25,1}$	256	8404	{116}	8	12	0.2575
$RG_{512,10,1}$	512	13500	{408}	9	13	0.1032	$RG_{512,15,1}$	512	20081	{384}	9	16	0.1535
$RG_{512,20,1}$	512	26446	{267}	9	10	0.2022	$RG_{512,25,1}$	512	33103	{326}	9	13	0.2531
$RG_{1024,10,1}$	1024	53315	{553}	10	13	0.1018	$RG_{1024,15,1}$	1024	79258	{829}	10	15	0.1513
$RG_{1024,20,1}$	1024	105629	{246}	10	9	0.2017	$RG_{1024,25,1}$	1024	131816	{881}	10	9	0.2517
Random graph with optima know													
$RGO_{64,10,1}$	64	267	{48}	6	74	0.1324	$RGO_{64,15,1}$	64	354	{50}	6	108	0.1756
$RGO_{64,20,1}$	64	456	{32}	6	87	0.2262	$RGO_{64,25,1}$	64	548	{38}	6	140	0.2718
$RGO_{128,10,1}$	128	932	{30}	7	117	0.1147	$RGO_{128,15,1}$	128	1325	{42}	7	186	0.163
$RGO_{128,20,1}$	128	1705	{17}	7	167	0.2098	$RGO_{128,25,1}$	128	2090	{65}	7	122	0.2571
$RGO_{256,10,1}$	256	3461	{119}	8	250	0.106	$RGO_{256,15,1}$	256	5119	{124}	8	208	0.1568
$RGO_{256,20,1}$	256	6925	{163}	8	169	0.2122	$RGO_{256,25,1}$	256	8353	{1}	8	211	0.2559
$RGO_{512,10,1}$	512	13460	{272}	9	288	0.1029	$RGO_{512,15,1}$	512	20347	{337}	9	409	0.1555
$RGO_{512,20,1}$	512	26371	{300}	9	297	0.2016	$RGO_{512,25,1}$	512	32891	{317}	9	320	0.2514
$RGO_{1024,10,1}$	1024	53634	{778}	10	440	0.1024	$RGO_{1024,15,1}$	1024	79458	{51}	10	550	0.1517
$RGO_{1024,20,1}$	1024	105615	{871}	10	479	0.2016	$RGO_{1024,25,1}$	1024	131157	{421}	10	353	0.2504