Federal University of Rio Grande do Norte
Center for Exact and Earth Sciences
Department of Informatics and Applied Mathematics
Baccalaureate in Computer Science

A Transgenetic Algorithm for the Quadratic Minimum Spanning Tree Problem

Fernanda Menezes Paes Isabel

Natal-RN
November 2018

Fernanda Menezes Paes Isabel

A Transgenetic Algorithm for the Quadratic Minimum Spanning Tree Problem

Undergraduate thesis submitted to the Deparment of Informatics and Applied Mathematics of the Center for Exact and Earth Sciences of the Federal University of Rio Grande do Norte as a partial requirement for obtaining the degree of bachelor in Computer Science.

Advisor

Dr. Sílvia Maria Diniz Monteiro Maia

Federal University of Rio Grande do Norte – UFRN
Department of Informatics and Applied Mathematics – DIMAp

Natal-RN

November 2018

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Setorial Prof. Ronaldo Xavier de Arruda - CCET

Isabel, Fernanda Menezes Paes.
   A transgenetic algorithm for the quadratic minimum spanning tree problem / Fernanda Menezes Paes Isabel. - 2018.
   49f.: il.

   Monografia (Bacharelado em Ciência da Computação) - Universidade Federal do Rio Grande do Norte, Centro de Ciências Exatas e da Terra, Departamento de Informática e Matemática Aplicada, Curso de Ciência da Computação. Natal, 2018.
   Orientadora: Sílvia Maria Diniz Monteiro Maia.

   1. Computação - Monografia. 2. QMST - Monografia. 3. Meta-heurística - Monografia. 4. Algoritmo transgenético - Monografia. I. Maia, Sílvia Maria Diniz Monteiro. II. Título.

RN/UF/CCET   CDU 004

Elaborado por Joseneide Ferreira Dantas - CRB-15/324

Undergraduate thesis under the title *A Transgenetic Algorithm for the Quadratic Minimum Spanning Tree Problem* presented by Fernanda Menezes Paes Isabel and accepted by the Department of Informatics and Applied Mathematics of the Center for Exact and Earth Sciences of the Federal University of Rio Grande do Norte, being approved by all members of the examining board specified below:

Dr. Sílvia Maria Diniz Monteiro Maia
Advisor
Department of Informatics and Applied Mathematics
Federal University of Rio Grande do Norte

Dr. Elizabeth Ferreira Gouvêa Goldbarg
Department of Informatics and Applied Mathematics
Federal University of Rio Grande do Norte

Dr. Marco César Goldbarg
Department of Informatics and Applied Mathematics
Federal University of Rio Grande do Norte

Natal-RN, November 30, 2018.

To my parents, Liana and Paulo.

Acknowledgments

I would first like to thank my advisor, Sílvia, for all the orientation and knowledge she gave me during this one year and a half. She was able to guide me in the right direction every time I needed it and always made me believe that everything would work out.

I would also like to thank professors Elizabeth Goldbarg and Marco Goldbarg, for being part of the examining board.

Although he will never read this, I would like to thank my dog, Alfredo, for all the emotional support he has given me. Every time I felt desperate or exhausted, hugging him made me feel better and gave me strength to continue.

I would also like to thank my friends, the ones who are close and the ones who are far away, for all the academic help they provided me and for all the laughs and nice moments we had together. Without them I wouldn't have had the required energy to accomplish this work.

Finally, I want to thank my parents, Liana and Paulo, for all the support and unconditional love. They have always been and will always be my compass and my rock. They have always believed in me and that made me believe in myself. I hope that, with this work, I continue to make them proud.

Success is the ability to go from one failure to another with no loss of enthusiasm. – WINSTON CHURCHILL [...] [This quote] pinpoints the single greatest strength of computers. [...] [C]omputers are infinitely persistent. [They] can fail billions of times with no trace of frustration. [They] embark upon [their] billionth attempt at solving a problem with the same energy as [their] first. Humans cannot do that.

Dan Brown

Um Algoritmo Transgenético para o Problema da Ávore Geradora Mínima Quadrática

Autor(a): Fernanda Menezes Paes Isabel

Orientador(a): Dra. Sílvia Maria Diniz Monteiro Maia

Resumo

O problema da árvore geradora mímina quadrática é uma variação do problema clássico da árvore geradora mínima que considera o custo de pares de arestas além do custo de arestas. Esse problema é NP-difícil e portanto não é conhecida uma solução exata em tempo polinomial para ele. Desde que foi descrito pela primeira vez, várias aborgadens meta-heurísticas foram aplicadas a esse problema, tentando encontrar soluções quase ótimas. Destas, a técnica de busca tabu foi a que apresentou resultados mais promissores, já tendo sido aplicada com sucesso a esse problema muitas vezes. Este trabalho propões uma nova abordagem meta-heurística para esse problema, um algoritmo transgenético. O algoritmo transgenético é um algoritmo evolucionário que já mostrou bons resultados para outros problemas de otimização. O algoritmo transgenético proposto foi implementado, assim como uma busca tabu da literatura para comparação. Experimentos computacionais foram realizados e os resultados são descritos e analisados neste trabalho.

Palavras-chave: AGMQ, meta-heurística, algoritmo transgenético.

A Transgenetic Algorithm for the Quadratic Minimum Spanning Tree Problem

Author: Fernanda Menezes Paes Isabel

Advisor: Dr. Sílvia Maria Diniz Monteiro Maia

Abstract

The Quadratic Minimum Spanning Tree Problem is a variation of the classic Minimum Spanning Tree Problem which considers the costs of pairs of edges as well as the costs of edges. This problem is NP-hard and therefore there isn't a known polynomial time solution for it. Since it was first described, several metaheuristic approaches have been applied to this problem, trying to find close-to-optimal solutions. Out of those, the tabu search technique has shown the most promising results, having been successfully applied to this problem many times. This work proposes a new metaheuristic approach to this problem, a transgenetic algorithm. The transgenetic algorithm is an evolutionary algorithm that has shown good results for other optimization problems. The proposed transgenetic algorithm was implemented, as well as a tabu search from the literature for comparison purposes. Computational experiments were performed on those algorithms and the results are described and analyzed in this work.

Keywords: QMST, metaheuristic, transgenetic algorithm.

List of tables

List of abbreviations

QMST	quadratic minimum spanning tree
AQMST	adjacent-only quadratic minimum spanning tree
MST	minimum spanning tree
QAP	quadratic assignment problem

List of symbols

$G(V, E)$ connected undirected graph with vertex set V and edge set E

n number of nodes in a graph G

m number of edges in a graph G

v_i node of graph G, where $i \in [1, n]$

e_i edge of graph G, where $i \in [1, m]$

$\mathcal{T}$ set of all spanning trees of graph G

T a spanning tree of graph G

$E(T)$ edge set of tree T

c_i cost of edge i

b_i cost of edge i

a_{ij} intercost of pair of edges (i, j)

List of algorithms

Contents

1 Introduction

This work studies the quadratic minimum spanning tree (QMST) problem and proposes a transgenetic algorithm to find an approximate solution to it. The QMST problem is a combinatorial optimization problem proposed by (ASSAD; XU, 1992) and consists of finding a spanning tree of minimum cost given a graph with weights associated to each edge and each pair of edges, where the cost of a tree is the sum of the weights of the edges (linear cost) and the pair of edges (quadratic cost) in that tree. It is an extension of the well known minimum spanning tree problem. When the quadratic costs exist only for adjacent edges, the problem is named adjacent-only quadratic minimum spanning tree (AQMST). Both QMST and AQMST problems are NP-hard, thus no exact algorithm known so far can solve them in polynomial time. Several metaheuristic approaches have been proposed to find approximate solutions in polynomial time. This work presents a literature review of this problem and these metaheuristics developed to solve it. We are proposing a new metaheuristic approach to this problem: a transgenetic algorithm.

1.1 Motivation

This problem is applied in real situations when the interaction between edges must be considered, instead of just the contribution of each individual edge. It often appears when planning distribution networks. When planning the distribution network for oil and its derivatives, for example, it's necessary to take into account the cost of active equipment installed in the connections of ducts. Another example is the design of energy networks with overground and underground cables: special protection equipment is required when there's a connection between over and underground cables and vice versa. That is, in both cases, the cost of the intersections matters.

1.2 Objectives

The objective of this work is to develop a transgenetic algorithm to solve the QMST problem. This is the first time this metaheuristic is being applied to this problem. Transgenetic algorithms were already applied to the bi-objective version of AQMST in (MAIA, 2013) and (PINHEIRO, 2016). According to (MAIA, 2013) and (PINHEIRO, 2016), this approach was successful when compared to other evolutionary approaches, while tabu search had better results. We aim to investigate the application of transgenetic algorithms to solve QMST, hoping to improve literature results.

1.3 Justification

Due to the QMST problem being NP-hard, finding optimal solutions with exact algorithms takes a large amount of time even for small test cases. Solving for the optimal solution proves to be impossible for large test cases, as it would take hundreds of years. The alternative to this is solving not for the optimal solution but for approximate solutions, using methods such as approximation algorithms, heuristics and metaheuristics, that run in polynomial time and try to find close-to-optimal solutions. Approximation algorithms know the distance from the approximate solution to the optimal one, obtained from mathematical proof (WILLIAMSON; SHMOYS, 2011). Heuristics and metaheuristics find good approximate solutions, although with no guarantees. Heuristics are problem specific, designed specifically for each problem, while metaheuristics are general methods that can be slightly adapted to be used in almost every optimization problem (TALBI, 2009). Metaheuristics usually find better solutions than heuristics, although their running time is usually higher.

For the QMST problem, metaheuristics have shown good results. More specifically, the tabu search algorithm, which has been implemented in different ways for this problem in various papers, finds close-to-optimal results most of the time.

The transgenetic algorithm, an evolutionary metaheuristic, hasn't yet been applied to this problem, but has shown promising results to several NP-hard problems such as the bi-objective spanning tree problem (ROCHA; GOLDBARG; GOLDBARG, 2007). In light of that we will also design a transgenetic algorithm for the QMST problem.

1.4 Methodology

After the design and implementation of the transgenetic algorithm, experimental tests were performed and the results were analyzed. The tabu search algorithm proposed by (CORDONE; PASSERI, 2012) was also implemented, for comparison purposes.

The results were compared to the existing ones in the literature. In order to do that, computational experiments were carried out using benchmark instances provided by (CORDONE; PASSERI, 2012). The instances are graphs that range from 10 to 50 vertices, with densities of 33%, 67% or 100% and linear and quadratic costs randomly generated in the range of [1;10] or [1;100]. These instances have already been used in many of the mentioned articles, making it easier to compare results. Statistical testing was performed to analyze the relevance of obtained results.

1.5 Outline of this work

In chapter 2, we provide the problem definition, along with similar problems, the analysis of these problems and the literature review of related works to these problems.

Chapter 3 describes the transgenetic algorithm metaheuristic and the developed algorithm for the QMST problem.

In chapter 4, the computational experiments performed are described and their results are reported and analyzed.

Finally, chapter 5 has some final considerations.

2 Theoretical background

In this chapter we provide a formal definition for the QMST problem, as defined by (ASSAD; XU, 1992), along with examples of similar problems and an analysis of these problems. This chapter also features the literature review for this problem.

2.1 Problem definition

Let $G(V, E)$ be a connected undirected graph with vertex set $V = \{v_1, ..., v_n\}$ and edge set $E = \{e_1, ..., e_m\}$. Let $\mathcal{T}$ denote the set of all spanning trees of graph G, that is, $T \in \mathcal{T}$ if and only if T is a tree, $E(T) \subseteq E$ and $|E(T)| = n - 1$.

The minimum spanning tree (MST) problem is a well known problem in computer science. It is defined as finding a spanning tree T such that

$$\min \left\{ \sum_{e_i \in E(T)} c_i \mid T \in \mathcal{T} \right\}$$

where c_i is the cost of edge e_i.

(ASSAD; XU, 1992) defined an extension to this problem which considers the quadratic cost of the tree as well as the linear cost, known as the quadratic minimum spanning tree (QMST) problem. It consists on finding a spanning tree T such that

$$\min \left\{ \sum_{e_i \in E(T)} \sum_{\substack{e_j \in E(T) \\ e_i \neq e_j}} a_{ij} + \sum_{e_i \in E(T)} b_i \mid T \in \mathcal{T} \right\}$$

where a_{ij} is the cost of the pair of edges (e_i, e_j) and b_i is the cost of edge e_i. When the a_{ij} costs are considered only for adjacent edges, the problem is known as the adjacent-only QMST (AQMST).

Other variations of this problem include fuzzy QMST (GAO; LU, 2005), uncertain

QMST (ZHOU; HE; WANG, 2014) and biobjective adjacent-only quadratic spanning tree (MAIA; GOLDBARG; GOLDBARG, 2013).

2.2 Problem analysis

There are several greedy approaches capable of solving the MST problem in polynomial time, such as the ones proposed by (KRUSKAL, 1956) and (PRIM, 1957). However, the same does not apply to the QMST and AQMST problems. (ASSAD; XU, 1992) proved theses problems to be NP-hard, which means no algorithm known so far is able to solve them in polynomial time.

The QMST is a combinatorial optimization problem. Optimization problems are those searching for a best solution from all feasible solutions. When dealing with discrete variables, it's called a combinatorial optimization problem (PAPADIMITRIOU; STEIGLITZ, 1982).

2.3 Related work

Lower bounds to the problem have been given by (ASSAD; XU, 1992), (ÖNCAN; PUNNEN, 2010) and (PEREIRA; GENDREAU; CUNHA, 2015), while exact branch-and-bound algorithms have been given by (ASSAD; XU, 1992), (CORDONE; PASSERI, 2012) and (PEREIRA; GENDREAU; CUNHA, 2015).

Many heuristics have been proposed to find a solution that best approximates the optimal one for this problem in polynomial time. (ASSAD; XU, 1992) described two heuristics along with the definition of the problem. Since then, other authors have been able to obtain better solutions, mostly with the use of metaheuristics. The tabu search method has been applied in various works, such as (CORDONE; PASSERI, 2012), (PALUBECKIS; RUBLIAUSKAS; TARGAMADZė, 2010), (LOZANO et al., 2014), and has shown good results for this problem. Simulated annealing (PALUBECKIS; RUBLIAUSKAS; TARGAMADZė, 2010), artificial bee colony (SUNDAR; SINGH, 2010), genetic algorithm (ZHOU; GEN, 1998), (PALUBECKIS; RUBLIAUSKAS; TARGAMADZė, 2010), (SOAK; CORNE; AHN, 2006) and different types of local search (ÖNCAN; PUNNEN, 2010), (CORDONE; PASSERI, 2012), (FU; HAO, 2015) approaches have also been applied to this problem.

2.3.1 (ASSAD; XU, 1992)

(ASSAD; XU, 1992), along with the problem definition, described a lower bound for the problem, as well as an exact branch-and-bound algorithm for solving the problem and two heuristics for finding approximate solutions to the problem in polynomial time.

The lower bound to the QMST problem is obtained tby solving $m+1$ MST problems. For each edge e_i in $\{e_1, ..., e_m\}$ an MST M_i is calculated by fixing edge e_i and selecting other edges that form a spanning tree such that the sum of the intercosts of those edges and edge e_i is minimized. Then, a final MST is calculated by selecting M_is that minimize the sum of their costs while the set of edges e_i forms a spanning tree. A leveling procedure is used to adjust parameters and calculate several lower bounds.

A couple of exact branch-and-bound algorithms are also described, one for the QMST and one for the AQMST.

This paper also describes a couple of heuristic algorithms. The first one estimates the average contribution of each edge to the tree value, then solves an MST with those averages as edges' costs. The second one also estimates the average contribution of each edge to the tree value, but does so n times: in each of n steps, it fixes one new edge, the one with lower average contribution, and then recalculates the contributions of the other edges. Both heuristics have a time complexity of $O(n^2)$.

Because this was the first time this problem was studied and solutions for it were proposed, the authors were not able to compare results with other works. They did state, however, that possible better solutions will come from techniques effective to the quadratic assignment problem (QAP), like tabu search metaheuristics.

2.3.2 (ZHOU; GEN, 1998)

This paper was the first to proposed a metaheuristic approach to the QMST problem. It presents a genetic algorithm for approximating a solution for the QSMT. This algorithm uses the Prüfer number to encode the tree.

The genetic algorithm imitates the process of natural selection by starting from a set of candidate solutions and improving them step by step through biological evolutionary processes like crossover and mutation.

The Prüfer number allows us to represent a tree with n vertices with $n-2$ digits, where each digit is an integer between 1 and n, inclusive. This encoding is suitable for this algorithm because it's capable of representing all possible trees in a complete graph with n vertices uniquely, and modification through crossover or mutation will still generate a tree.

The algorithm proposed in this paper uses uniform crossover, which generates a random mask (a binary string of the size of the encoding) and then exchanges digits between parents according to the mask to create offspring. Mutation is performed by replacing a digit in a random position with a random number.

The evaluation is the calculation of the fitness of each chromosome, and is done by converting a chromosome into a tree and calculating the cost of that tree, according to the QMST equation. The selection of the next generation according to their fitness value is done with a mixed strategy; μ best chromosomes are selected from μ parents and λ offspring ($(\mu + \lambda)$-selection) and if there are no μ different chromosomes, the missing ones are chosen with *roulette wheel* selection.

The steps for the genetic algorithm are: initialize the parent population; evaluate the parent population; create offspring; evaluate the offspring; select a new parent population out of the parents and offspring; go back to creating offspring if terminating condition hasn't been reached.

The genetic algorithm, when compared to the heuristic algorithms proposed by (ASSAD; XU, 1992), presented an improvement of the solution with an average of 9.6%, although the genetic algorithm takes more CPU time.

2.3.3 (SOAK; CORNE; AHN, 2006)

(SOAK; CORNE; AHN, 2006) proposes a genetic algorithm with a new way of encoding trees, the "edge-window-decoder" (EWD) representation. This was motivated by the fact that Prüfer number encoding is known to have poor locality and usually is outperformed by other types of tree representation.

This algorithm with the EWD encoding showed better results than others, including the Prüfer number.

2.3.4 (SUNDAR; SINGH, 2010)

This paper introduces an artificial bee colony (ABC) algorithm for solving the QMST problem.

Here, the spanning tree is represented by a list of its edges. The algorithm is initialized with a randomly generated spanning tree associated with each employed bee. Binary tournament selection was used instead of roulette wheel selection. To determine a neighboring food source, an edge of the spanning tree is removed and we try to replace it with an edge from another food source. If after some iterations this is not possible, this spanning tree/food source is abandoned and the bee associated with it becomes a scout and is employed to a randomly generated spanning tree.

After the end of the algorithm, the best solution found is further improved, or at least tried to, by a local search.

Computational results showed this algorithm to be superior to the genetic algorithms proposed by (ZHOU; GEN, 1998) and Soak et al, 2006.

2.3.5 (ÖNCAN; PUNNEN, 2010)

(ÖNCAN; PUNNEN, 2010) start by proposing a Lagrangian relaxation scheme for obtaining lower bound values for the QMST problem.

They also propose a local search algorithm for solving the problem. This local search is performed in a neighborhood scheme that is defined as the replacement of k edges in the current solution tree by k edges outside this tree that still form a spanning tree. In the paper, they only used a neighborhood scheme of $k = 1$, as it showed good results with a reasonable CPU time. After finding the best possible solution from the neighborhood of the current solution, *i.e.* a local minimum, the algorithm also performs a series of random moves on this solution.

The computational experiments reported in (ÖNCAN; PUNNEN, 2010) show that the proposed local search algorithm yields better results than the heuristics proposed by (ASSAD; XU, 1992), although these results were never optimal.

2.3.6 (PALUBECKIS; RUBLIAUSKAS; TARGAMADZė, 2010)

This paper presents multistart simulated annealing, hybrid genetic and iterated tabu search algorithms for solving the QMST problem.

The simulated annealing algorithm is based on the idea of heating up a material to a very high temperature then slowly cooling it down to reach the lowest energy state. The algorithm starts with a "high temperature" (high cost spanning tree) chosen out of thousands of random spanning trees. This temperature is then multiplied by a cooling factor at each iteration, trying to find a temperature that is stable.

Unlike in (ZHOU; GEN, 1998), the genetic algorithm described in this paper doesn't use Prüfer number for encoding trees, because this number can only be used when the graph is complete; the tree's list of edges is used instead. First, a random population is created. Then, crossover of two trees is done by maintaining the edges both have in common and choosing randomly the ones that are present in only one tree and still maintain a spanning tree. Then, in the hybrid version of the genetic algorithm, the offspring is submitted to a local search procedure. The population is updated by replacing the worst individual with the offspring.

Tabu search is a local search method that allows worsening solutions to be explored, but keeps memory of some solutions that shouldn't be explored again, called "tabu". In this paper, the tabu search method is iterative, meaning the procedure is done several times. At the beginning of each iteration, a perturbation mechanism is used to generate a new spanning tree.

Computational results showed the tabu search algorithm to be the one with the best solutions, followed by the hybrid genetic algorithm.

2.3.7 (CORDONE; PASSERI, 2012)

(CORDONE; PASSERI, 2012) also developed a tabu search algorithm. It runs for a certain amount of iterations I, and starts from a randomly generated spanning tree every I/r iterations. Each iteration, a local search is performed by checking feasible solutions on the neighborhood. The neighborhood here is defined as all possible spanning trees obtained by exchanging one edge in the current tree by one that isn't in the tree. A move, *i.e.* an edge substitution, is considered tabu if the edge to be added has been in the tree in one of the last l_{in} iterations or if the edge to be removed has not been in the tree in one of the last l_{out} iterations. In that case, the solution cost obtained by that move can only be considered if it's better than the best solution found yet. At the end of each iteration, the tree is updated to the one with the best considered solution value. The values l_{in} and l_{out} can vary inside some intervals, and are updated at the end of each iteration, decreasing if the solution has improved and increasing if it has worsened.

This paper also describes a variable neighborhood search and a branch and bound algorithm.

The computational experiments done in this paper show that the proposed tabu search performs better than the variable neighborhood search and most of the state-of-the-art heuristics that existed in the literature.

2.3.8 (LOZANO et al., 2014)

This paper proposes a strategic oscillation approach to the QMST problem that also uses a tabu search mechanism. The strategic oscillation consists of a destructive phase, where edges are removed from a solution, and a constructive phase, where edges are added until a solution in formed. In the destructive phase, some of the removed edges are labeled as tabu and can't be added again in the constructive phase. At the end of the constructive phase, a tabu search method is called to improve the current solution.

The tabu search method first computes measures of each edges' contribution to the solution, then assigns probabilities to these edges proportional to these measures and chooses one randomly, but given those probabilities, to be removed. After evaluating the edges that form feasible solutions, the one that results in the best solution is added to the tree. The measures are recalculated for the next iteration. The moved edges are labeled as tabu and can't be added to or removed from the solution in some predefined amount of iterations.

This hybrid method proposed by (LOZANO et al., 2014) performs better than (PALUBECKIS; RUBLIAUSKAS; TARGAMADZė, 2010)'s tabu search and (SUNDAR; SINGH, 2010)'s artificial bee colony algorithms for large instances.

2.3.9 (FU; HAO, 2015)

(FU; HAO, 2015) proposes a three-phase search approach for the QMST problem.

The first phase starts from a randomly generated solution and consists of a descent-based neighborhood search. This search finds a local optimal solution by first examining the neighborhood formed by removing one edge and then, if no improving solution is found, by examining the neighborhood formed by removing two edges.

The second phase explores the current search space to find possible improvements, by performing direct perturbation.

The third phase performs a diversified perturbation to go from one search area to a more distant one, by strongly modifying the tree.

This algorithm performs at least as well as most ones in the literature, showing highly competitive results.

2.3.10 (PEREIRA; GENDREAU; CUNHA, 2015)

This paper produces new lower bounds for the QMST problem that are tighter than the ones previously defined in the literature. These lower bounds are obtained through linear programming and a Lagrangian relaxation scheme.

(PEREIRA; GENDREAU; CUNHA, 2015) also describes two new branch-and-bound algorithms, both based on Lagrangian relaxation schemes. These algorithms were implemented using parallel programming, making there more efficient and consequently able to find optimal solutions for bigger test cases.

3 Transgenetic algorithm

This chapter describes the metaheuristic called transgenetic algorithm, an evolutionary algorithm proposed by (GOLDBARG; GOLDBARG, 2009), and the designed algorithm that applies this technique to the QMST problem. This is the first time this metaheuristic is being applied to this problem. It has been previously applied in the bi-objective adjancent-only version of the QMST, in (MAIA, 2013) and (PINHEIRO, 2016).

3.1 Technique

The transgenetic algorithm is an evolutionary algorithm, which is a class of metaheuristic approaches that are based on biological evolution. These types of algorithms work over a population of individuals, i.e. candidate solutions, in which mechanisms such as mutation and reproduction are applied to form new generations. Individuals are evaluated according to a fitness function, and the algorithm runs iteratively until a stopping criteria is reached (VIKHAR, 2016).

Unlike other evolutionary algorithms like the genetic algorithm, where the information is passed vertically from parent to offspring, the transgenetic algorithm is based on the endosymbiosis and the horizontal gene transfer. Endosymbiosis is the relationship between one organism (individual of the population) and the organism it lives inside of (host). Horizontal gene transfer is the insertion of outside genes into an individual (GOLDBARG; GOLDBARG, 2009). Thus, the evolutionary context under consideration is the evolution of a population of endosymbionts within their host.

Transgenetic vectors, also called agents, are used to exchange information between hosts and individuals from the population. There are four transgenetic vectors: plasmid, recombinant plasmid, transposon and virus. These vectors use 4 types of procedures to manipulate individuals: attack, which checks if the individual can be manipulated by the vector; transcription, that specifies how the information will be transcribed in the individual if the attack is positive; blocking, which doesn't allow the transcribed information to be modified for some period of time; and identification, that limits the vector's action to some part of the individual.

The plasmid vector uses the procedures of attack and transcription, where the information is a subset of the individual obtained from a host, and it is transcribed in the individual. The recombinant plasmid is similar and uses the same procedures, the only difference being the origin of the plasmid, that can be from many hosts or built from other sources, as well as a combination of both. The virus vector is similar to plasmids, but it requires the transcribed information to stay unaltered in the individual for some period of time, that is, it also uses the blocking procedure. Finally, the transposon vector chooses a subset of the genes of the individual that will be the only area it manipulates, meaning it used the procedures of attack, transcription and identification. The transposon can permutate genes inside that area, exchange genes between that area and the rest of the individual or randomly modify the genes in that area (GOLDBARG; GOLDBARG, 2009).

The transgenetic algorithm is an iterative algorithm, and runs until a stopping criteria is reached. The initial population and the host repository are generated in the beginning of the algorithm. At each iteration, for each of some chosen number of individuals of the population, a transgenetic vector is chosen to manipulate this individual. Also at each iteration, the population and the host repository may be updated.

3.2 Applied Algorithm

An algorithm which applies the technique described above to the QMST problem was designed as the central part of this work. Some of the design decisions were inspired by (MAIA, 2013).

In this algorithm, individuals of the population and hosts are trees, represented as a set of edges.

3.2.1 Overview

The algorithm starts by generating the initial population and the host repository, then enters its main loop, which runs for I iterations. The algorithm uses three transgenetic vectors: plasmid, virus and transposon. The probabilities of the transgenetic vectors are updated every r iterations, while the plasmid set is updated every iteration. At each iteration, for each individual of the population, an agent is chosen randomly according to the agents' probabilities to manipulate that individual. The current individual is updated if the new individual is better. The host repository is updated at the end of every iteration, replacing its worst solution with the best one from the iteration. The fitness function used to evaluate an individual is the cost of the tree, according to equation presented in Chapter 2. A pseudo code of the algorithm can be seen in algorithm 1.

Algorithm 1 Transgenetic Algorithm

```
 1: Generate initial population
 2: Generate host repository
 3: for $i = 1$ ... $I$ do
 4:     Update vector probabilities
 5:     Update plasmid set
 6:     for all individuals of the population do
 7:         $p$ = random number in [0.0, 1.0]
 8:         if $p <= plasmid\_probability$ then
 9:             Apply plasmid agent
10:         else if $p <= virus\_probability$ then
11:             Apply virus agent
12:         else
13:             Apply transposon agent
14:         end if
15:         if new individual is better than current one then
16:             Update current individual with new one
17:         end if
18:     end for
19:     Update host repository
20: end for
```

3.2.2 Initial population and host repository

The initial population was generated semi randomly using a modified Kruskal's algorithm. Edges are first ordered by their influence costs, which are their linear cost plus their quadratic cost with every other edge. Every edge receives a weight inversely proportional to their influence, from 1 to the number of edges. At each iteration, an edge is chosen randomly to be added to the tree, with a discrete distribution according to those weights. Weights are updated every iteration such that every edge that closes a cycle in the tree receives weight 0, and other edges have their weight adjusted such that the highest weight is the number of available edges to choose from. We can quickly identify if the edge closes a cycle by maintaining a union find data structure of the tree. The procedure stops once the tree reaches $n-1$ edges. Except for the ordering of the edges, which is done once at the beginning, these steps are done for each individual, i.e. tree. The individual is added to the population if it is not already a part of it, until the population reaches its pre-specified size.

The host repository is created using a priori information from the Heuristic 2 described by (ASSAD; XU, 1992). For each of 11 vector of weights {[1.0, 0.0], [0.9, 0.1], [0.8, 0.2], ..., [0.2, 0.8], [0.1, 0.9], [0.0, 1.0]}, a tree is created using Heuristic 2 with a linear combination of the weights and the linear and quadratic influences of the edges. This was done to span the diverse types of inputs, where linear costs can be way bigger or smaller than the quadratic costs.

3.2.3 Transgenetic vectors

Three different agents (transgenetic vectors) were developed: a plasmid, a virus and a trasposon. Each one starts with a probability that will be used to choose the agent for each individual of the population at each iteration, and these probabilities are updated through the algorithm. At the beginning, the plasmid has a higher probability of being chosen due to it taking data from the host repository, which is created with a priori information.

3.2.3.1 Plasmid

Every iteration of the algorithm, a new set of plasmids is created from the current host repository, which is also updated every iteration. The plasmid is a subset of the host, and therefore it is also a set of edges. Each plasmid is made from a randomly chosen information from host's repository, and has a randomly chosen size from a specified range. First, the edges of the host are ordered according to their influence costs. Then, the edges from the host that will form the plasmid are chosen randomly with a discrete distribution, where the probability of an edge being chosen is inversely proportional to its influence cost.

When the plasmid vector is chosen as the agent for manipulating an individual, a plasmid is chosen randomly from the set of plasmids. The new individual will contain all the edges from this plasmid, and some from the current individual. To generate the new individual, first we add all the edges from the plasmid. Then, we find the remaining edges, which are the edges from the current individual which don't close any cycles with the new individual. The influence costs of the remaining edges are computed in regards to the edges in the new individual. Next, while the new individual has less edges than the current one, the remaining edge with lowest influence cost is added to the new individual. After inserting each edge, the remaining edges are updated to exclude the ones that close cycles with the edges in the new individual and their influence costs are updated with the quadratic costs associated with the inserted edge. Again, checking whether a cycle is being closed can be quickly done by maintaining a union find data structure of the new individual. Pseudo code for the plasmid vector can be seen in algorithm 2.

Algorithm 2 Plasmid vector

```
1: Choose random plasmid from plasmid set
2: Initialize new individual with edges from the plasmid
3: Initialize remaning edges with the edges outside the current individual that don't close
   cycles in the new individual
4: Compute influence costs of remaining edges
5: while new individual has less edges than current individual do
6:     Add remaining edge with lowest influence cost to the new individual
7:     Remove from remaining edges the ones that close cycles in the updated new indi-
   vidual
8:     Update influence costs of remaining edges with the quadratic costs associated with
   the inserted edge
9: end while
```

3.2.3.2 Virus

This virus vector is similar to an autonomous recombinant plasmid, in the sense that the information chain does not come from the host repository (GOLDBARG; GOLDBARG, 2009). Instead, this information chain is built from the edges not in the current individual, and therefore it is different and personalized for each individual. Its size is chosen randomly from a specified range, and its edges are the ones with lowest influence cost in regards to the current individual, out of the edges that are not in the current individual.

To generate the new individual, a procedure inspired by the tabu search of (CORDONE; PASSERI, 2012) was developed. The blocking procedure of the virus vector is done using the tabu criteria. For each edge of each individual, we save the last iteration it was inserted or removed from that individual. Parameters l_{in} and l_{out} indicate the minimum number of iterations for an edge to be able to be inserted again after being removed and vice versa, respectively.

The new individual starts equal to the current individual. For each edge of the information chain, from lowest to highest influence cost, we find the cycle it closes in the new individual. Then we analyze the exchange of each of the edges in this cycle for the edge of the information chain and choose the best one that is not considered tabu. To find the cycle, we used the same mechanism of keeping a vector representation of a rooted tree as in the tabu search, described in (CORDONE; PASSERI, 2012). If the exchange results in the best solution found yet in the algorithm, we allow it even if it's tabu, which we call an aspiration criteria. The new individual is updated according to the chosen exchange. The new individual will not necessarily contain every edge from the information chain, as all the exchanges with an edge of the information chain may be considered tabu and not satisfy the aspiration criteria. Pseudo code for the virus vector can be seen in algorithm 3.

Algorithm 3 Virus vector

```
 1: Create information chain from edges that are not in the current individual
 2: Initialize new individual with edges from the current individual
 3: for all edges of the information chain, from lowest to highest influence cost do
 4:     Find cycle it closes in the new individual
 5:     for all edges of the cycle do
 6:         if exchange of edge of the cycle for the edge of the information chain that is
    not tabu or satisfies aspiration criteria then
 7:             Consider this exchange
 8:         end if
 9:     end for
10:     Update new individual best considered exchange
11: end for
```

3.2.3.3 Transposon

The transposon vector starts by determining the action range, a subset of the current individual which has the edges that will be replaced in the new individual. The size of the action range is randomly chosen from a specified range. The edges of the action range are chosen randomly with a discrete distribution, where the probability of an edge being chosen is proportional to the influence cost of that edge.

The new individual starts as the subset of edges from the current individual that are not in the action range. Next, we find the remaining edges, which are the edges outside of the current individual that do not close a cycle in the new individual, and compute their influence costs in regards to the edges in the new individual. Then, much like in the plasmid vector, we will add the edges from the remaining edges to the new individual. While there are still remaining edges, the remaining edge with lowest influence cost is added to the new individual. After inserting each edge, the remaining edges are updated to exclude the ones that close cycles with the edges in the new individual and their influence costs are updated with the quadratic costs associated with the inserted edge. If the new individual is not the same size as the current individual, it means the action range contained edges that couldn't be replaced by remaining edges, so we add back the necessary edges from the action range. Pseudo code for the transposon vector can be seen in algorithm 4.

Algorithm 4 Transposon vector

```
1: Find action range
2: Initialize new individual with edges from the current individual that are not in the
   action range
3: Initialize remaining edges with the edges outside the current individual that don't
   close cycles in the new individual
4: Compute influence costs of remaining edges
5: while there are remaining edges do
6:     Add remaining edge with lowest influence cost to the new individual
7:     Remove from remaining edges the ones that close cycles in the updated new indi-
       vidual
8:     Update influence costs of remaining edges with the quadratic costs associated with
       the inserted edge
9: end while
10: if new individual has less edges than current individual then
11:     Add necessary edges from action range to new individual
12: end if
```

3.2.4 Failed alternative methods

When implementing the algorithm, almost every part had an alternative way of being done, and most of those ways were actually implemented and compared to the current algorithm. A small subset of tests were performed to choose the best option in each part, as testing for every input case takes a lot of time. The options were tested for about two instances with between 5 and 10 executions each.

We started the implementation generating a completely random initial population, which resulted in slightly worse solutions. As the initial population is only generated once, the additional cost of doing some extra computations there was barely noticeable, and therefore worth it.

The host repository initially had only three solutions generated using a priori information, two from Heuristics 1 and 2 described in (ASSAD; XU, 1992) and one from a slight modification of Heuristic 1 (the exclusion of a coefficient in a formula, that seemed to improve the results). The rest of the solutions in the host repository were the best solutions from the initial population. The opportunity of having a priori information at the beginning of the algorithm helps improve its results, and, like the initial population, the host repository is only generated once, so the extra cost is small and therefore worth it.

In the plasmid and in the transposon, where the edge with lowest influence cost is inserted in the tree, we also tried to make the choice non deterministic, choosing randomly with a discrete distribution from the five edges with lowest influence cost. This is how it was done in (MAIA, 2013), but in the case of these two vectors in our algorithm it worsened the results by a lot. In contrast, making the choice of edges in the plasmid and in the the action range non deterministic improved the results.

Trying to introduce the plasmid into the individual gradually as it is done in the virus also did not achieve better results.

We also experimented with updating the plasmids only from time to time, instead of at every iteration, as well as updating the host repository only if the best solution from the iteration was better than the worst solution already in it, but both worsened the results.

Multiple starting values and rates of change were applied to the vector probabilities, including basing those on the success rate of each vector, but the current values presented the best results.

4 Computational experiments and results

Two versions of the transgenetic algorithm, one with the recombinant plasmid vector (version 1) and one without (version 2) were implemented, as well as the tabu search from (CORDONE; PASSERI, 2012). In the preliminary experiments, version 1 of the transgenetic algorithm presented better results than version 2, but took a lot more time to run, so we decided to perform the experiments and report the results for both.

All the algorithms were implemented in C++, compiled with g++ -std=c++11 with the optimization flag -O3 set and run in a 3.4 GHz Intel i7-2600k with 4GB of RAM memory. In this section, the benchmark instances and the parameters used for the experiments are described, and the results are reported and analyzed.

4.1 Benchmark instances

The benchmark instances used are from (CORDONE; PASSERI, 2012) and can be found at http://www.dti.unimi.it/cordone/research/qmst.html. They range from 10 to 50 nodes, with a density of either 33%, 67% or 100% of edges, with random linear and quadratic costs uniformly distributed in the range of either [1,10) or [1,100). For each number of nodes there are 12 instances, 4 for each density which are the 4 combinations of the ranges for linear and quadratic costs.

4.2 Parameters

Except for the probability of the transgentic vectors, versions 1 and 2 of the transgenetic algorithm have the same parameters. They run for 1400 iterations with a population of size 100. The plasmid set is made up of 3 plasmids. The size of the plasmid is randomly chosen with a uniform distribution in the range of [0.1, 0.3], while the sizes of the action range of the transposon and the recombinant plasmid are randomly chosen with a uniform distribution in the range of [0.1, 0.2]. The numbers of iterations an edge is considered tabu for are $l_{in} = 2$ and $l_{out} = 0.4 * n$.

In version 1, the plasmid vector starts with a probability of being chosen of 0.5, the recombinant plasmid vector with 0 and the transposon vector with 0.5. Every $I/5$ iterations, the plasmid and the transposon probabilities decrease 0.1, while the recombinant plasmid probability increases 0.2.

In version 2, the plasmid vector starts with a probability of being chosen of 0.8, the recombinant plasmid vector with 0 and the transposon vector with 0.2. Every $I/13$ iterations, the plasmid probability decreases 0.05 and the transposon probability increases 0.05, while the recombinant plasmid probability remains 0, so that it is never used.

These parameters were chosen by performing a small number of preliminary tests, where many other possibilities were experimented with and presented worse results.

The tabu search algorithm uses the same parameters as described in (CORDONE; PASSERI, 2012). It runs for 100000 iterations, only generating a random solution once. Values of the tabu range were set as $l_{in}^{min} = 1$, $l_{in}^{max} = 1$, $l_{out}^{min} = 0.35*n$ and $l_{out}^{max} = 0.45*n$.

4.3 Results

Each instance was executed 30 times, and the result costs and CPU times reported here are the mean of the 30 executions. Table 1 shows the result mean costs for each instance for the three algorithms, the best known results (BK) and the gap between the mean cost and the best known cost. Min and max costs for each instance can be seen in 2. Table 3 shows the CPU mean time in seconds for each instance for all three algorithms.

Table 1: Mean costs found by the algorithms for the benchmark instances.

Instance					Transgenetic 1		Transgenetic 2		Tabu Search	
n	d	C	Q	BK	Mean	Gap	Mean	Gap	Mean	Gap
10	33	10	10	350	350	0.00%	350	0.00%	350	0.00%
10	33	10	100	3122	3122	0.00%	3122	0.00%	3122	0.00%
10	33	100	10	646	646	0.00%	646	0.00%	646	0.00%
10	33	100	100	3486	3486	0.00%	3486	0.00%	3486	0.00%
10	67	10	10	255	255	0.00%	255	0.00%	255	0.00%
10	67	10	100	2042	2042	0.00%	2042	0.00%	2042	0.00.%
10	67	100	10	488	488	0.00%	488	0.00%	488	0.00%
10	67	100	100	2404	2404	0.00%	2404	0.00%	2404	0.00%

10	100	10	10	239	239	0.00%	239	0.00%	239	0.00%
10	100	10	100	1815	1815	0.00%	1815	0.00%	1815	0.00%
10	100	100	10	426	426	0.00%	426	0.00%	426	0.00%
10	100	100	100	2197	2197	0.00%	2197	0.00%	2205.87	0.40%
15	33	10	10	745	745	0.00%	745	0.00%	745.167	0.02%
15	33	10	100	6539	6539	0.00%	6539	0.00%	6544.27	0.08%
15	33	100	10	1236	1236	0.00%	1236	0.00%	1236	0.00%
15	33	100	100	7245	7245	0.00%	7245	0.00%	7245	0.00%
15	67	10	10	659	659	0.00%	659.4	0.06%	659	0.00%
15	67	10	100	5573	5573	0.00%	5578	0.09%	5573	0.00%
15	67	100	10	966	966	0.00%	966	0.00%	966	0.00%
15	67	100	100	6188	6188	0.00%	6193.6	0.09%	6196	0.13%
15	100	10	10	620	620	0.00%	620.267	0.04%	621.467	0.24%
15	100	10	100	5184	5184	0.00%	5184.9	0.02%	5185.8	0.03%
15	100	100	10	975	975	0.00%	975	0.00%	975	0.00%
15	100	100	100	5879	5879	0.00%	5879	0.00%	5880.2	0.02%
20	33	10	10	1379	1379	0.00%	1379	0.00%	1379	0.00%
20	33	10	100	12425	12425	0.00%	12425	0.00%	12425	0.00%
20	33	100	10	1972	1972	0.00%	1972	0.00%	1972	0.00%
20	33	100	100	13288	13288	0.00%	13288	0.00%	13288	0.00%
20	67	10	10	1252	1252	0.00%	1252	0.00%	1252	0.00%
20	67	10	100	10893	10893	0.00%	10893	0.00%	10893	0.00%
20	67	100	10	1792	1792	0.00%	1792.67	0.04%	1792.13	0.01%
20	67	100	100	11893	11893	0.00%	11893	0.00%	11893	0.00%
20	100	10	10	1174	1174	0.00%	1174	0.00%	1174.1	0.01%
20	100	10	100	10215	10215	0.00%	10230.8	0.15%	10215	0.00%
20	100	100	10	1544	1544	0.00%	1544	0.00%	1544	0.00%
20	100	100	100	11101	11102.5	0.01%	11104.3	0.03%	11101	0.00%
25	33	10	10	2185	2185	0.00%	2185	0.00%	2185	0.00%
25	33	10	100	19976	19976	0.00%	19976	0.00%	19976	0.00%
25	33	100	10	2976	2976	0.00%	2976.5	0.02%	2976	0.00%
25	33	100	100	21176	21176	0.00%	21176	0.00%	21176	0.00%
25	67	10	10	2023	2023	0.00%	2023	0.00%	2023	0.00%
25	67	10	100	18251	18251	0.00%	18251	0.00%	18251	0.00%
25	67	100	10	2546	2546	0.00%	2546.9	0.04%	2546	0.00%
25	67	100	100	19207	19207	0.00%	19259.5	0.27%	19207	0.00%
25	100	10	10	1943	1945.4	0.12%	1947.67	0.24%	1943	0.00%
25	100	10	100	17411	17473.8	0.36%	17525.3	0.66%	17411	0.00%
25	100	100	10	2471	2471.7	0.03%	2477.23	0.25%	2471	0.00%
25	100	100	100	18370	18451.2	0.44%	18515.8	0.79%	18373.6	0.02%
30	33	10	10	3205	3205	0.00%	3205	0.00%	3205	0.00%
30	33	10	100	29731	29731	0.00%	29731	0.00%	29731	0.00%
30	33	100	10	4070	4070	0.00%	4070	0.00%	4070	0.00%
30	33	100	100	31077	31077	0.00%	31077	0.00%	31077	0.00%
30	67	10	10	2998	3002.63	0.15%	3005	0.23%	2998	0.00%
30	67	10	100	27581	27637.5	0.20%	27670.5	0.32%	27581	0.00%
30	67	100	10	3649	3649	0.00%	3649.4	0.01%	3649	0.00%
30	67	100	100	28777	28911.3	0.47%	28965.6	0.66%	28777	0.00%
30	100	10	10	2874	2891.77	0.62%	2895.97	0.76%	2874	0.00%
30	100	10	100	26146	26238.4	0.35%	26309.9	0.63%	26149.9	0.01%
30	100	100	10	3483	3483.9	0.03%	3484.93	0.06%	3483	0.00%
30	100	100	100	27314	27467.9	0.56%	27556.3	0.89%	27314	0.00%
35	33	10	10	4474	4475.2	0.03%	4485.77	0.26%	4474	0.00%
35	33	10	100	42305	42319.9	0.04%	42391.8	0.21%	42305	0.00%

35	33	100	10	5423	5423	0.00%	5423	0.00%	5423	0.00%
35	33	100	100	43629	43682.4	0.12%	43850.1	0.51%	43629	0.00%
35	67	10	10	4147	4156.53	0.23%	4165.93	0.46%	4147	0.00%
35	67	10	100	38490	38562.2	0.19%	38744.1	0.66%	38490	0.00%
35	67	100	10	4981	4983.97	0.06%	4994.83	0.28%	4981	0.00%
35	67	100	100	39660	39740.3	0.20%	39877.2	0.55%	39660	0.00%
35	100	10	10	4000	4002	0.05%	4005	0.13%	4000	0.00%
35	100	10	100	36723	36921.9	0.54%	36949.8	0.62%	36723	0.00%
35	100	100	10	4770	4774.17	0.09%	4783.17	0.28%	4770	0.00%
35	100	100	100	38049	38142.2	0.24%	38165.1	0.31%	38049	0.00%
40	33	10	10	5945	5952.6	0.13%	5962.03	0.29%	5945	0.00%
40	33	10	100	56237	56250.3	0.02%	56291.6	0.10%	56237	0.00%
40	33	100	10	6925	6926.37	0.02%	6940	0.22%	6925	0.00%
40	33	100	100	57874	57880.3	0.01%	57913	0.07%	57874	0.00%
40	67	10	10	5567	5590.67	0.43%	5596.63	0.53%	5567.47	0.01%
40	67	10	100	51851	52218.7	0.71%	52401.5	1.06%	51861.2	0.02%
40	67	100	10	6456	6466.6	0.16%	6480.07	0.37%	6456	0.00%
40	67	100	100	53592	53818	0.42%	54024.7	0.81%	53604.6	0.02%
40	100	10	10	5368	5421.67	1.00%	5426.2	1.08%	5368.13	0.00%
40	100	10	100	49817	50375.6	1.12%	50552.2	1.48%	49818.1	0.00%
40	100	100	10	6208	6212.27	0.07%	6226.13	0.29%	6208	0.00%
40	100	100	100	51229	51853.9	1.22%	52038.3	1.58%	51229	0.00%
45	33	10	10	7521	7524.8	0.05%	7552.2	0.41%	7521	0.00%
45	33	10	100	70603	70649.9	0.07%	70790	0.26%	70603	0.00%
45	33	100	10	8720	8720.83	0.01%	8761.6	0.48%	8720	0.00%
45	33	100	100	72676	72755.1	0.11%	72772.3	0.13%	72676	0.00%
45	67	10	10	7161	7219.6	0.82%	7227.4	0.93%	7175.9	0.21%
45	67	10	100	66889	67588.1	1.05%	67808.5	1.37%	66889	0.00%
45	67	100	10	8225	8241.93	0.21%	8260	0.43%	8225	0.00%
45	67	100	100	68737	69304.4	0.83%	69595.5	1.25%	68752.3	0.02%
45	100	10	10	6944	7021.53	1.12%	7016.53	1.04%	6947.67	0.05%
45	100	10	100	64840	65487.6	1.00%	65532.7	1.07%	64859.7	0.03%
45	100	100	10	7827	7830.13	0.04%	7862.57	0.45%	7827	0.00%
45	100	100	100	66508	67378.5	1.31%	67459.2	1.43%	66557.3	0.07%
50	33	10	10	9393	9438.4	0.48%	9477.8	0.90%	9393	0.00%
50	33	10	100	88942	89391.5	0.51%	89674.8	0.82%	88942	0.00%
50	33	100	10	10717	10717.5	0.00%	10722.8	0.05%	10717	0.00%
50	33	100	100	91009	91332.9	0.36%	91633.3	0.69%	91009	0.00%
50	67	10	10	8958	9038.43	0.90%	9102.7	1.62%	8959.5	0.02%
50	67	10	100	84020	85218.4	1.43%	85615.5	1.90%	84027.3	0.01%
50	67	100	10	10100	10118	0.18%	10167.1	0.66%	10100	0.00%
50	67	100	100	86231	87357.6	1.31%	87875.3	1.91%	86234.2	0.00%
50	100	10	10	8713	8826	1.30%	8853.13	1.61%	8717.63	0.05%
50	100	10	100	81858	83136.8	1.56%	83670.7	2.21%	81978.9	0.15%
50	100	100	10	9836	9865.3	0.30%	9892.37	0.57%	9838.57	0.03%
50	100	100	100	83838	84809.1	1.16%	85040.4	1.43%	83882.3	0.05%

Table 2: Min and max costs found by the algorithms for the benchmark instances.

Instance					Transgenetic 1		Transgenetic 2		Tabu Search	
n	d	C	Q	BK	Min	Max	Min	Max	Min	Max
10	33	10	10	350	350	350	350	350	350	350
10	33	10	100	3122	3122	3122	3122	3122	3122	3122
10	33	100	10	646	646	646	646	646	646	646
10	33	100	100	3486	3486	3486	3486	3486	3486	3486
10	67	10	10	255	255	255	255	255	255	255
10	67	10	100	2042	2042	2042	2042	2042	2042	2042
10	67	100	10	488	488	488	488	488	488	488
10	67	100	100	2404	2404	2404	2404	2404	2404	2404
10	100	10	10	239	239	239	239	239	239	239
10	100	10	100	1815	1815	1815	1815	1815	1815	1815
10	100	100	10	426	426	426	426	426	426	426
10	100	100	100	2197	2197	2197	2197	2197	2197	2330
15	33	10	10	745	745	745	745	745	745	750
15	33	10	100	6539	6539	6539	6539	6539	6539	6618
15	33	100	10	1236	1236	1236	1236	1236	1236	1236
15	33	100	100	7245	7245	7245	7245	7245	7245	7245
15	67	10	10	659	659	659	659	663	659	659
15	67	10	100	5573	5573	5573	5573	5648	5573	5573
15	67	100	10	966	966	966	966	966	966	966
15	67	100	100	6188	6188	6188	6188	6200	6188	6200
15	100	10	10	620	620	620	620	628	620	631
15	100	10	100	5184	5184	5184	5184	5211	5184	5211
15	100	100	10	975	975	975	975	975	975	975
15	100	100	100	5879	5879	5879	5879	5879	5879	5888
20	33	10	10	1379	1379	1379	1379	1379	1379	1379
20	33	10	100	12425	12425	12425	12425	12425	12425	12425
20	33	100	10	1972	1972	1972	1972	1972	1972	1972
20	33	100	100	13288	13288	13288	13288	13288	13288	13288
20	67	10	10	1252	1252	1252	1252	1252	1252	1252
20	67	10	100	10893	10893	10893	10893	10893	10893	10893
20	67	100	10	1792	1792	1792	1792	1794	1792	1794
20	67	100	100	11893	11893	11893	11893	11893	11893	11893
20	100	10	10	1174	1174	1174	1174	1174	1174	1177
20	100	10	100	10215	10215	10215	10215	10469	10215	10215
20	100	100	10	1544	1544	1544	1544	1544	1544	1544
20	100	100	100	11101	11101	11114	11101	11153	11101	11101
25	33	10	10	2185	2185	2185	2185	2185	2185	2185
25	33	10	100	19976	19976	19976	19976	19976	19976	19976
25	33	100	10	2976	2976	2976	2976	2991	2976	2976
25	33	100	100	21176	21176	21176	21176	21176	21176	21176
25	67	10	10	2023	2023	2023	2023	2023	2023	2023
25	67	10	100	18251	18251	18251	18251	18251	18251	18251
25	67	100	10	2546	2546	2546	2546	2549	2546	2546
25	67	100	100	19207	19207	19207	19207	19307	19207	19207
25	100	10	10	1943	1943	1948	1945	1948	1943	1943
25	100	10	100	17411	17411	17589	17411	17667	17411	17411
25	100	100	10	2471	2471	2478	2473	2478	2471	2471
25	100	100	100	18370	18370	18545	18370	18609	18370	18479
30	33	10	10	3205	3205	3205	3205	3205	3205	3205
30	33	10	100	29731	29731	29731	29731	29731	29731	29731
30	33	100	10	4070	4070	4070	4070	4070	4070	4070
30	33	100	100	31077	31077	31077	31077	31077	31077	31077

30	67	10	10	2998	2998	3005	3004	3006	2998	2998
30	67	10	100	27581	27581	27677	27581	27743	27581	27581
30	67	100	10	3649	3649	3649	3649	3655	3649	3649
30	67	100	100	28777	28777	29024	28866	29142	28777	28777
30	100	10	10	2874	2880	2902	2886	2906	2874	2874
30	100	10	100	26146	26146	26344	26146	26390	26146	26262
30	100	100	10	3483	3483	3484	3484	3488	3483	3483
30	100	100	100	27314	27314	27678	27385	27720	27314	27314
35	33	10	10	4474	4474	4486	4474	4498	4474	4474
35	33	10	100	42305	42305	42361	42305	42538	42305	42305
35	33	100	10	5423	5423	5423	5423	5423	5423	5423
35	33	100	100	43629	43629	43813	43629	43976	43629	43629
35	67	10	10	4147	4147	4180	4147	4192	4147	4147
35	67	10	100	38490	38490	38817	38524	39167	38490	38490
35	67	100	10	4981	4981	4995	4981	5016	4981	4981
35	67	100	100	39660	39660	39815	39719	40359	39660	39660
35	100	10	10	4000	4000	4011	4000	4017	4000	4000
35	100	10	100	36723	36723	37283	36902	37245	36723	36723
35	100	100	10	4770	4770	4782	4770	4805	4770	4770
35	100	100	100	38049	38049	38171	38152	38266	38049	38049
40	33	10	10	5945	5945	5963	5960	5976	5945	5945
40	33	10	100	56237	56237	56317	56237	56411	56237	56237
40	33	100	10	6925	6925	6939	6925	6968	6925	6925
40	33	100	100	57874	57874	57953	57874	58091	57874	57874
40	67	10	10	5567	5567	5602	5586	5621	5567	5581
40	67	10	100	51851	51911	52524	52092	52763	51851	51911
40	67	100	10	6456	6456	6475	6475	6503	6456	6456
40	67	100	100	53592	53635	54048	53664	54290	53592	53664
40	100	10	10	5368	5372	5455	5391	5450	5368	5372
40	100	10	100	49817	50007	50629	50066	50978	49817	49828
40	100	100	10	6208	6208	6228	6208	6232	6208	6208
40	100	100	100	51229	51229	52170	51506	52479	51229	51229
45	33	10	10	7521	7521	7553	7521	7583	7521	7521
45	33	10	100	70603	70603	70841	70634	71141	70603	70603
45	33	100	10	8720	8720	8742	8720	8795	8720	8720
45	33	100	100	72676	72676	72835	72758	72890	72676	72676
45	67	10	10	7161	7204	7238	7211	7255	7161	7189
45	67	10	100	66889	66889	68010	67159	68212	66889	66889
45	67	100	10	8225	8225	8258	8249	8283	8225	8225
45	67	100	100	68737	68828	69767	69200	69888	68737	68967
45	100	10	10	6944	6955	7063	6955	7071	6944	6955
45	100	10	100	64840	64840	66048	65044	65903	64840	64893
45	100	100	10	7827	7827	7846	7839	7888	7827	7827
45	100	100	100	66508	66761	67878	66789	67877	66508	66743
50	33	10	10	9393	9417	9475	9438	9549	9393	9393
50	33	10	100	88942	88942	89655	89367	90065	88942	88942
50	33	100	10	10717	10717	10722	10717	10742	10717	10717
50	33	100	100	91009	91174	91823	91174	92635	91009	91009
50	67	10	10	8958	8988	9098	9040	9153	8958	8989
50	67	10	100	84020	84217	85741	85043	86294	84020	84064
50	67	100	10	10100	10100	10161	10115	10225	10100	10100
50	67	100	100	86231	86759	87876	87061	88527	86231	86263
50	100	10	10	8713	8766	8865	8804	8904	8713	8752
50	100	10	100	81858	82480	83636	83161	84187	81885	82156

50	100	100	10	9836	9836	9912	9854	9952	9836	9851
50	100	100	100	83838	84083	85295	84725	85623	83838	84054

Table 3: CPU time means in seconds of the algorithms for the benchmark instances.

Instance				Transgenetic 1	Transgenetic 2	Tabu Search
n	d	C	Q	CPU	CPU	CPU
10	33	10	10	0.902	0.373	0.257
10	33	10	100	0.89	0.362	0.262
10	33	100	10	0.861	0.371	0.257
10	33	100	100	0.895	0.357	0.258
10	67	10	10	0.994	0.433	0.528
10	67	10	100	0.986	0.44	0.529
10	67	100	10	0.955	0.419	0.519
10	67	100	100	0.949	0.436	0.512
10	100	10	10	1.061	0.497	0.73
10	100	10	100	1.072	0.497	0.743
10	100	100	10	1.032	0.485	0.738
10	100	100	100	1.052	0.496	0.725
15	33	10	10	1.727	0.573	0.593
15	33	10	100	1.745	0.576	0.587
15	33	100	10	1.701	0.583	0.602
15	33	100	100	1.723	0.588	0.614
15	67	10	10	1.814	0.705	1.242
15	67	10	100	1.866	0.716	1.222
15	67	100	10	1.682	0.668	1.153
15	67	100	100	1.798	0.702	1.19
15	100	10	10	1.94	0.846	1.811
15	100	10	100	1.936	0.835	1.757
15	100	100	10	1.918	0.794	1.75
15	100	100	100	1.93	0.819	1.82
20	33	10	10	2.881	0.923	1.123
20	33	10	100	2.89	0.927	1.111
20	33	100	10	2.822	0.918	1.11
20	33	100	100	2.885	0.915	1.118
20	67	10	10	3.393	1.232	2.194
20	67	10	100	3.442	1.213	2.209
20	67	100	10	3.315	1.184	2.146
20	67	100	100	3.446	1.238	2.2
20	100	10	10	4.196	1.5	3.281
20	100	10	100	4.207	1.509	3.295
20	100	100	10	4.101	1.504	3.24
20	100	100	100	4.192	1.506	3.286
25	33	10	10	4.246	1.296	1.762
25	33	10	100	4.261	1.26	1.78
25	33	100	10	4.094	1.308	1.779
25	33	100	100	4.239	1.277	1.774
25	67	10	10	5.822	1.777	3.663
25	67	10	100	5.85	1.757	3.691

25	67	100	10	5.729	1.728	3.659
25	67	100	100	5.856	1.767	3.76
25	100	10	10	5.876	2.315	5.653
25	100	10	100	5.925	2.367	5.483
25	100	100	10	5.766	2.175	5.595
25	100	100	100	5.862	2.367	5.587
30	33	10	10	6.199	1.789	2.72
30	33	10	100	6.417	1.797	2.733
30	33	100	10	6.283	1.776	2.711
30	33	100	100	6.371	1.791	2.71
30	67	10	10	7.725	2.685	5.49
30	67	10	100	7.726	2.78	5.493
30	67	100	10	7.473	2.665	5.573
30	67	100	100	7.71	2.74	5.475
30	100	10	10	9.069	3.645	8.307
30	100	10	100	9.043	3.591	8.264
30	100	100	10	8.623	3.535	8.172
30	100	100	100	9.033	3.609	8.278
35	33	10	10	9.917	2.442	3.911
35	33	10	100	9.937	2.44	3.842
35	33	100	10	9.947	2.459	3.837
35	33	100	100	9.92	2.369	3.977
35	67	10	10	10.738	3.92	7.963
35	67	10	100	10.676	4.009	7.934
35	67	100	10	10.68	3.816	7.976
35	67	100	100	10.727	4.004	7.978
35	100	10	10	12.584	5.337	11.894
35	100	10	100	12.581	5.328	11.94
35	100	100	10	12.43	5.213	12.168
35	100	100	100	12.599	5.407	11.873
40	33	10	10	12.333	3.509	5.214
40	33	10	100	12.296	3.456	5.144
40	33	100	10	12.03	3.4	5.192
40	33	100	100	12.233	3.479	5.162
40	67	10	10	15.204	5.621	10.64
40	67	10	100	15.087	5.635	10.707
40	67	100	10	14.954	5.508	10.729
40	67	100	100	15.179	5.628	10.735
40	100	10	10	18.841	7.989	16.307
40	100	10	100	18.842	8.012	16.264
40	100	100	10	18.172	7.722	16.066
40	100	100	100	18.808	8.078	16.284
45	33	10	10	16.868	4.53	6.976
45	33	10	100	17.003	4.414	6.956
45	33	100	10	16.836	4.437	7.065
45	33	100	100	17.345	4.364	6.973
45	67	10	10	21.679	7.538	14.411
45	67	10	100	21.809	7.689	14.613
45	67	100	10	21.407	7.452	14.274
45	67	100	100	21.668	7.708	14.462
45	100	10	10	28.468	11.932	22.156
45	100	10	100	28.425	11.971	22.323
45	100	100	10	27.065	11.174	21.523

45	100	100	100	28.258	11.937	22.151
50	33	10	10	21.785	5.965	8.77
50	33	10	100	21.639	5.993	8.833
50	33	100	10	21.383	5.641	8.839
50	33	100	100	21.655	5.97	8.824
50	67	10	10	29.767	10.877	18.677
50	67	10	100	29.762	10.947	18.694
50	67	100	10	29.449	10.951	18.024
50	67	100	100	29.824	10.912	18.597
50	100	10	10	42.597	18.225	28.041
50	100	10	100	42.588	18.457	28.255
50	100	100	10	41.655	17.979	28.431
50	100	100	100	42.868	18.333	28.284

4.4 Analysis

In table 1, we can see that the version 1 of the transgenetic algorithm presented better results than the tabu search for 9 instances up to 20 nodes. For the remaining instances up to 20 nodes, both metaheuristic found the optimal solution. After that, the tabu search continues finding optimal solutions while the transgenetic algorithm deteriorates. The version 2 of the transgenetic is worse than both others for every instance, with its only advantage being that it is faster. A graph with the CPU times of each algorithm can be seen in figure 1. Due to the non uniform nature and small difference of the costs, a graph with the costs from every instance was hard to read.

For every instance, a Mann-Whitney-Wilcoxon test was performed with the mean costs from the 30 executions of the best transgenetic algorithm (version 1) and of the tabu search. Our null hypothesis is that both samples come from the same population. We assumed a 0.05 significance level, meaning that if the p-value is lower than that, we reject the null hypothesis. The p-values for all instances can be seen in table 4. Red values mean the statistical results favored the tabu search approach. No instance had p-value below the level of significance in favor of the transgenetic algorithm. P-values presented in table 4 show that the null hypothesis was rejected for almost every instance of 25 nodes or more, confirming that indeed the algorithms present different behaviours and that the tabu search performed better.

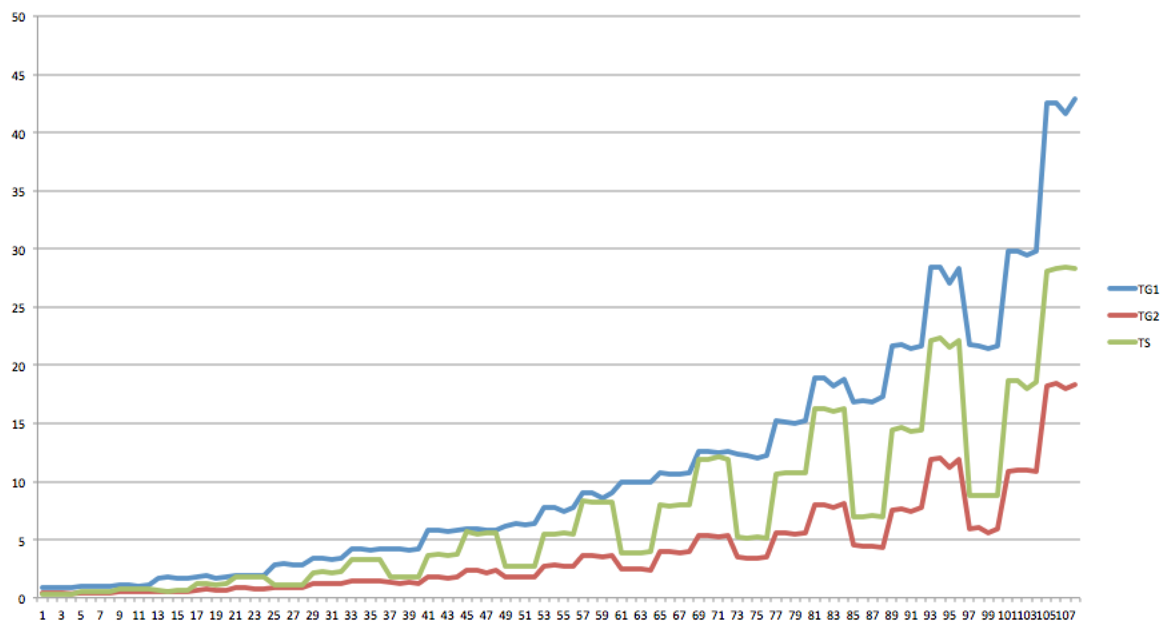

Figure 1: CPU time in seconds for all instances.

Table 4: p-values of Mann-Whitney-Wilcoxon tests for the best transgenetic algorithm (version 1) and the tabu search algorithm, for each benchmark instance. 'NA' means both samples were equal.

Instance				Transgenetic 1 vs Tabu Search
n	d	C	Q	p-value
10	33	10	10	NA
10	33	10	100	NA
10	33	100	10	NA
10	33	100	100	NA
10	67	10	10	NA
10	67	10	100	NA
10	67	100	10	NA
10	67	100	100	NA
10	100	10	10	NA
10	100	10	100	NA
10	100	100	10	NA
10	100	100	100	0.346
15	33	10	10	1
15	33	10	100	0.346
15	33	100	10	NA
15	33	100	100	NA
15	67	10	10	NA
15	67	10	100	NA
15	67	100	10	NA
15	67	100	100	0.000
15	100	10	10	0.072
15	100	10	100	NA
15	100	100	10	NA

15	100	100	100	0.072
20	33	10	10	NA
20	33	10	100	NA
20	33	100	10	NA
20	33	100	100	NA
20	67	10	10	NA
20	67	10	100	NA
20	67	100	10	0.346
20	67	100	100	NA
20	100	10	10	NA
20	100	10	100	NA
20	100	100	10	NA
20	100	100	100	0.048
25	33	10	10	NA
25	33	10	100	NA
25	33	100	10	NA
25	33	100	100	NA
25	67	10	10	NA
25	67	10	100	NA
25	67	100	10	NA
25	67	100	100	NA
25	100	10	10	0.000
25	100	10	100	0.000
25	100	100	10	0.149
25	100	100	100	0.000
30	33	10	10	NA
30	33	10	100	NA
30	33	100	10	NA
30	33	100	100	NA
30	67	10	10	0.000
30	67	10	100	0.000
30	67	100	10	NA
30	67	100	100	0.000
30	100	10	10	0.000
30	100	10	100	0.000
30	100	100	10	0.000
30	100	100	100	0.000
35	33	10	10	0.149
35	33	10	100	0.000
35	33	100	10	NA
35	33	100	100	0.001
35	67	10	10	0.000
35	67	10	100	0.000
35	67	100	10	0.002
35	67	100	100	0.000
35	100	10	10	0.002
35	100	10	100	0.000
35	100	100	10	0.000
35	100	100	100	0.000
40	33	10	10	0.000
40	33	10	100	0.014
40	33	100	10	0.034
40	33	100	100	0.036
40	67	10	10	0.000

40	67	10	100	0.000
40	67	100	10	0.000
40	67	100	100	0.000
40	100	10	10	0.000
40	100	10	100	0.000
40	100	100	10	0.006
40	100	100	100	0.000
45	33	10	10	0.001
45	33	10	100	0.000
45	33	100	10	0.098
45	33	100	100	0.000
45	67	10	10	0.000
45	67	10	100	0.000
45	67	100	10	0.000
45	67	100	100	0.000
45	100	10	10	0.000
45	100	10	100	0.000
45	100	100	10	0.000
45	100	100	100	0.000
50	33	10	10	0.000
50	33	10	100	0.000
50	33	100	10	0.053
50	33	100	100	0.000
50	67	10	10	0.000
50	67	10	100	0.000
50	67	100	10	0.000
50	67	100	100	0.000
50	100	10	10	0.000
50	100	10	100	0.000
50	100	100	10	0.000
50	100	100	100	0.000

These results show that the transgenetic algorithm did not better the results from the tabu search for this problem. These results corroborate what was reported by (MAIA, 2013) and (PINHEIRO, 2016), in which computational experiments showed that a tabu search algorithm also had better results than a transgenetic one for the bi-objective adjacent-only QMST problem. But this research differs from (MAIA, 2013) in three main aspects: the investigated problem is mono-objective; the quadratic nature is stronger than the adjacent only version, and the focus of this research was the investigation of transgenetic algorithms, so several versions of transgenetic agents could be explored and tested. Thus, this work accomplishes its purpose of investigating the application of transgenetic algorithms to the QMST problem.

5 Final considerations

This work designed a transgenetic algorithm for the quadratic minimum spanning tree problem, an NP-hard combinatorial optimization problem. Many metaheuristics had previously been applied to this problem, with tabu search algorithms showing the most promising results. This was the first time the transgenetic algorithm approach was applied to this problem.

The transgenetic algorithm developed used three different transgenetic vectors to manipulate the individuals of the population: plasmis, recombinant plasmid and transposon.

For comparison purposes, the (CORDONE; PASSERI, 2012) tabu search was also implemented. This algorithm presented some of the better solutions for this problem in the literature.

Computational experiments were perfomed using (CORDONE; PASSERI, 2012) benchmark instances. The results show that the transgenetic algorithm did not present better solutions than the tabu search.

For future work, we could try improving the results by executing parameter tests with the irace, including heuristic solutions in the population and testing other mechanisms of combining the agents. Computational experiments could be done for larger instances with more than 50 nodes. The behaviour of transgenetic algorithms compared to other evolutionary approaches to QMST should also be investigated. Finally, it could be interesting to explore a hybrid of transgenetic and tabu search algorithms, with potential for better solutions.

References

ASSAD, A.; XU, W. The quadratic minimum spanning tree problem. *Naval Research Logistics*, Wiley-Blackwell, v. 39, n. 3, p. 399–417, apr 1992.

CORDONE, R.; PASSERI, G. Solving the quadratic minimum spanning tree problem. *Applied Mathematics and Computation*, v. 218, n. 23, p. 11597 – 11612, 2012. ISSN 0096-3003.

FU, Z.-H.; HAO, J.-K. A three-phase search approach for the quadratic minimum spanning tree problem. *Engineering Applications of Artificial Intelligence*, v. 46, p. 113 – 130, 2015. ISSN 0952-1976.

GAO, J.; LU, M. Fuzzy quadratic minimum spanning tree problem. *Appl. Math. Comput.*, Elsevier Science Inc., New York, NY, USA, v. 164, n. 3, p. 773–788, maio 2005. ISSN 0096-3003.

GOLDBARG, E. F. G.; GOLDBARG, M. C. Transgenetic algorithm: A new endosymbiotic approach for evolutionary algorithms. In: *Foundations of Computational Intelligence Volume 3*. [S.l.]: Springer Berlin Heidelberg, 2009. p. 425–460.

KRUSKAL, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, American Mathematical Society (AMS), v. 7, n. 1, p. 48–48, jan 1956.

LOZANO, M. et al. Tabu search with strategic oscillation for the quadratic minimum spanning tree. *IIE Transactions*, Taylor Francis, v. 46, n. 4, p. 414–428, 2014.

MAIA, S. *O Problema Biobjetivo da Arvore Geradora Quadratica em Adjacencia de Arestas.* Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, 2013.

MAIA, S. M.; GOLDBARG, E. F.; GOLDBARG, M. C. On the biobjective adjacent only quadratic spanning tree problem. *Electronic Notes in Discrete Mathematics*, v. 41, p. 535 – 542, 2013. ISSN 1571-0653.

ÖNCAN, T.; PUNNEN, A. P. The quadratic minimum spanning tree problem: A lower bounding procedure and an efficient search algorithm. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 37, n. 10, p. 1762–1773, out. 2010. ISSN 0305-0548.

PALUBECKIS, G.; RUBLIAUSKAS, D.; TARGAMADZė, A. Metaheuristic approaches for the quadratic minimum spanning tree problem. *Information Technology and Control*, v. 39, p. 257–268, 01 2010.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial Optimization: Algorithms and Complexity.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982. ISBN 0-13-152462-3.

PEREIRA, D. L.; GENDREAU, M.; CUNHA, A. Lower bounds and exact algorithms for the quadratic minimum spanning tree problem. v. 63, 05 2015.

PINHEIRO, L. *Algoritmos Experimentais para o Problema Biobjetivo da Árvore Geradora Quadrática em Adjacência de Arestas.* Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, 2016.

PRIM, R. C. Shortest connection networks and some generalizations. *Bell System Technical Journal*, Institute of Electrical and Electronics Engineers (IEEE), v. 36, n. 6, p. 1389–1401, nov 1957.

ROCHA, D. A. M.; GOLDBARG, E. F. G.; GOLDBARG, M. C. A new evolutionary algorithm for the bi-objective minimum spanning tree. In: *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*. [S.l.]: IEEE, 2007.

SOAK, S.-M.; CORNE, D. W.; AHN, B.-H. The edge-window-decoder representation for tree-based problems. *IEEE Transactions on Evolutionary Computation*, v. 10, n. 2, p. 124–144, April 2006. ISSN 1089-778X.

SUNDAR, S.; SINGH, A. A swarm intelligence approach to the quadratic minimum spanning tree problem. *Information Sciences*, v. 180, n. 17, p. 3182 – 3191, 2010. ISSN 0020-0255.

TALBI, E.-G. *Metaheuristics: From Design to Implementation.* [S.l.]: Wiley Publishing, 2009. ISBN 0470278587, 9780470278581.

VIKHAR, P. A. Evolutionary algorithms: A critical review and its future prospects. In: *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. [S.l.]: IEEE, 2016.

WILLIAMSON, D. P.; SHMOYS, D. B. *The Design of Approximation Algorithms.* [S.l.]: Cambridge University Press, 2011. ISBN 0521195276.

ZHOU, G.; GEN, M. An effective genetic algorithm approach to the quadratic minimum spanning tree problem. *Computers Operations Research*, v. 25, n. 3, p. 229 – 237, 1998. ISSN 0305-0548.

ZHOU, J.; HE, X.; WANG, K. Uncertain quadratic minimum spanning tree problem. v. 9, p. 385–390, 05 2014.