UNIVERSIDADE FEDERAL FLUMINENSE

EDCARLLOS GONÇALVES DOS SANTOS

ON TWO LOGISTIC PROBLEMS OF SMART CITIES

NITERÓI

2019

UNIVERSIDADE FEDERAL FLUMINENSE

EDCARLLOS GONÇALVES DOS SANTOS

ON TWO LOGISTIC PROBLEMS OF SMART CITIES

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação.
Área de concentração: ALGORITMOS E OTIMIZAÇÃO

Orientador:
LUIZ SATORU OCHI

Co-orientador:
LUIDI GELABERT SIMONETTI

NITERÓI

2019

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S237o Santos, Edcarllos Gonçalves dos
On Two Logistic Problems of Smart Cities / Edcarllos Gonçalves dos Santos ; Luiz Satoru Ochi, orientador ; Luidi Gelabert Simonetti, coorientador. Niterói, 2019.
71 f. : il.

Tese (doutorado)-Universidade Federal Fluminense, Niterói, 2019.

DOI: http://dx.doi.org/10.22409/PGC.2019.d.09783480600

1. Metaheurística. 2. Roteamento. 3. Programação dinâmica. 4. Produção intelectual. I. Ochi, Luiz Satoru, orientador. II. Simonetti, Luidi Gelabert, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

EDCARLLOS GONÇALVES DOS SANTOS

ON TWO LOGISTIC PROBLEMS OF SMART CITIES

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação.
Área de concentração: ALGORITMOS E OTIMIZAÇÃO

Aprovada em AGOSTO de 2019.

BANCA EXAMINADORA

Prof. LUIZ SATORU OCHI - Orientador, UFF

Prof. LUIDI GELABERT SIMONETTI - Coorientador, UFRJ

Véverton dos Santos Souza

Prof. UEVERTON DOS SANTOS SOUZA, UFF

Yuri Abitbol de M. Frota

Prof. YURI ABITBOL DE MENEZES FROTA, UFF

Prof. ALFREDO CANDIA VÉJAR, Universidad de Talca

Igor Machado Coelho

Prof. IGOR MACHADO COELHO, UERJ

Niterói

2019

Agradecimentos

Mesmo as palavras sendo insuficientes para expressar a gratidão que tenho a todos que contribuiram durante esta etapa, gostaria de agradecer a minha família pelos conselhos, compreensão e apoio emocional que, de alguma forma, me fizeram acreditar que poderia alcançar os meus objetivos de vida. Aos meus pais Edgar e Luciene e irmãs Karlla e Gildeene.

Ao amor da minha vida, Mariana, por sempre me apoiar e incentivar durante essa longa jornada acadêmica. Espero compartilhar uma vida inteira contigo.

Aos meus orientadores, Luiz Satoru e Luidi Simonetti, não somente pela orientação ou pelo suporte, mas também pelos ensinamentos que não se limitaram somente ao meio acadêmico. Saibam que a confiança de vocês foi essencial.

Aos irmãos da *OptHouse*: Alan, Heder, Igor, Jânio, João, Marcos Guerine, Marcos Melo, Matheus Haddad, Pablo, Paulo, Schneider e Uéverton. Agradeço não somente pelo incrível acolhimento e aprendizado que adquiri durante todos esses anos, mas também pela convivência e pelos momentos de descontração.

Aos muitos amigos que fiz durante essa etapa. Se me senti bem acolhido nesse lugar com certeza foi por causa de cada um de vocês.

Aos demais professores IC/UFF e também aos membros da banca pelos comentários e contribuições que tornaram possível a concretização deste trabalho.

Ao CNPQ e a CAPES pelo apoio financeiro, sem o qual seria difícil conduzir o trabalho.

Muito obrigado a todos!

Resumo

Problemas de otimização são encontrados em diversos setores da produção industrial, buscando em geral minimizar os custos e maximizar os lucros. No contexto das Cidades Inteligentes, se torna prioridade a transparência para o cidadão e uma logística efetiva de transporte de pessoas e bens de consumo, motivando uma série de propostas acadêmicas para o tema. O estudo desses problemas é essencial para se manter a competitividade de cadeias produtivas, também sendo um desafio a tarefa de encontrar uma solução de boa qualidade em um tempo computacional baixo. De forma específica, a logística têm sido bastante estudada devido a seu grande número de aplicações práticas e, neste sentido, este trabalho explora o estudo de dois problemas de transportes. O Problema de Caminho com Coleta de Prêmios, consiste em encontrar um (s, t)-caminho que minimiza a soma dos pesos de suas arestas (tempo total de transporte) menos o prêmio total dos nós em tal caminho. Por sua vez, para o Problema de Roteamento de Veículos para o Transporte de Funcionários, deve-se minimizar os custos de transporte e, respeitar em contrapartida, as restrições de qualidade de serviço para os funcionários. Ambos os problemas estão presentes no núcleo de diversas aplicações relevantes em áreas como Telecomunicações, Transporte Público e Manutenção de Equipamentos. Neste trabalho serão exploradas diferentes técnicas de resolução que vão desde métodos exatos até heurísticas inteligentes.

Palavras-chave: Coleta de Prêmios; Roteamento de Veículos; *Treewidth*.

Abstract

Optimization problems are found in different sectors of industrial production where it is desired to minimize costs and maximize profits. In the context of Smart Cities, transparency for citizens, and effective logistics for the transportation of people and consumer goods becomes a priority, motivating a series of academic proposals on the subject. The study of these problems is essential for maintaining the competitiveness of supply chains; however, the task of finding a good quality solution in a short computational time is challenging. At this point, logistics have been extensively studied due to their large number of practical applications and, in this sense, this work explores the study of two transportation problems. The Prize-Collecting Path Problem consists of finding a $(s,\, t)$-path that minimizes the sum of its edge weights (total transportation time) minus the total prize collected on nodes of that path. In turn, for the Vehicle Routing Problem for Transportation of Employees, the overall costs should be minimized according to the quality of service constraints defined for the employees. Both problems are present at the core of several relevant applications in areas such as Telecommunications, Public Transportation, and Equipment Maintenance. In this work, different resolution techniques will be explored, ranging from exact methods to intelligent heuristics.

Keywords: Prize-collecting; Vehicle Routing Problem; Treewidth.

List of Figures

List of Tables

Nomenclature

HP	:	Hamiltonian Path;
PCP	:	Prize Collecting Path;
PCST	:	Prize Collecting Steiner Tree;
SBRP	:	School Bus Routing Problem;
SPAW	:	Shortest Path with Arbitrary Weights;
VRPTE	:	Vehicle Routing Problem for Transportation of Employees;

Contents

Chapter 1

Introduction

In recent years, the theme of Smart Cities has been developed actively and often is treated as a multidisciplinary topic. At this point, the area gathers researchers from Humanities, Urban Planning, Transportation Engineering, and Computer topics such as the Internet of Things (IoT), Computational Intelligence for Decision Making, and Multi-agent systems. A smart city can be defined as a city that incorporates information and communication technologies to enhance the quality and performance of public services. In general, these services are present in important sectors like transportation, governance, security, communication, energy, and sustainability, which, in turn, represent a major expense for the economy. In transportation, for example, logistics costs corresponded to about 12.6% of Brazil's GDP in 2016, according to data from ILOS (Instituto de Logística e Supply Chain). The study also concluded that the transportation costs related to cargo freight represented between half and two-thirds of the overall logistics costs. In order to minimize these operational costs, the application of intelligent techniques has been used to solve such problems. In contrast, another market trend in the industry is to maximize the profits by establishing an appropriate level of service for customers and employees, also known as Quality of Service (QoS).

In the context of public transportation, there are also several issues to note, such as long waiting times, poor road quality, or low-quality service. In [48], some data on school transportation in Santa Catarina were collected over the period 2001-2011, and it was observed that the average time spent by students traveling between home and school increased from 25.1% to 36.7% for men and 18.8% to 29.2% for women. Other studies confirm the problems related to transportation in large cities; in [45], the results indicated that the average speed of traffic in the city of São Paulo in rush hours was only 19.30 km/h. Moreover, the average time spent daily by residents in traffic is about

2h42min, which represents an average of 27 days a year trapped in urban traffic. To work around these problems, some solutions may be adopted: for the citizens, reducing car dependency or adoption of mechanisms of carpooling; for the government, the increase of political support for sustainable transport or the optimization of public transport in suburban areas.

About transportation systems, another current trend to solve these problems is the combination of logistics with computational intelligence. In this sense, several transportation problems in Smart Cities have been addressed, such as variations of the classic Shortest Path Problem (SPP), as well as Vehicle Routing Problems (VRPs). Both problems are typical themes in combinatorial optimization and have been extensively studied in the last 50 years [31]. For the VRP, a fleet of vehicles located in a depot is selected to compose a set of routes that meets the demand of a geographically distributed set of customers in order to minimize specific criteria such as time, distance or operating costs.

The class of problems involving vehicle routing has many practical applications, being the product distribution the best-known case. However, PRV's can also be found in the context of collecting (waste management; agricultural supply chains; mining) and in the service sector (gas or water meter inspection; periodic maintenance of elevators and fire extinguishers). The distribution network does not necessarily have to be made up of roads or streets. It may consist of railways, power lines or rivers, that is, where there is a set of moving objects (trucks, trains, boats, and even pedestrians) who must visit a set of locations. All of these examples illustrate the broad applicability and economic importance of VRP.

In this work, two logistics problems in smart cities are discussed. The first one is the Prize-collecting Path Problem (PCP), a variation of the SPP, which is relevant in the sense of the transportation of vital resources, confidential data, and troops. The last one is the Vehicle Routing Problem for Transportation of Employees (VRPTE), a practical application of a logistic problem for an energy industry in Brazil.

1.1 Objective

The main objective of this work is to present relevant contributions to solve logistic problems in the area of smart cities. Besides, the specific objectives for the PCP are:

- A NP-completeness proof for the problem;

- Development of a study of the complexity for specific graph classes;
- Presentation of an FPT-algorithm for bounded treewidth graphs based on a dynamic programming.

On the other hand, the particular objectives for the VRPTE are the following:

- A concisely introduction for a new practical problem in the logistics field;
- Development of an efficient heuristic to solve the problem in a reasonable time;
- Formulation of a nonlinear mathematical model to validate the problem;
- Creation of a module to generate instances according to the characteristics of the real scenario;
- Development of a novel visualization system based on the integration of the routing module with the OpenStreetMaps.

Chapter 2

Prize Collecting Path Problem

Recent world events such as terrorist attacks and natural disasters have demonstrated the need to consider vulnerabilities which disrupt network infrastructures. This kind of infrastructure is typically used to lead electricity, water, troops, or vital resources to specific locations, so the reliability of the system is essential to prevent or reduce further damage in the network. In this way, the Network Interdiction problem can be thought of as a competition of two agents, a leader, and a follower. The leader has a fixed budget that is used to deteriorate critical parts of the network; an action called an interdiction. In turn, the follower agent solves an optimization problem in the interdicted network. In this work, the Prize Collecting Path (PCP), a lower-level problem of an interdiction bilevel network, is introduced, and some theoretical aspects, as well as, optimization techniques are discussed.

2.1 Introduction

Let $G = (V, A)$ be a directed and connected graph, where V is the set of nodes, and A is the set of arcs. We assume that there exist n nodes and m arcs. Associated with the set of nodes there is a prize function $\mathbf{p} : V \to \mathbb{R}_{>0}$. Likewise, associated with the set of arcs, there is a transportation time function $\mathbf{t} : A \to \mathbb{R}_{>0}$. Node $s \in V$ and $t \in V$ correspond, respectively, to source and target node. Let $\mathcal{P}_{st}$ be the set of all (s, t)-paths in G connecting s and t. The Prize Collecting Path problem consists on finding a (s, t)-path that minimizes the total transportation time cost minus the total prize of the nodes belonging to the (s, t)-path. PCP can also be defined as the following 0-1 integer programming problem:

$$\min \quad \sum_{(ij)\in A} t_{ij}x_{ij} - \sum_{j\in V} p_j z_j \tag{2.1}$$

$$\text{s.t} \quad \mathbf{x} \in \mathcal{P}_{st}, \qquad 0 \le x_{ij} \le 1 \tag{2.2}$$

Variables are represented by the binary vectors $\mathbf{x} \in \{0,1\}^{|A|}$ and $\mathbf{z} \in \{0,1\}^{|V|}$; such that $x_{ij} = 1$ if arc $(ij) \in A$ is used by a $(s,\, t)$-path and $x_{ij} = 0$ otherwise, and $z_i = 1$ if node $i \in V$ is visited by a $(s,\, t)$-path, and $z_i = 0$ otherwise. A feasible $(s,\, t)$-path is induced by a vector $\mathbf{x}$ that belongs to the following set:

$$\mathcal{P}_{st} = \left\{ \mathbf{x} \in \{0,1\}^{|A|} \;\middle|\; \begin{array}{c} \displaystyle\sum_{(sj)\in\delta^+(s)} x_{sj} = 1 \\ \displaystyle\sum_{(jh)\in\delta^+(t)} x_{jh} - \sum_{(kj)\in\delta^-(j)} x_{kj} = 0, \forall j \in V \setminus \{s,t\} \\ \displaystyle\sum_{(jt)\in\delta^-(t)} x_{jt} = 1 \end{array} \right\} \tag{2.3}$$

For a given vector $\mathbf{x} \in \mathcal{P}_{st}$, variables $\mathbf{z}$ are related as follows:

$$z_j \le \sum_{(ij)\in\delta^-(j)} x_{ij}, \forall j \in V \setminus \{s,t\} \tag{2.4}$$

$$z_s \ge 1 \tag{2.5}$$

$$z_t \ge 1 \tag{2.6}$$

A natural extension of the *Steiner Tree* problem is closely related to PCP. Given an undirected and connected graph with prizes associated to the set of nodes and weights associated to the set of edges, the Prize Collecting Steiner Tree problem (PCST) consists of finding a subtree which minimizes the sum of the weights of its edges plus the prizes of the nodes not spanned. Although the PCST have been introduced by [4], a variation was firstly considered by [47]. Besides, the PCP objective function (Equation 2.1) is the minimization version of the objective function of Net-Worth problem presented in [25].

Optimization techniques to solve PCST consists of approximation algorithms and hybrid heuristics combined with reduction tests and preprocessing procedures. A 2-approximation algorithm was proposed by [20] and some improvements for the approximation ratio and running time can be seen in [25]. In [8], a multi-start local search heuristic that consists on generate initial solutions through a primal-dual approximation algorithm was developed. A Path relinking was used to improve solutions in the local search phase, and a heuristic based on Variable Neighborhood Search was utilized as a post-optimization procedure. An integer programming formulation using cutting planes algorithms to obtain lower bounds was described in [37]. To identify vertices and edges that are guaranteed not to be in any optimal solution, some reduction tests were developed.

In [10], seven variations of PCSTP were presented. Four of these were shown to be solvable in $\mathrm{O}\big(m+log\ n\big)$ time and for one case was provided a $\mathrm{O}\big(m\big)$ algorithm which combines existent techniques of other optimization problems. Also, some reduction tests were proposed to reduce the instance input size. Exacts approaches include Branch-and-cut algorithms for a PCSTP with budget and quota constraints [55] and a Non-Delayed Relax-and-Cut algorithm that use Dual Lagrangian information of feasible solutions provided by a Lagrangian heuristic [11]. Recently, a matheuristic based on clustering algorithm and preprocessing procedures was proposed by [1]. Finally, some results for parameterized complexity to different variants of Steiner Tree problems are presented in [26, 18, 40].

In this work, the complexity behavior of the problem is analyzed. For some cases is proved that PCP is NP-complete, these results lead to the generation of new sets of benchmark instances that are computationally hard according to natural characteristics of the problem. Besides, a polynomial-time algorithm is described for bounded treewidth graphs, and a mathematical formulation is introduced to solve general instances of PCP.

2.2 NP-completeness

Definition 1 *Given a graph G, a $(s,\ t)$-hamiltonian path is a simple path between two nodes (source and target) that visits each node exactly once. The $(s,\ t)$-Hamiltonian Path problem (HP) determines whether a given graph contains a Hamiltonian path starting in s and finishing in t.*

Theorem 2.2.1 *HP $\propto$ PCP.*

Proof. Given a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges. Let nodes $\{s, t\} \in V$, represents the source and target nodes, respectively. We construct an instance G' of PCP as follows: (i) set $G' = G$; (ii) for each edge e of G' we associate a transportation time t_e of value 1; and (iii) for each node v of G' we assign a prize p_v of value 2 ($p_s = p_t = 0$).

($\Rightarrow$) Since Hamiltonian path visits all nodes of a graph, all possible (s, t)-hamiltonian paths in G is composed by $|V|$ nodes and $|V| - 1$ edges, such way in G' the sum of transportation time for edges of these paths is equal $|V| - 1$ and the sum of prizes for nodes is $2 \cdot (|V| - 2)$. Thus such (s, t)-hamiltonian paths have cost according to Equation 2.1 equal to $-|V| + 3$ in G'.

($\Leftarrow$) Let $p \in \mathcal{P}_{st}$ be a solution of PCP with $-|V| + 3$ cost in G' and $I_p = V(p) \backslash \{s, t\}$. Since all paths in $\mathcal{P}_{st}$ are simple path, *i.e.*, each node is visited just once. For each node $v \in I_p$, there is exactly one incoming edge $e_i \in E$ used in p. As any node $v \in I_p$ has prize $p_v = 2$, then each node v contribute with $(t_{e_i} - p_v) = -1$ in PCP objective function. Hence p has cost $-|I_p| + 1$, once t has no prize associated. Consequently $-|I_p| + 1 = -|V| + 3$ and $|I_p| = |V| - 2$ which implies that path p is hamiltonian in G. ∎

Corollary 2.2.2 *PCP is NP-hard.*

Proof. Follows from Theorem 2.2.1 and the fact that Hamiltonian Path problem is NP-complete [28].

2.3 Tractability

2.3.1 Shortest Path with Arbitrary Weights

Definition 2 *Given an acyclic digraph $D = (V, A)$ where V is the set of nodes and E is the set of arcs. Let nodes $\{s, t\} \in V$, represents the source and target nodes, respectively. For each arc $a \in A$ there is an arbitrary weight w_a (not necessarily higher than zero). The Shortest Path with Arbitrary Weights problem (SPAW) is about to determine the shortest simple path from s to t in G.*

Theorem 2.3.1 *PCP $\propto$ SPAW.*

Proof. Given a graph $G = (V, E)$ where $V(G)$ is the set of nodes and $E(G)$ is the set of edges. Let $\{s, t\} \in V(G)$ be nodes representing the source and target vertices, respectively.

Each edge $e \in E(G)$ has an associated transportation time t_e, and every node $v \in V(G)$ has an assigned prize p_v, where $p_s = p_t = 0$.

From G we construct an instance G' of SPAW as follows: (i) set $G' = G$; (ii) each undirected edge $e = (u, v) \in E(G)$ is converted in two arcs $a_1 = (u, v), a_2 = (v, u) \in E(G')$ of opposite ways, where the weights w_{a_i} of a_i ($i \in \{1, 2\}$) is defined by $(t_e - p_{h(e)})$, where $p_{h(e)}$ is the prize assigned to the head node of a_i.

($\Rightarrow$) Let $p = s, v_1, v_2, \ldots, v_k, t \in \mathcal{P}_{st}$ be a solution of PCP in G. The cost for path p in G, according to Equation 2.1, is

$$\sum_{e \in E(p)} t_e - \sum_{v \in V(p)} p_v.$$

In G', the cost of p is defined by

$$\sum_{a \in A(p)} w_a$$

that is equivalent to

$$\sum_{e \in E(p)} (t_e - p_{h(e)})$$

which, by additivity, is equal to

$$\sum_{e \in E(p)} t_e - \sum_{e \in E(p)} p_{h(e)}.$$

As source and target nodes have null prizes, then p has the same cost in both problems.

($\Leftarrow$) By construction, any (s, t)-path p of G' is also an (s, t)-path of G, and as shown previously, p has the same cost in both instances. Figure 2.1 illustrates the entire process for a generic graph. ∎

By Theorem 2.2.1 and Theorem 2.3.1 HP $\propto$ PCP $\propto$ SPAW. From that, it is possible to get a mapping of the complexity of the PCP problem, providing sufficient conditions for the problem becomes polynomial or NP-hard. Figure 2.2 illustrates the relation of complexity between the problems. More precisely, Theorem 2.2.1 and Theorem 2.3.1 implies the following corollaries.

Corollary 2.3.2 *For any graph class $\mathcal{C}$ such that HP on $\mathcal{C}$ is NP-hard, the PCP problem on $\mathcal{C}$ is also NP-hard.*

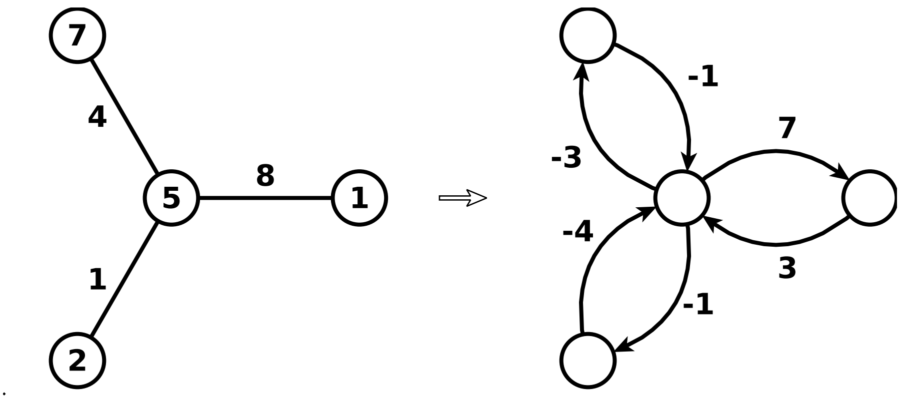

Figure 2.1: Example of transformation between PCP and SPAW instances

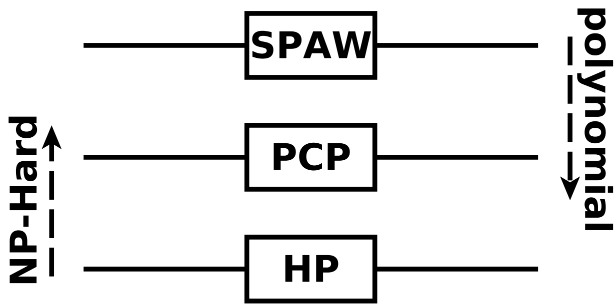

Figure 2.2: Mapping of complexity for PCP problem

Corollary 2.3.3 *Any instance $\mathcal{I}$ of PCP can be solved in polynomial time whether the instance g($\mathcal{I}$) of SPAW can be solved in polynomial time, where g returns a digraph constructed as described in the proof of Theorem 2.3.1.*

Corollary 2.3.4 *PCP can be solved in polynomial time when the input G does not contain cycles $C = v_1, v_2, \ldots, v_k$ such that $\sum(t_e - p_{v_j}) < 0$ for any $e = v_i, v_j$, $1 \leq i \leq k$, $j = ((i+1)$* mod $(k+1))$.

Proof. Follows from Lemma 2.3.3 and the result for the minimum path of graphs with no negative cycles.

Now, we study the complexity of the problem for particular graph classes.

2.3.2 Grid graphs

Given that Minimum Path, Hamiltonian Path, and Longest Path are polynomial on Rectangular Grid graphs [14, 29]. We will analyze the complexity of PCP on Rectangular Grid graphs.

Definition 3 *[24] Let G^∞ be the infinite graph whose vertex set consists of all points of the plane with integer coordinates and in which two vertices are connected if and only if the Euclidean distance between them is equal 1. A Grid graph is a node-induced finite subgraph of the infinite grid. It is a rectangular grid if its set of nodes is the product of two intervals.*

Lemma 2.3.5 [24] *Hamiltonian Path problem on Grid graphs is NP-complete.*

Corollary 2.3.6 *(s,t)-Longest Path problem on Grid graphs is NP-complete.*

Theorem 2.3.7 *PCP on Rectangular Grid graphs is NP-complete*

Proof. This proof uses a reduction from (s,t)-Longest Path problem on Grid graphs. Let G be an instance of (s,t)-Longest Path. As shown in [24] this problem remains NP-hard even when a grid embedding of G is known. Given such embedding, we can recognize in polynomial time the set V' of vertices and the set E' of edges to add in G in order to make it a rectangular grid.

Set $H = (V \cup V', E \cup E')$. Adding time equal to 1 for every edge in E, prize equal to 2 for each node in V, prize 0 for any node in V' and time equal to $|E|+1$ for any edge in E', we obtain a rectangular graph H such that G has an (s,t)-longest path of size k if and only if H has a prize collecting path of cost $-k$. Figure 2.3 gives an example of the process for an arbitrary graph. ∎

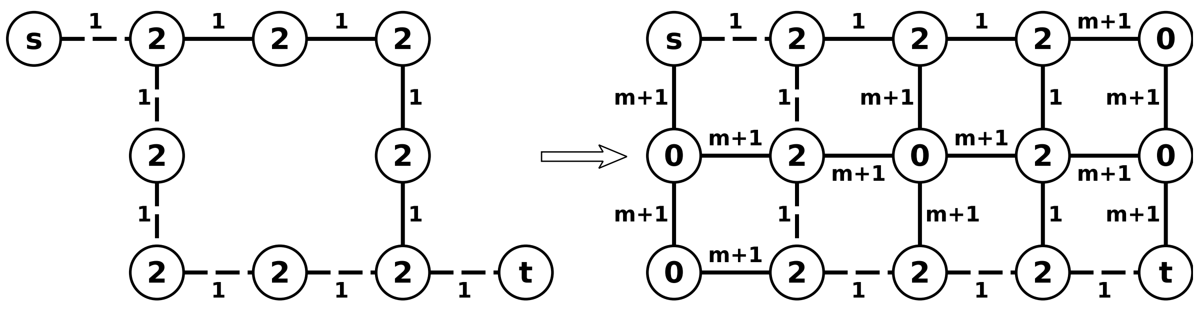

Figure 2.3: An example of NP-completeness for grid graphs

2.4 An Algorithm for Graphs with Bounded Treewidth

The solution for many real problems frequently requires an algorithmic approach. Sometimes both input data and the frequency of access are significant; therefore, efficient algorithms are required (generally of polynomial time). The NP-completeness theory was developed to determine which problems probably cannot be solved by polynomial algorithms. However, since many NP-hard problems need to be solved in practice, one possibility is to explore approximation algorithms or heuristics instead of exact algorithms.

A recent alternative for the tractability of these problems is to resort to analysis through the Parametrized Complexity theory. This theory studies the existence of algorithms whose exponential complexity depends only on specific aspects of the input data; such algorithms are called Fixed-Parameter Tractable (FPT). Formally, given an NP-hard problem π and a parameter k of the problem, π is treatable by a fixed parameter regarding k, if the problem can be solved at execution time $f(k) \cdot n^{O(1)}$, where f is an arbitrary function. The complexity class corresponding to these problems is defined as FPT. Note that the parameter is an input data that is isolated for further analysis. Since the analysis and development of FPT-algorithms are dependent on the parameter in question, the researcher needs to identify parameters that make sense in practice.

In the next sections, a dynamic programming FPT-algorithm that take advantage of a tree structure is presented, to solve the PCP in an effective way for some graph classes. At first, the notion of tree-likeness is introduced using approaches of graph decomposition and treewidth. Moreover, the algorithm is formally described, and a practical example is detailed to improve the understanding of the concepts.

2.4.1 Bounded Treewidth graphs

Trees are one of the most useful classes of graphs. A tree is defined as a connected acyclic graph, *i.e.*, a graph which any pair of vertices are connected strictly by one path. This kind of structure is seen in several areas such as molecular evolution, computer science, and the study of electrical circuits. Specifically, in computer science, trees are often used because its simple design enables the development of efficient algorithms. In fact, several NP-hard problems are polynomially solvable on trees (*e.g.* colourability, independent set [19], hamiltonian path [23], maximum cut [5] and graph isomorphism [17]).

There are distinct ways to determine the tree-likeness of a graph. Some parameters trying to measure this aspect were proposed; most of them include the number of cycles, the amount of vertices removal to turn a graph into acyclic or the bounded-size parts connected in a tree-like way. Nevertheless, these indicators are not sufficient to provide a satisfactory abstraction of the concept of tree-likeness, as can be seen in Figure 2.4. In the next sections, a new approach to deal with tree-like nature on graphs is introduced, the so-called treewidth.

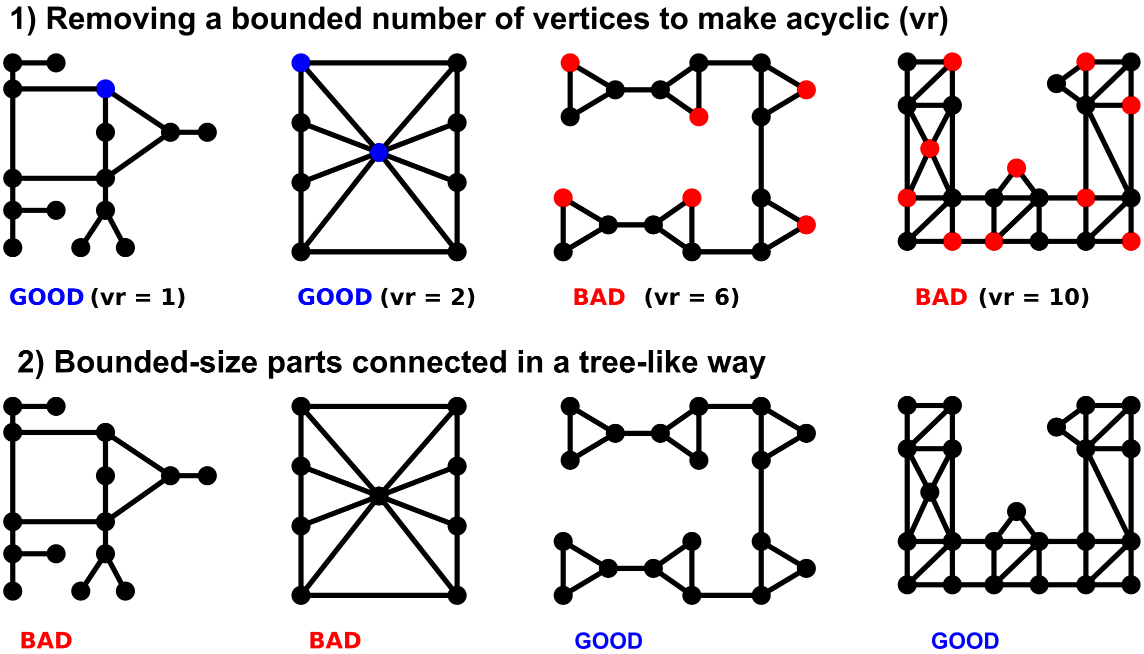

Figure 2.4: Tree-likeness metrics with different results.

2.4.1.1 Graph decomposition and treewidth

Graph decomposition is a quite popular technique in structural graph theory and has been the subject of research for the solution of several optimization problems. A decomposition $H = \{X_1, X_2, \ldots, X_I\}$ of a graph $G = (V, E)$ is a collection of I edge-disjoint subgraphs, such that every edge $e \in E$ belongs to at least one subgraph X_i. For simplicity and convenience, in this work, each subgraph in the decomposition will be called a bag. Decomposition can be defined as a path decomposition if the following conditions hold:

1. (node coverage) $\bigcup_{i=1}^{I} X_i = V$, *i.e.*, each vertex belongs to at least one bag;
2. (edge coverage) for every edge $uv \in E$, there is at least one bag X_i that contains u and v;
3. (coherence) for $1 \le i \le i' \le i'' \le I$, $X_i \cap X_{i''} \subseteq X_{i'}$, *i.e.*, all bags including any vertex v form a contiguous subsequence of the whole sequence.

In this way, coverage conditions 1 and 2 ensures that all nodes and edges of G are present in the decomposition graph and, moreover, as a consequence of the coherence condition, the subgraph connecting all bags that include any vertex v must be a connected path. Through these statements, the width w of a path decomposition can be defined as the size of the largest bag minus one[1], that is, $w(H) = \max_{1 \le i \le I} |X_i| - 1$. In this context, a pathwidth pw of a graph is the minimum width between all possible path decompositions.

The pathwidth can also be seen as the vertex separation number of a graph [30]. In fact, the aspect of separation is strictly related to pathwidth. Given two consecutive bags X_i and X_{i+1} and the set $S_{i,i+1} = X_i \cap X_{i+1}$, also known as separator, two connected components $C_i = \bigcup_{c=1}^{i} X_c$ and $C_{i+1} = \bigcup_{c=i+1}^{I} X_c$ are created, by splitting the vertices in V. On the whole, every bag X_i separates vertices in the bags before i with the vertices of the bags following after i. It is worthwhile to mention that the order of a separation is defined as $|S_{i,i+1}|$ and that any path from C_i to C_{i+1}, must include at least one vertex of the separator set $S_{i,i+1}$. In this way, a path decomposition of width w is a sequence of separations of order at most w.

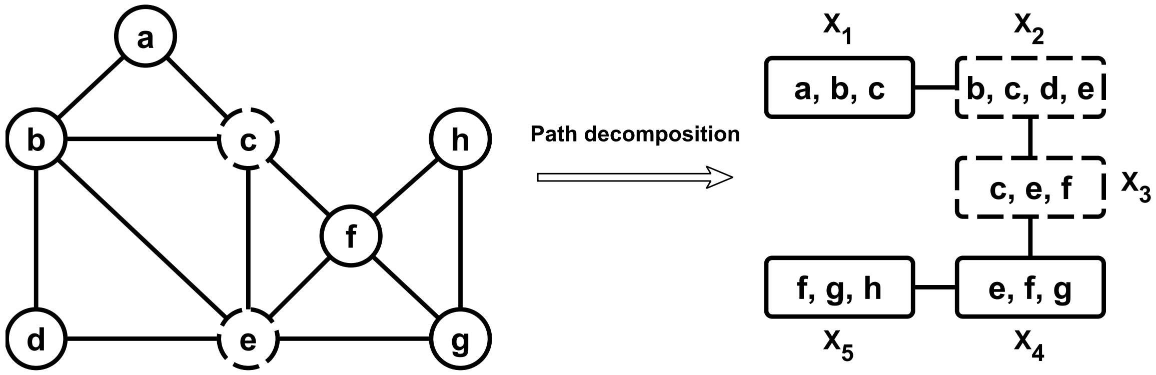

Figure 2.5: A graph and its path decomposition of width 3 in which dashed vertices c and e are separators induced by bags X_2 and X_3.

Since the separators behave like a cut-set, the original problem can be split into dependent sub-problems of limited size, which helps the development of efficient algorithms, especially those which work with the divide-and-conquer paradigm. So, the general idea behind decomposition is to break large structures into small pieces, in order to exploit this separator attribute exclusively. Figure 2.5 shows an example for some of the previous properties.

As mentioned earlier, the natural structure of the trees helps to make many hard problems easy to solve. In this regard, path decompositions can be seen as a special case

[1]The subtraction is just to ensure that the pathwidth of a graph with one edge is 1, not 2

of tree decomposition. Similar previous conditions are applied to decomposition graph, but a tree is used to connect the bags instead of a path. Formally, a tree decomposition of a graph $G = (V, E)$ is a pair $H = (T, B)$, where T is a tree whose for each node (*a.k.a.* bag) a function B associates a set of vertices of G, such that:

1. (node coverage) $\bigcup_{i=1}^{I} X_i = V$, *i.e.*, each vertex belongs to at least one bag;
2. (edge coverage) for every edge $uv \in E$, there is a bag X_i that contains u and v;
3. (coherence) for every vertex $v \in V$, the bags containing v form a connected subtree.

Particularly, as in path decompositions, the coverage of nodes and edges are kept whereas a new coherence condition is introduced. Since each graph has a tree decomposition, the crucial question is whether there is a decomposition in which all bags are small. In order to evaluate this point, metrics like treewidth must be used. In this sense, the width for tree decompositions remains as in path decomposition, that is, $\max_{1 \leq i \leq I} |X_i| - 1$. Likewise, the treewidth is still defined as the minimum width between all feasible decompositions.

Class	**Description**	**Treewidth**
Tree	two vertices are connected by exactly one path	1
Cycle	some number of vertices connected in a closed chain	2
Cactus	any two simple cycles contain at most one vertex in common	2
Series-parallel	k_4-minor-free[a]	2
Outerplanar	has a crossing-free embedding in the plane such that all vertices are on the same face	2
Pseudoforest	every connected component has at most one cycle	2
Halin	is formed by embedding a tree that has no degree-2 vertices in the plane and connecting its leaves by a cycle that crosses none of its edges	3

[a]G is a minor of H if G can be obtained from H by a series of vertex deletions, edge deletions and/or edge contractions (replacing two adjacent vertices u,v by a vertex that is adjacent to all neighbors of u or v)[16].

Table 2.1: A short list of graph classes with constant bounded treewidth

Determining whether a graph has treewidth at most an integer w is NP-complete, however, it is important to note that, if w is a fixed constant, the problem is solved in polynomial-time [3]. For this decision problem, some linear [6] and approximation [15, 7] algorithms were proposed. Moreover, fortunately, there are several classes of graphs, which are important in practice, and they have been proved to have constant bounded treewidth. The best-known classes are described in Table 2.1.

2.4.2 A FPT-algorithm for bounded treewidth graphs

In the next sections, a new FPT-algorithm that takes advantage of graphs with constant bounded treewidth will be introduced. Thus, a useful canonical structure for dynamic programming algorithms is presented, as well as all steps for the proposed algorithm.

2.4.2.1 Extended Nice Tree Decomposition

In order to improve the design of the algorithm, it is more convenient to deal with nice decompositions. A tree decomposition is nice if every bag is one of the following types:

Leaf: a bag i with no children, *i.e.*, $|X_i| = 1$;

Introduce: a bag i with exactly one child j such that $X_i = X_j \cup \{v\}$ for some vertex $v \in V$;

Forget: a bag i with one child j such that $X_i = X_j \backslash \{v\}$ for some vertex $v \in V$;

Join: a bag i with two children j, k such that $X_i = X_j = X_k$.

Nice decompositions offer a simple way to build iteratively the original graph G by adding (introduce) or removing (forget) one vertex at a time. A tree decomposition of width w and n bags can be turned into a nice tree decomposition of width w and $O(wn)$ nodes in time $O(w^2n)$. Figure 2.6 illustrates, in detail, a nice decomposition for a graph.

Moreover, an extended version of nice tree decompositions, which includes a new type of bag, is shown. It becomes necessary since whenever a vertex v is introduced in a bag, all incident edges to v are inevitably included in the partial solution. Once a path characterizes the PCP solution, each node has a degree at most two, then the particular case quoted above should be avoided. In this regard, a new type of bag called "introduce edge" is presented. This adjustment enables the inclusion of edges, one by one, which often helps the execution of the algorithm. Formally, introduce edge bags are defined as follows:

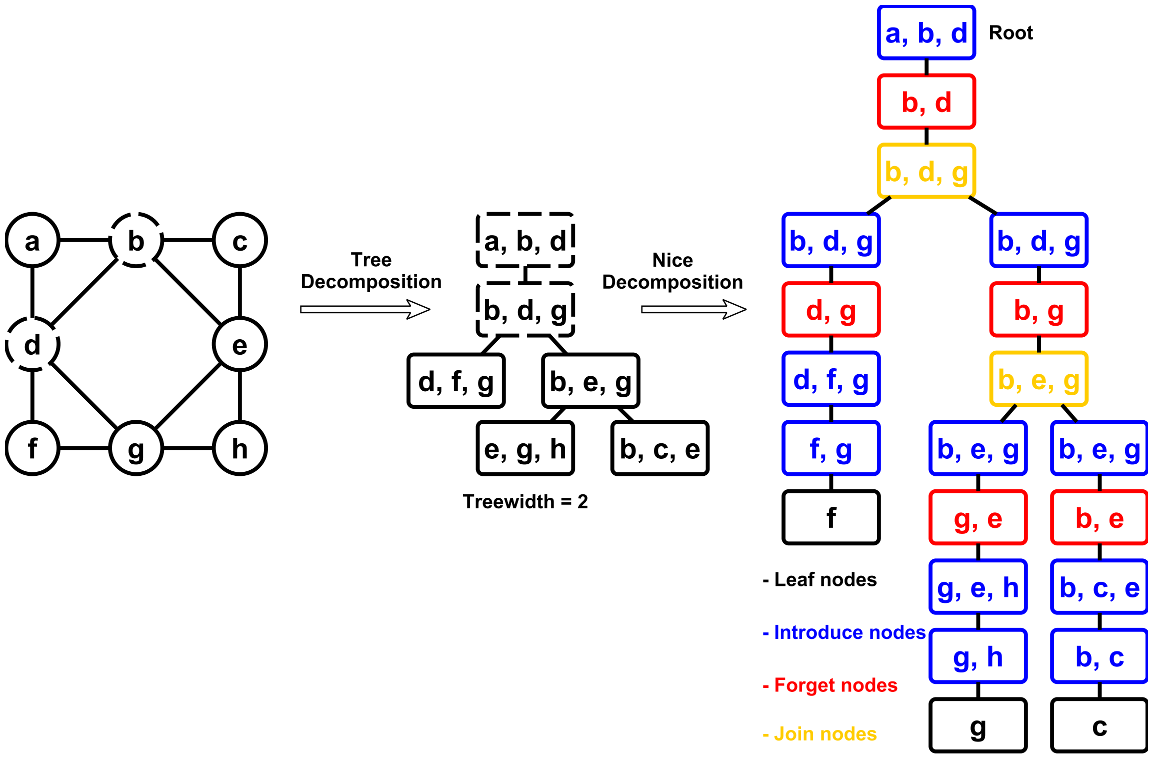

Figure 2.6: A graph and its tree and nice decomposition of treewidth 2. Dashed nodes b and d are separators induced by root bag and his child.

Introduce edge: a bag i, labeled with an edge $uv \in E$ such that $u, v \in X_i$ and with one child j such that $X_i = X_j$.

For the proposed algorithm, each edge in E must be introduced precisely once in the whole decomposition. This extended version for nice tree decompositions can be implemented in time $w^{O(1)}$ through a single top-down approach.

2.4.2.2 A Dynamic Programming approach

In order to develop efficient algorithms based on the divide-and-conquer paradigm, it is essential to understand the structure of the problem. Therefore, the proposed algorithm uses the properties of trees to exploit small separators in graphs with constant bounded treewidth for its tree decomposition. In this way, to design a dynamic programming for tree decompositions, some questions should be taken into account:

- what are the crucial pieces of information about the partial solutions of the problem?
- are there data structures to store these pieces of information efficiently?
- is it possible to obtain the solution from the set of records at the root of the tree decomposition?

- is there a quick way to compute the records for each type of bag of a nice tree decomposition?

Bearing these questions in mind, all the components of the proposed algorithm will be presented in details.

Preprocessing procedure

Given a graph G and terminals vertices s and t, the PCP objective is to find a (s,t)-path p that minimizes the difference between total transportation time cost and the total prize of the nodes belonging to p. In advance, a preprocessing is fundamental to generate the input data for the algorithm. In this sense, the process can be described in four steps:

Step I from the original graph G, is built an instance G' of a SPAW problem;

Step II a procedure is performed on G' to identify whether it admits a tree decomposition T of treewidth at most w;

Step III the tree T is converted into an extended nice tree decomposition T' of G';

Step IV an arbitrary terminal is included for each bag $X_i \in T'$.

The strategy to convert the original graph in a SPAW instance is detailed in Theorem 2.3.1 and can easily be implemented in a $O(V)$-time. The following step is characterized by the construction of a tree decomposition of treewidth at most w. As mentioned earlier, the problem of determining the treewidth of a graph is NP-hard. However, the main interest is to work just on graphs with constant and small treewidth, since there are several effective algorithms in the literature for such graphs. Next, in the third step, an extended version for a nice tree decomposition (*i.e.* including introduce edge bags) is built through a simple method that runs in polynomial time. Finally, a random terminal s or t is chosen and added for each bag in the tree decomposition to help the definition of the state of the dynamic programming.

Formal definition

Let T be an extended nice tree decomposition (see Step III), p be a path connecting the terminal vertices and V_i be the union of all bags in the subtree $T_i \subseteq T$ rooted at the bag X_i. A segment of the path p in T_i (*a.k.a.* partial solution) is a forest F. Moreover, let K be the set of terminal vertices and u^* be an arbitrary terminal of K. In regard to Step

IV of the preprocessing, all bags X_i must own a terminal u^* in such a way that every connected component of F intersects X_i.

For each bag $X_i \in T$, the essential information of all partial solutions is stored in a function $c[T_i, X, P]$ by keeping, for each subset $X \subseteq X_i$ and every partition P of X, the minimum objective value of a forest F in G_i such that:

- $\{K \cap V_i\} \subseteq V(F)$, *i.e.*, each terminal vertex in subgraph G_i must be in forest F;
- $X_i \cap V(F) = X$, *i.e.*, vertices of $X_i \backslash X$ are untouched by forest F;
- forest F has exactly q connected components $C_1, C_2, \ldots, C_q$ in such a manner that, $\forall w \in \{1, \ldots, q\}$, $P_w = V(C_w) \cap X_i$, *i.e.*, the intersections of connected components with the vertices of bag X_i form the partitions P of X.

Every time a new vertex is introduced or partial solutions are joined, different configurations of partitions become available; thereby, it is fundamental to keep updated all the pieces of information about partial solutions. Furthermore, it is worth mentioning that, given the Step IV in the preprocessing, the optimal solution value is given by $c[r, \{u^*\}, \{\{u^*\}\}]$, where r is the root bag of the tree decomposition T. Finally, if no compatible forest F can be found, $c[T_i, X, P] = +\infty$. Figure 2.7 provides more details about the procedure.

Recursive formulas for bag types

As previously mentioned, one of the most critical aspects for the development of an efficient dynamic programming algorithm is to compute, in a quick way, the recursive function for each type of bag in a tree decomposition. In this sense, all values are stored in a way that they can be quickly recovered in order to avoid unnecessary computations.

Leaf: According to Step IV, a bag i is a leaf node, if and only if, $X_i = \{u^*\}$. Once u^* is a terminal, this unique vertex must be in the partial solution; otherwise, if this condition is not satisfied, it is an infeasible case. Hence,

$$c[T_i, X, P] = \begin{cases} 0 & , \text{if } v \in X; \\ +\infty & , \text{if } v \notin X. \end{cases}$$

Introduce vertex: Let i be an introduce vertex bag with a child i' such that $X_i = X_{i'} \cup \{v\}$ for any $v \notin X_{i'}$. Since vertex v is introduced at this moment, no adjacent edges

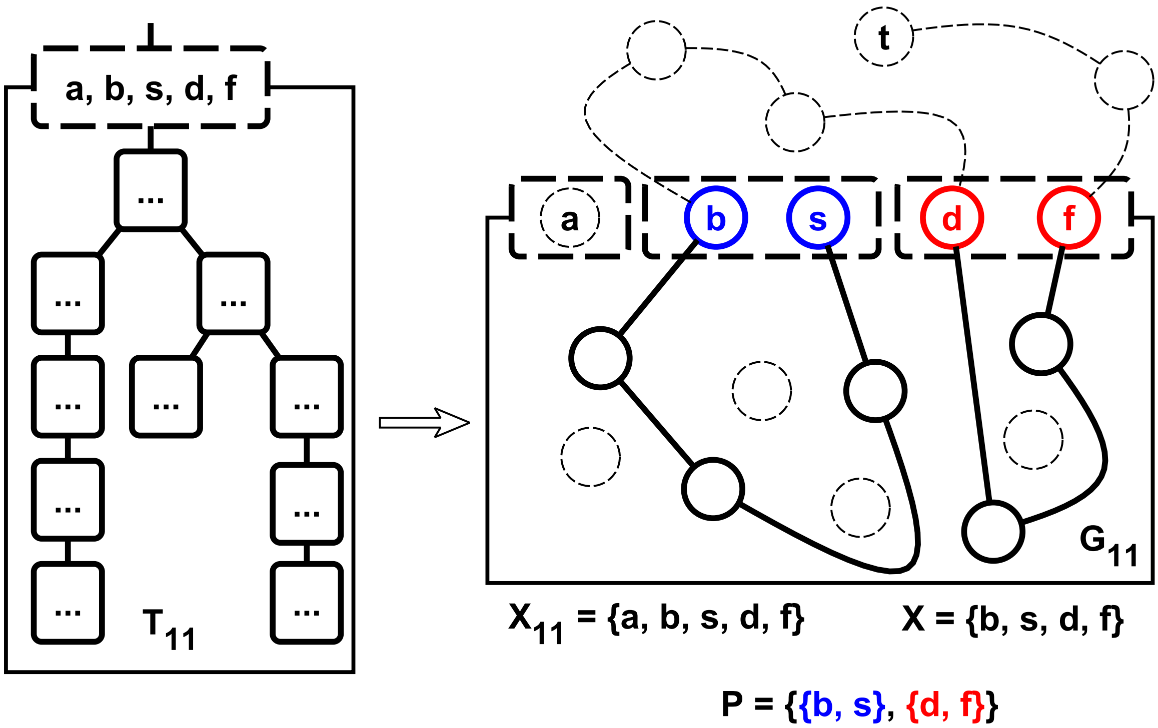

Figure 2.7: In the first part, there is a bag X_{11} and its connected subtree induced on tree T. Next, a partial solution, a subset X of X_{11} and its respective partition is presented.

were added yet and consequently, v is isolated in G_i. In this way, if vertex v belongs to the partial solution, it is inserted as a singleton on its connected component. Conversely, if v is not a part of path p, the partial solution needs to be preserved and remains the same value of the child bag. Finally, if v is a terminal and is not included in the solution, the case is infeasible. Thereby, the recursive formula for introduce vertex bags can be described as follows:

$$c[T_i, X, P] = \begin{cases} c[T_{i'}, X\backslash\{v\}, P\backslash\{\{v\}\}] & \text{, if } v \in X; \\ c[T_{i'}, X, P] & \text{, if } v \notin X \wedge v \notin K; \\ +\infty & \text{, if } v \notin X \wedge v \in K. \end{cases}$$

Forget: Let i be a forget bag with a child i' such that $X_i = X_{i'}\backslash\{v\}$ for any $v \in X_{i'}$. For each set $X \subseteq X_i$ and partition $P = \{P_1, P_2, \ldots, P_q\}$ of X, two possible situations must be taken into account: (1) if v is used in a partition of a subset of $X_{i'}$, it should be included to one of the connected components of P and (2) if v is not included, the same partition of X should be considered. Otherwise, when v is forgotten in partitions P' for computing recursive formulas of X_i, multiple values of the same partition arise and the configuration with the smallest value must be kept. Therefore, the recursive formula for forget bags can be defined as follows:

$$c[T_i, X, P] = \min\left\{\min_{P'} \; c[T_{i'}, X \cup \{v\}, P'], c[T_{i'}, X, P]\right\},$$

where the inner minimum is taken over all partitions P' of $X \cup \{v\}$ that are obtained from P by adding v to one of the existing blocks. An enlightening example is given by the bag $X_i = \{x, y, z\}$ and its child $X_{i'} = \{v, x, y, z\}$. Let $X = \{y, z\}$ be a subset of X_i and $P = \{\{y\}, \{z\}\}$ a possible partition of X. To compute the c value for X and P, the partitions P' of $X_{i'}$ should be noticed. In this way, let consider partitions $P'_1 = \{\{v, y\}, \{z\}\}$, $P'_2 = \{\{y\}, \{v, z\}\}$ and $P'_3 = \{\{y\}, \{z\}\}$ of P'. Once v is forgotten, the same configuration $\{\{y\}, \{z\}\}$ is seen for P'_1, P'_2 and P'_3, but just the partition with the smallest value is considered.

Introduce edge: Let i be an introduce edge bag which introduces an edge uv and let i' be the child of t. For each set $X \subseteq X_i$ and partition $P = \{P_1, P_2, \ldots, P_q\}$ of X, some different situations must be noticed. If one of the vertices u or v is not included in the partial solution, the partition remains the same. An equivalent result is expected when u and v are both in X but are not in the same connected component of P. Hence,

$$c[T_i, X, P] = \begin{cases} c[T_{i'}, X, P] & \text{, if } u \in X \wedge v \in X, \text{ but} \\ & \text{not in same block;} \\ c[T_{i'}, X, P] & \text{, if } u \notin X \vee v \notin X. \end{cases}$$

For the case where u and v are both in X and in the same connected component, some points must be taken into account. (1) If edge uv is not in the solution, the same partition P at $T_{i'}$ should be considered. (2) When the edge uv is picked to the solution, the connected component of u and v in P is retrieved from merging two connected components of smallest values, one containing u and the second containing v. Thus, if $u \in X \wedge v \in X$, but in the same block, then

$$c[T_i, X, P] = \min\left\{\min_{P'} \; c[T_{i'}, X, P'], c[T_{i'}, X, P]\right\}.$$

Join: Let i be a join bag with children i_1 and i_2 in such a way that $X_i = X_{i_1} = X_{i_2}$. For this case, two partial solutions, one deriving from G_{i_1} and other from G_{i_2}, are merged in one. However, this merging process can result in multiple partitions of different weight and, among them, the partition of minimum value is kept. Furthermore, this join operation can create cycles, as can be seen in Figure 2.8. The cycles can be easily avoided through the use of an auxiliary structure. Given a partition P of X,

G_F is a forest with a vertex set X, such that the set of connected components in G_F is P. For each block of P there is a tree in G_P with the same vertex set, *i.e.*, a partition $P = \{P_{i_1}, P_{i_2}, \ldots, P_{i_q}\}$ of X is an acyclic merge of partitions P_{i_1} and P_{i_2} if the merge of two forests G_{i_1} and G_{i_2} is a forest whose family of connected components is exactly P. Hence,

$$c[T_i, X, P] = \min_{P_{i_1}, P_{i_2}} c[T_{i_1}, X, P_{i_1}], c[T_{i_2}, X, P_{i_2}].$$

where in the minimum we consider all pairs of partitions P_{i_1}, P_{i_2}, such that P is an acyclic merge of them.

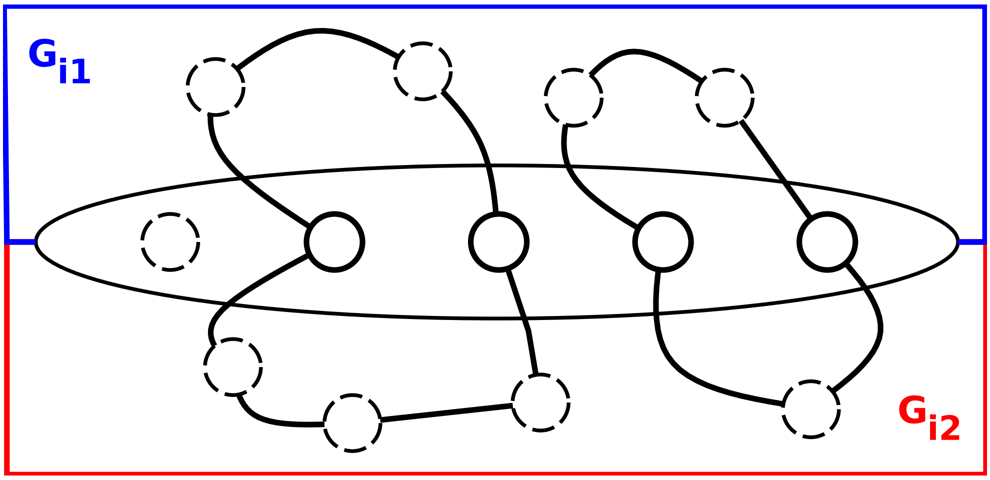

Figure 2.8: An example of cycles generated by the merging process.

These descriptions conclude the description of the recursive formulas for the values of c. Recall that every bag of the decomposition has size at most $k + 2$. Hence, the number of states per node is at most $2^{k+2} \cdot (k + 2)^{k+2} = k^{O(k)}$, since for a bag i there are $2^{|X_i|}$ subsets $X \subseteq X_i$ and at most $|X|^{|X|}$ partitions of X. The computation of a value for every state requires considering at most all the pairs of states for some other nodes, which means that each value can be computed in time $(k^{O(k)})^2 = k^{O(k)}$. Thus, up to a factor polynomial in k, which is anyhow dominated by the O-notation in the exponent, for every node the running time of computing the values of c is $k^{O(k)}$.

2.5 Conclusion and future works

In this work, an optimization problem, the Prize-Collecting Path, is presented. A NP-hardness proof is discussed, and the complexity behavior is analyzed for some graph classes of the problem. In some cases, the PCP was proved to be NP-complete which, in turn, will lead to the generation of new sets of benchmark instances. The summary of the results can be seen in Table 2.2.

Complexity	
NP-c	**Poly**
Bipartite	Tree
Grid	Cactus
Complete	Halin
Chordal	Chordal $\cap$ Bounded clique
Planar	Outerplanar
Split	Series parallel

Table 2.2: Complexity of PCP for some graph classes

Also, a polynomial FPT-algorithm is described for graphs with bounded treewidth, and a mathematical formulation is introduced to solve general instances of PCP. The ongoing investigation will consist of the implementation of the proposed FPT-algorithm for graphs with bounded treewidth and the generation of new sets of benchmark instances that are computationally hard according to natural characteristics of the problem.

Chapter 3

Vehicle Routing Problem for Transportation of Employees

3.1 Introduction

Practical applications involving employees transportation often need to handle several aspects during the optimization process, including the capability to deal with different business models depending on strategic decisions that better suit the company and its employees. These decisions also need to be made fast, due to the dynamic nature of the variables involved in the decision making, e.g., traffic and environmental conditions, employees work schedule, production management, and operational decisions. In order to both consider the interests of the company and the employees, a Logistics Service Level Agreement (LSLA) can be devised [21, 39]. The LSLA includes non-computational services such as pickup/delivery logistics and the desired quality-of-service (QoS) rules for transportation (waiting times, maximum route length, time window preferences, and other quality indicators for the users).

Similar LSLA approaches considering transportation and customer satisfaction optimization are being systematically studied for public transportation services [54], drone delivery services [27] and also transportation considering real-time traffic information [35]. Most of these transportation problems involve not only one-way deliveries, but also pickups, creating a set of routes with desired QoS such as maximum route length/time, delivery/pickup sequences, battery quality control (or electric vehicles) and time-window preferences.

As a practical application, the problem studied in this work includes several QoS constraints while minimizing routing distances. This kind of problem can be seen as an extension of the challenging Vehicle Routing Problem (VRP), which is known to be NP-Hard [12], i.e., there is no known algorithm capable of finding optimal solutions in polynomial time. As a consequence, approximation methods such are often employed to find near-optimal solutions in a reasonable time, since the high computational effort required by exact methods is impracticable. For real-life VRP applications, the resolution becomes even harder, as in general, the considered problems usually involve large-scale instances [22].

In this context, this paper deals with a practical industrial problem involving employees transportation in some Brazilian cities, as well as, the minimization of operational costs, the achievement of QoS requirements and visualization of the routes. The study case is about an energy company, with several operational units, which offers transportation to their employees, taking them from their residences to the workplace and, at the end of the expedient, bringing them back to their respective residences. Therefore, the company is responsible for hiring vehicles and also to define the routes that each one must go through.

This transportation service problem can be seen as a VRP variant with multiple attributes, but it is also possible to find characteristics of the School Bus Routing Problem (SBRP) or School Bus Routing and Scheduling Problem [42]. The SBRP aims to plan a schedule for a fleet of school buses in which, each vehicle, collects students at geographically dispersed bus stops and delivers them to their respective schools. Several constraints must be satisfied, such as the maximum capacity of a bus, the maximum travel time of a student and the time window of a school. This class of problems consists of different sub-problems that involve data preparation, selection of bus stops, generation of bus lines, school bell time adjustment, and the buses scheduling. The SBRP resolution can involve one or more sub-problem depending on the variant. Besides, many objectives can be found. The most common are minimizing of the number of buses [34], the travel distance [46] or total cost [52]

Recently, a similar problem involving a bus transport service for employees of a company was solved as an SBRP [32]. In this way, the Vehicle Routing Problem for Transportation of Employees (VRPTE) addressed in this paper, can be treated as an SBRP case, since the company offers transportation to its employees, taking them from their residences to the workplace and vice versa. In VRPTE, similar decisions like the ones in SBRP have to be made, of which may be mentioned: bus stops definition; fleet size and mix; vehicles scheduling and vehicle routing. It is worth to note that, for this VRPTE variant, the bus stops are previously known and hence, an input data.

Since the company does not own the fleet of vehicles necessary to transport their employees, a third-party company is hired to provide the transportation service. In general, several types of contracts with different vehicle types are made with the providers. Each lease agreement has its data set, and each vehicle type may belong to a single type of contract. The transportation system must assist in the choice of vehicles and the planning of their routes through an optimizer module. The objective of this module is to determine the best fleet composition for the VRPTE, as well as the set of routes that minimizes the contract costs.

This work proposes an algorithm based on the Iterated Local Search (ILS) metaheuristic [36], with composes several local search strategies, specially designed to improve different aspects of a VRPTE solution. A mathematical programming model is also proposed based on mixed-integer non-linear programming (MINLP) to present the problem constraints and the LSLA evaluation process formally. Finally, an extensive set of instances inspired by real application data is given as a benchmark for the evaluation of the algorithm.

This chapter is organized as follows. Section 3.2 details the VRPTE, including the Logistic SLA requirements and a Mathematical Programming Model. In Section 3.3, the proposed metaheuristic is presented in details. Finally, Section 3.4 describes computational results on real scenarios and Section 3.5 concludes the work.

3.2 Problem definition

The VRPTE deals with two types of expedients: single-shift and multi-shift, being that the majority of the operational units have staffs in both schemes. In this way, each unit has two different scenarios related to their expedient types. In *single-shift*, each vehicle only performs one round-trip per workday. The passengers are arranged in working groups according to the location of their residences. All passengers in a group must board and land together at predetermined bus stops on the round-trips. For each group, two bus stops are associated, which in turn will be defined as places of boarding and landing by the routing module. The outward route of a vehicle is determined by a sequence of bus stops (*a.k.a.*, boarding points), finishing at the workplace.

On the other hand, the return route starts at the workplace and must have the bus stops of each group in the reverse sequence of the outward route (*a.k.a.*, landing points). No outward route, from the first bus stop, must exceed the maximum time and the maximum distance predetermined. Such limits may be different for return routes. Finally, routes should be radial relative to the workplace, *i.e.*, it is undesirable for a vehicle during its trip to approach regions previously visited.

In *multi-shift*, there are different working teams, and each passenger belongs only to a specific one. For each workday, passengers of these teams are transported to the workplace and, at the end of the shift (*e.g.* morning, afternoon and night), are led to their respective residences. Passengers are grouped into bus stops according to the place of their residences, regardless of the working team to which they belong. Typically, each group contains only one passenger.

Moreover, in both scenarios, vehicles should perform boarding and landing of passengers at bus stops which, usually, are coincident and located near the employee's residence addresses. In these cases, there is no practical distinction between a boarding point and a landing point. It is noteworthy that, even in a typical scenario, the case described above does not exclude the possibility of having groups with more than one passenger and bus stops located in different geographic coordinates. The same definitions for round-trip routes and the maximum limits for time and distance in the single-shift scenario are applied to multi-shift, as well as the desirable radial characteristic of these routes.

3.2.1 Formal definition

Given a complete graph $G = (V, A)$, consider a set of vertex $V = \{0, 0', 1, 2, \dots, P\}$ is the set of $P + 2$ vertices and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the set of arcs. Let 0 be an artificial vertex, called virtual depot, used as starting point for all routes in solution, while the vertex $0'$ represents the workplace. The set $V' = V \setminus \{0, 0'\}$ defines the P bus stops, while $A' = A \setminus A^0$ where $A^0 = \{(0, i) : i \in V'\} \cup \{(i, 0) : i \in V'\}$ is a set of zero cost arcs, *i.e.*, $d_{i0} = d_{0i} = 0$. In this way, each working group comprises of two bus stops described by consecutive vertices p and $p + 1$, from which must be assigned, in the best way, as boarding and landing points of a round trip. Moreover, the number of passengers associated with bus stops $p, p + 1 \in V'$ for a working team $t \in T$ from the same group is represented by q_p^t, where $|T| = 1$ for single-shift scenario and $|T| > 1$ for multi-shift.

Each team $t \in T$ works in one of the B shifts of the E workdays in a month. It is important to remark that a team can work in different shifts but not in a consecutive way and, also, all employees must be in the same number of shifts per month. Since the work schedule for working teams regularly repeats in a month, the routing module only defines the routes and the operational costs for a single scale, *i.e.*, just while the schedule does not repeat. Hence, in order to determine the final cost of a contract in a month, it is necessary to add in the cost function a normalization factor α which is defined as $\frac{E \cdot B}{T}$.

In order to transport the employees, the company hires third-party logistics providers who own a heterogeneous fleet of vehicles, *i.e.*, a fleet composed of different types of vehicles. In this sense, for each type $m \in M$, K_m is the number of available vehicles, Q_m is the number of seats available in a vehicle and P_m is the maximum number of passengers that can be transported per vehicle. Once the fleet used do not belong to the company, a contract C_m^b must be signed with the third-party provider for each type of vehicle. The definition for all b types of contracts and the monthly costs associated with each one are as follows:

- **Fixed and variable costs ($\mathbf{C_m^1}$):** Each used vehicle has a fixed cost of contracting and a variable cost linked to the distance traveled.

$$C_m^1 = \alpha \cdot (F_m \cdot u_m + V_m \cdot \sum_k^{K_m} dt^k)$$

  where u_m is the number of vehicles used to transport the employees, F_m is the fixed cost per vehicle used, V_m is the variable cost per vehicle and dt^k is the sum of the distances traveled on the outward and return trip for each vehicle $k \in K_m$ used in a shift. Finally, α defines a constant factor to normalize the contractual cost according to the monthly work schedule.

- **Individual franchise ($\mathbf{C_m^2}$):** Each used vehicle has a fixed cost F_m, which allows it to travel up to a maximum hired distance $K_m^{\max}$ per shift. A variable cost V_m is applied to the travel distance that exceeds the maximum hired distance.

$$C_m^2 = \alpha \cdot [F_m \cdot u_m + V_m \cdot \sum_v^{u_m} \max(0, dt^k - K_m^{\max})]$$

- **Total franchise ($\mathbf{C_m^3}$):** Each used vehicle has a fixed cost F_m and, in turn, all available type of vehicles contributes with a constant distance $K_m^{\max}$ to compose the total franchise per shift. A variable cost V_m is applied to the travel distance that exceeds the total franchise.

$$C_m^3 = \alpha \cdot [F_m \cdot u_m + V_m \cdot \max(0, \sum_k^{K_m} dt^k - (K_m^{\max} \cdot u_m))]$$

Regardless of the contract type, for each arc $(i, j) \in A$, there is a variable quota per distance traveled defined by variable V_m. For franchise agreements, the variable quota is applied only if the total travel distance exceeds, for each vehicle in C_m^2 and for all vehicles of the same type in C_m^3, the total franchise $K_m^{\max}$ in a shift. Finally, the objective is to determine the best fleet composition, as well as the set of routes that minimizes the sum of fixed and variable costs. The logistic SLA for the problem must respect the following QoS constraints:

- every route Rt consists of two trips ($Rt = R \cup \bar{R}$) associated with the same vehicle. In this way, the outward trip R begins on the boarding of the first group of passengers and ends at the workplace ($i_1 = 0, i_{|R|} = 0'$ and $\{i_2, \ldots, i_{|R|-1}\} \subseteq V'$). In turn, the return trip $\bar{R}$ is in reverse order of visits, *i.e.*, starts at the workplace and ends at the landing of the last group of passengers ($i_1 = 0', i_{|\bar{R}|} = 0$ and $\{i_{|\bar{R}|-1}, \ldots, i_2\} \subseteq V'$);
- all routes should be radial relative to the workplace, *i.e.*, it is undesirable for a vehicle during its trip to approach regions previously visited;
- the transport of a group of passengers of the same team, associated with a bus stop, must be performed integrally and exclusively by the same vehicle in the work schedule, *i.e.* the vehicle must carry all passengers from the bus stop, being forbidden the boarding/landing fractioned in the scale;
- the maximum number of passengers of a vehicle must be respected;
- each bus stop belongs only to one trip (outward or return).

Finally, it is worth noting the distance and time asymmetry between vertices, *i.e.*, d_{ij} is not necessarily equal to $d_{ji}, \forall i, j \in V \backslash \{0\}$, as well as, l_{ij} can be different from $l_{ji}, \forall i, j \in V \backslash \{0\}$.

3.2.2 Mathematical model

In the following, a new MINLP is presented for the VRPTE. The model works with the decision variables as follows:

$\boldsymbol{x_{ij}^{tk}} \in \mathbb{B}$ number of times that the directed arc (i, j) is visited by vehicle k in a outward trip to attend a working team t

$\boldsymbol{\bar{x}_{ij}^{tk}} \in \mathbb{B}$ number of times that the directed arc (i, j) is visited by vehicle k in a return trip to attend a working team t

$\boldsymbol{y^{tk}} \in \mathbb{B}$	define if a vehicle k is dispatched to attend a working team t
$\boldsymbol{z^{k}} \in \mathbb{B}$	indicate if a vehicle k is used in any route
$\boldsymbol{w_m} \in \mathbb{B}$	denote if a vehicle of type m is dispatched
$\boldsymbol{u_m} \in \mathbb{Z}$	number of vehicles of type m in solution
$\boldsymbol{dt^{k}} \in \mathbb{R}$	distance traveled per shift for a vehicle k
$\boldsymbol{q_p^{t}} \in \mathbb{Z}$	number of passengers collected in bus stop p for a working team t
$\boldsymbol{c_m^{b}} \in \mathbb{B}$	denote if a contract of type b is active for a vehicle of type m
$\boldsymbol{C_m} \in \mathbb{R}$	sum of operational costs for a vehicle of type m

The formulation for the VRPTE is then given by the following model:

$$\min \sum_{m \in M} C_m \tag{3.1}$$

s.t.

$$\sum_{m \in M} \sum_{k=1}^{K_m} \sum_{j \in V' \setminus \{p,p+1\}} \sum_{\Delta=0}^{1} x_{j,i+\Delta}^{tk} = 1, \qquad \forall i \in V' | i \bmod 2 = 1, \forall t \in T | q_i^t > 0 \tag{3.2}$$

$$\sum_{j \in V' \setminus \{i\}} x_{ji}^{tk} - \sum_{(i,j) \in A} x_{ij}^{tk} = 0, \qquad \forall i \in V', \forall t \in T, \forall m \in M, \forall k \in K_m \tag{3.3}$$

$$\sum_{(j,i) \in A} \bar{x}_{ji}^{tk} - \sum_{j \in V' \setminus \{i\}} \bar{x}_{ij}^{tk} = 0, \qquad \forall i \in V', \forall t \in T, \forall m \in M, \forall k \in K_m \tag{3.4}$$

$$\sum_{j \in V'} x_{0,j}^{tk} = y^{tk}, \qquad \forall t \in T, \forall m \in M, k \in K_m \tag{3.5}$$

$$\sum_{j \in V'} x_{j,0'}^{tk} = y^{tk}, \qquad \forall t \in T, \forall m \in M, k \in K_m \tag{3.6}$$

$$\sum_{i \in V'} \sum_{(j,i) \in A'} q_{it} \cdot x_{ji}^{tk} \leq P_m \cdot y^{tk}, \qquad \forall t \in T, \forall m \in M, k \in K_m \tag{3.7}$$

$$y^{tk} \geq y^{t,k+1}, \qquad \forall t \in T, \forall m \in M, k \in \{1, \dots |K_m| - 1\} \tag{3.8}$$

$$\sum_{(i,j) \in A(S)} x_{ij}^{tk} \leq (|S| - 1) \cdot y^{tk}, \qquad \forall t \in T, \forall m \in M, \forall k \in K_m, S \subseteq V' \tag{3.9}$$

$$\sum_{(i,j) \in A} d_{ij} \cdot x_{ij}^{tk} \leq D \cdot y^{tk}, \qquad \forall t \in T, \forall m \in M, k \in K_m \tag{3.10}$$

$$\sum_{(i,j) \in A} l_{ij} \cdot x_{ij}^{tk} \leq L \cdot y^{tk}, \qquad \forall t \in T, \forall m \in M, k \in K_m \tag{3.11}$$

$$\sum_{(i,j) \in A} d_{ij} \cdot \bar{x}_{ij}^{tk} \leq \bar{D} \cdot y^{tk}, \qquad \forall t \in T, \forall m \in M, k \in K_m \tag{3.12}$$

$$\sum_{(i,j) \in A} l_{ij} \cdot \bar{x}_{ij}^{tk} \leq \bar{L} \cdot y^{tk}, \qquad \forall t \in T, \forall m \in M, k \in K_m \tag{3.13}$$

$$\sum_{\Delta=0}^{1} (x_{0,i+\Delta}^{tk} - \bar{x}_{i+\Delta,0}^{tk}) = 0, \qquad \forall i \in V', i \bmod 2 = 1, \forall t \in T, \forall m \in M, \forall k \in K_m \tag{3.14}$$

$$\sum_{\Delta=0}^{1} (x_{i+\Delta,0'}^{tk} - \bar{x}_{0',i+\Delta}^{tk}) = 0, \qquad \forall i \in V', i \bmod 2 = 1, \forall t \in T, \forall m \in M, \forall k \in K_m \tag{3.15}$$

$$\sum_{\Delta_1=0}^{1}\sum_{\Delta_2=0}^{1}(x_{i+\Delta_1,j+\Delta_2}^{tk}-\bar{x}_{j+\Delta_1,i+\Delta_2}^{tk})=0, \quad \forall i,j\in V', i \bmod 2=1, j \bmod 2=1, \tag{3.16}$$

$$i\neq j, \forall t\in T, \forall m\in M, \forall k\in K_m$$

$$w_m\geq\sum_{k=1}^{K_m}y^{tk}, \quad \forall t\in T, \forall m\in M \tag{3.17}$$

$$z^k\geq y^{tk}, \quad \forall t\in T, \forall m\in M, k\in K_m \tag{3.18}$$

$$\sum_{k=1}^{K_m}z^k=u_m, \quad \forall m\in M \tag{3.19}$$

$$\sum_{t\in T}\sum_{i,j\in V}(d_{ij}\cdot x_{ij}^{tk}+d_{ji}\cdot\bar{x}_{ji}^{tk})=dt^k, \quad \forall m\in M, \forall k\in K_m \tag{3.20}$$

$$\sum_{b=1}^{3}c_m^b=w_m, \quad \forall m\in M \tag{3.21}$$

$$F_m\cdot u_m+\alpha\cdot V_m\cdot\sum_{k=1}^{K_m}dt^k-\mathcal{M}\cdot(1-c_m^b)\leq C_m, \quad \forall m\in M|b=1 \tag{3.22}$$

$$dt^k-K_m^{\max}\leq\bar{dr}^{k,2}, \quad \forall m\in M, k\in K_m \tag{3.23}$$

$$F_m\cdot u_m+\alpha\cdot V_m\cdot\sum_{k=1}^{K_m}\bar{dr}^{k,2}-\mathcal{M}\cdot(1-c_b^m)\leq C_m, \quad \forall m\in M|b=2 \tag{3.24}$$

$$\sum_{k=1}^{K_m}dt^k-K_m^{\max}\leq\bar{dr}^{k,3}, \quad \forall m\in M \tag{3.25}$$

$$F_m\cdot u_m+\alpha\cdot V_m\cdot\bar{dr}^{k,3}-\mathcal{M}\cdot(1-c_m^b)\leq C_m, \quad \forall m\in M|b=3 \tag{3.26}$$

$$\sum_{t\in T}\sum_{m\in M}\sum_{k=1}^{K_m}\sum_{i=1}^{n}(x_{2\cdot i,2\cdot i-1}^{tk}+x_{2\cdot i-1,2\cdot i}^{tk})=0, \tag{3.27}$$

$$\sum_{t\in T}\sum_{m\in M}\sum_{k=1}^{K_m}\sum_{i=1}^{n}(\bar{x}_{2\cdot i,2\cdot i-1}^{tk}+\bar{x}_{2\cdot i-1,2\cdot i}^{tk})=0, \tag{3.28}$$

$$\sum_{t\in T}\sum_{m\in M}\sum_{k=1}^{K_m}\sum_{i\in V'}(x_{0'i}^{tk}+\bar{x}_{i0'}^{tk})=0, \tag{3.29}$$

$$x_{ij}^{tk}\in\mathbb{B}, \quad \forall t\in T, \forall m\in M, \forall k\in K_m, \forall(i,j)\in A \tag{3.30}$$

$$\bar{x}_{ij}^{tk}\in\mathbb{B}, \quad \forall t\in T, \forall m\in M, \forall k\in K_m, \forall(i,j)\in A \tag{3.31}$$

$$x_{0',i}^{tk}\in\mathbb{B}, \quad \forall t\in T, \forall m\in M, \forall k\in K_m, \forall i\in V' \tag{3.32}$$

$$\bar{x}_{i,0'}^{tk}\in\mathbb{B}, \quad \forall t\in T, \forall m\in M, \forall k\in K_m, \forall i\in V' \tag{3.33}$$

$$y^{tk}\in\mathbb{B}, \quad \forall t\in T, \forall m\in M, \forall k\in K_m \tag{3.34}$$

$$z_k\in\mathbb{B}, \quad \forall m\in M, \forall k\in K_m \tag{3.35}$$

$$w_m\in\mathbb{B}, \quad \forall m\in M \tag{3.36}$$

$$u_m\in\mathbb{Z}, \quad \forall m\in M \tag{3.37}$$

$$dt^k\geq 0, \quad \forall m\in M, \forall k\in K_m \tag{3.38}$$

$$\bar{dt}^{k,2}\geq 0, \quad \forall m\in M, \forall k\in K_m \tag{3.39}$$

$$\bar{dt}^{k,3}\geq 0, \quad \forall m\in M \tag{3.40}$$

$$q_{it}\geq 0, \quad \forall i\in V', \forall t\in T \tag{3.41}$$

$$c_m^b\in\mathbb{B}, \quad \forall m\in M, \forall b\in\{1,\ldots,3\} \tag{3.42}$$

$$C_m\geq 0, \quad \forall m\in M \tag{3.43}$$

The objective function (3.1) seeks to minimize the sum of all contractual costs. Constraints (3.2) ensure that just one of the bus stops associated with a working group is used in an outward trip. Constraints (3.3-3.6) assure the flow conservation conditions for all routes. Constraints (3.7) guarantee that the maximum number of passengers per vehicle is respected. Constraints (3.8) impose symmetry breaking rules to allow vehicle $k + 1$ only be used if vehicle k is dispatched. Constraints (3.9) generalize the subtour elimination constraints and ensures that the solution is cycle free. Constraints (3.10-3.13) enforce that a maximum distance and time is considered for outward and return trips. Constraints (3.14-3.16) guarantee the consistency of the routing plan. Constraints (3.17) and (3.18) link variables y with variables w and z, respectively. Constraints (3.19) define the number of vehicles dispatched in solution, while constraints (3.20) determine the distance traveled per vehicle. Constraints (3.21) denote an active lease contract for a specific type of vehicle. Constraints (3.22), (3.24) and (3.26) are, respectively, the contractual costs for agreements of type "*fixed and variable*", "*individual franchise*"and "*total franchise*". Constraints (3.23) and (3.25) compute the residual distance according to the type of contract. Constraints (3.27) and (3.28) guarantee that both bus stops associated with a working group are on trips of opposite directions. Finally, constraints (3.29) avoid an outward trip to start in the workplace, as well as also forbid a return trip to end in the workplace.

3.3 Methodology

Based on the VRPTE characteristics, a routing module considering all attributes was developed. This module was then incorporated into the transportation system by developing the communication interface between the optimization module and the current system. It should be noted that the task of solving VRPTE is not trivial since its optimization version is NP-Hard, *i.e.*, there is no known algorithm able to find optimal solutions in polynomial time [33]. As a consequence, approximation methods are often used to solve the problem, since the high computational effort required for its resolution becomes impractical, in many cases, to use exact methods. In this context, metaheuristics appear as alternatives to the exact methods by obtaining generally good quality solutions in a significantly shorter time. The proposed routing module consists of developing a multi-start hybrid heuristic based on the Iterated Local Search (ILS) metaheuristic [36], which in local search uses a Randomized Variable Neighborhood Descent (RVND) procedure [51]. Some works in literature use this method for dealing with VRP with several types of attributes [53, 49, 41, 9], mainly for heterogeneous fleet [44].

Algorithm 3.1 presents all procedures required to the ILS. In an ILS-based algorithm, four procedures must be defined: (i) `BuildInitialSolution` (line 2), which provides

Algoritmo 3.1: Iterated Local Search

```
1 begin
2     s0 ← BuildInitialSolution()
3     s* ← LocalSearch(s0)
4     while stopping criterion is not satisfied do
5         s' ← Perturbation(s*)
6         s'* ← LocalSearch(s')
7         s* ← AcceptanceCriterion(s*, s'*)
8     return s*
```

initial solutions to the problem; (ii) `LocalSearch` (lines 3 and 6), essential to explore the solution space effectively; (iii) `Perturbation` (line 5), which modifies the current solution so that new region in solution space is explored, and (iv) `AcceptanceCriterion` (line 7), responsible for determining which solution will be used during the next perturbation phase.

3.3.1 Constructive algorithm

The constructive procedure used to create the initial solution was adapted from existing methods in the literature, taking into consideration the multi-start characteristic of the proposed algorithm. In this sense, the constructive procedure uses a greedy insertion method with a random starting point to generate diversified initial solutions for the algorithm.

The procedure is composed of three distinct phases and is presented in Algorithm 3.2. The first phase (lines 2-12) consists of transporting the larger groups of passengers, which in turn can only be led by a single-vehicle type, in general, the one with the highest capacity. For each multi-start iteration, the initial solution s is reinitialized (line 2), *i.e.*, all variables associated with the solution are reset to their respective default values. In order to diversify the built solution, a working group is randomly selected, and the assignment for bus stops associated with the respective group is inverted, *i.e.*, boarding points become landing points and vice versa (line 3). Then, for each team, a list composed of groups served exclusively by a vehicle type is created (lines 5-8). Finally, the list is shuffled (line 9) and each of its groups is added iteratively to the solution (lines 10-12), creating new routes.

The second phase adds to the solution vehicle types not yet used for passengers transportation, in order to ensure that the constructive algorithm respects all signed contracts.

Algoritmo 3.2: `BuildInitialSolution`

```
   Data: Solution s, Groups G, List LG, List LT
1  begin
       // PHASE 1
2      s ← Reinitialize()
3      s ← InvertPoints()
4      foreach Team t ∈ T do
5          LG ← Reinitialize()
6          foreach Group g ∈ Team t do
7              if Group g is served by Type h then
8                  LG ← LG + (g, h)
9          LG ← Shuffle(LG)
10         foreach Group g ∈ LG do
11             s ← CreateRoute(g, h)
12             Groups G ← G − {g}
       // PHASE 2
13     foreach Team t ∈ T do
14         LT ← CheckNonUsedTypes()
15         LT ← Shuffle(LT)
16         foreach Type h ∈ LT do
17             faça
18                 Group g ← Random(G)
19             enquanto Group g is unfeasible by capacity
20             s ← CreateRoute(g, h)
21             G ← G − {g}
       // PHASE 3
22     foreach Team t ∈ T do
23         while G ≠ ∅ do
24             Group g ← Random(G)
25             foreach Route r in Team t do
26                 if insertion of  Group g is feasible then
27                     cost ← EvaluateInsertion(g, r)
28                     if cost < bestCost then
29                         bestCost ← cost
30                         Route r* ← r
31             s ← InsertionInRoute(g, r*)
32             G ← G − {g}
33     return s
```

Firstly, all types of vehicles not yet used, are added to a list (line 14), which in turn is shuffled to provide diversification (line 15). Then, for each type of vehicle in the list (line 16), a group is randomly selected (line 18), and a feasibility check is performed on the creation of a new route (line 19); if the operation is not feasible, a new group is selected. Finally, a new route is created using the group and type of vehicle previously chosen (lines 20-21).

The third phase comprises the insertion of remaining groups not yet served. In this way, a group is randomly selected (line 24) and added to the best existing route (lines 31-32), using the Cheapest Feasible Insertion Criterion (MCFIC) as evaluation method. The MCFIC is denoted by $g(k, r) = (c^r_{ik} + c^r_{kj} - c^r_{ij})$ and determines, for a working team, the insertion cost g between the bus stop k and each pair of adjacent bus stops i and j already included in a route r.

3.3.2 Local search

Local search is an iterative process that explores the neighborhood of a solution to obtain a better one. For the routing module, the local search is performed by an RVND-based procedure, which is an extension of the *Variable Neighborhood Descent* (VND) method. VND is an iterative method that systematically applies a set of neighborhood structures in order to improve the current solution of a given problem. The method is based on the principle that, in general, the local optima of distinct neighborhoods are relatively close to each other. Unlike VND, where the application order of neighborhoods is specified deterministically, for RVND the order is randomly defined. This random approach has been shown through empirical tests that, on average, can produce better results than the deterministic version.

Algoritmo 3.3: Local search procedure

Data: Structure s, List $\mathcal{NL}$

begin
- $\mathcal{NL} \leftarrow$ `Initialize`()
- **while** $\mathcal{NL} \neq \varnothing$ **do**
  - $\mathcal{N}^{(h)} \leftarrow$`Random`($\mathcal{NL}$)
  - $s' \leftarrow$`BestNeighbor`(s, $\mathcal{N}^{(h)}$)
  - **if** $f(s') < f(s)$ **then**
    - $s \leftarrow s'$
    - $\mathcal{NL} \leftarrow$`Reinitialize`()
  - **else**
    - $\mathcal{NL} \leftarrow \mathcal{NL} - \{\mathcal{N}^{(h)}\}$
- **return** s

The pseudocode for the RVND is shown by Algorithm 3.3. Let $\mathcal{NL} = \{\mathcal{N}^1, \mathcal{N}^2, \ldots, \mathcal{N}^x\}$

be a list composed of x neighborhood structures associated with the movements described later in this section (line 2). At each iteration of the main loop (lines 3-10), a neighborhood $\mathcal{N}^{(h)} \in \mathcal{NL}$ is randomly chosen (line 4), and the best neighbor is defined (line 5). If the solution is improved, $\mathcal{NL}$ is reset with all neighborhoods (line 8); otherwise, $\mathcal{N}^{(h)}$ is removed from $\mathcal{NL}$ (line 10). The $\mathcal{NL}$ list is composed of intra-route and inter-route neighborhoods, which are exhaustively evaluated. However, it is worth noting that in these neighborhoods the costs and the feasibility are evaluated at constant time $\mathcal{O}(1)$.

3.3.2.1 Intra-route neighborhoods

Eight intra-route neighborhood structures were applied to improve the transportation cost in a single route. It is important to remark that in the intra-route structures, it is not necessary to check if the maximum number of passengers is exceeded since these movements only change the visiting order of the bus stops. Therefore, the number of passengers remains constant on the route. The cost of the new solution obtained by the application of the neighborhood is calculated, taking into account the distance traveled in the route. In order to know if the new route is feasible, it must be verified if the maximum distances and the maximum times are respected. The neighborhood structures, as well as the cost calculations for the new routes, are shown below.

- $\mathcal{N}^1$ - ***Exchange***: Permutation between two working groups.
- $\mathcal{N}^2$ - ***Reinsertion***: One working group is removed and inserted in another position of the route.
- $\mathcal{N}^3$ - ***Or-opt2***: Two adjacent working groups are removed and inserted sequentially in another position of the route.
- $\mathcal{N}^4$ - ***Or-opt3***: Three adjacent working groups are removed and inserted sequentially in another position of the route.
- $\mathcal{N}^5$ - ***2-opt***: Two nonadjacent arcs are deleted, and another two are added in such a way that a new route is generated.
- $\mathcal{N}^6$ - ***Split***: A route r_1, belonging to a working team t, is split into smaller routes for the solution s. For this purpose, let $H' = \{2, \ldots, t\}$ be a subset of H composed by all vehicles available under contract, except those with the least maximum number of passengers. In Figure 3.1, a route $r_1 \in s$ associated with a vehicle $k \in H'$ is randomly selected. Then, while r_1 is not empty, the working groups of r_1 are sequentially

transferred to a new route $r' \notin s$ associated with a vehicle $k' \in \{1, \ldots, k-1\}$, so that the maximum number of passengers is respected. The new routes generated r_2 and r_3 are added to the solution s whereas the route r_1 is removed. It is worth mentioning that the route r_1 is only split if vehicles are available.

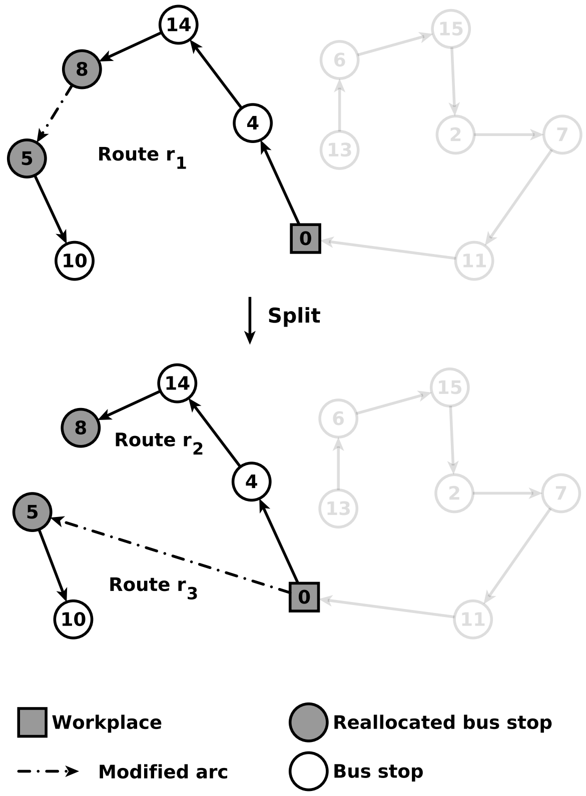

Figure 3.1: Intra-route neighborhood *Split*

- $\mathcal{N}^7$ - ***Trade***: Trade the order of boarding and landing for two bus stops of the same group.
- $\mathcal{N}^8$ - ***Upgrade***: A different type of vehicle with lower cost is assigned to an existing route. The sequence of bus stops remains unchanged in route; however, the vehicle used for the transportation of the passengers is changed. In this case, no calculation of distance or time is required. The cost of the solution is recalculated, taking into account the fixed and variable costs associated with the new type of vehicle selected.

The pseudocode for the neighborhoods $\mathcal{N}^1$ to $\mathcal{N}^5$, which have a similar structure, is shown in Algorithm 3.4. The parameters of the algorithm are the current solution s and the neighborhood η to be evaluated. At first, for each route in solution s, all pairs of distinct working groups i and j or subset of consecutive groups are evaluated (lines 2-4).

The number of consecutive groups changes according to the neighborhood considered and can be 1, 2, or 3 working groups.

The *computeMovement* method defines the value of the neighboring solution by applying the η movement (line 5). If the movement is feasible and, at the same time, lead to a better quality solution than the current iteration solution ($\bar{s}$), the solution $\bar{s}$ is updated (lines 6-8). The movement is feasible if it satisfies the distance and maximum time constraints of the route. At the end of the algorithm (lines 9 - 11), the best of all neighboring solutions found is returned.

Algoritmo 3.4: Intra-route procedure (s, η)

```
   Data: Solution s, Neighborhood η
1  begin
2      foreach Route r1 ∈ s do
3          foreach Group i ∈ r1 (or subset of consecutive groups r1) do
4              foreach Group j ∈ r1 (or subset of consecutive groups r1) do
5                  s' ← computeMovement(s, r1, i, j, η)
6                  if s' is feasible and f(s') < f(s̄) then
7                      f(s̄) ← f(s')
8                      s̄ ← s'
9      if f(s̄) < f(s) then
10         s ← s̄
11     return s
```

3.3.2.2 Inter-route neighborhoods

In order to reduce transportation costs between routes of the same workday, six inter-route neighborhood structures were implemented. As these structures involve more than one route, it is necessary to verify the feasibility of the maximum number of passengers on those routes. The neighborhoods are defined as follows.

- $\mathcal{N}^9$ - ***Swap(1,1)***: Permutation between a working group k from a route r_1 and a group l, from a route r_2.
- $\mathcal{N}^{10}$ - ***Swap(2,1)***: Permutation of two adjacent working groups, k and l, from a route r_1 by a group k' from a route r_2.
- $\mathcal{N}^{11}$ – ***Swap(2,2)***: Permutation between two adjacent working groups, k and l, from a route r_1 by another two adjacent groups k' and l', belonging to a route r_2.

- $\mathcal{N}^{12}$ – ***Shift(1,0)***: A working group k is transferred from a route r_1 to a route r_2.

- $\mathcal{N}^{13}$ – ***Shift(2,0)***: Two adjacent working groups, k and l, are transferred from a route r_1 to a route r_2.

- $\mathcal{N}^{14}$ – ***Join***: Two routes are concatenated in a new route. In Figure 3.2, two routes r_1 and r_2 are randomly selected. Both routes must be associated with a vehicle, which in turn is not the one with the largest available capacity. Hence, a new route r_3 is created, and a lower cost vehicle with enough capacity to transport the groups from routes r_1 and r_2 is selected. In this way, the groups of the route r_1 are, sequentially, transferred to route r_3, followed by the groups of the route r_2. The routes r_1 and r_2 are removed and the new route r_3 is added to the current solution.

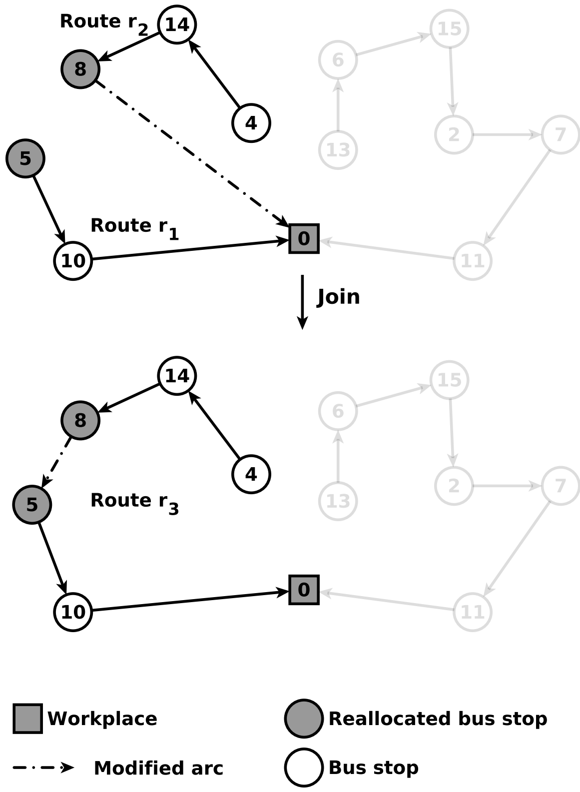

Figure 3.2: Intra-route neighborhood *Join*

The pseudocode of the algorithm for inter-route neighborhood structures, $\mathcal{N}^9$ to $\mathcal{N}^{14}$, is described in Algorithm 3.5. The parameters of the algorithm are the current solution s and the neighborhood η to be evaluated. Initially, for each distinct pair of routes r_1 and r_2, the algorithm explores feasible movements. In this way, a working group (or a subset of consecutive groups) of the route r_1 is selected to have its movement evaluated in case it is shifted or swapped by another group (or a subset of consecutive groups) of route r_2 (lines 6-7). The behavior of the procedure *computeMovement* is similar to the same method presented in the intra-route procedure.

Algoritmo 3.5: Inter-route procedure (s, η)

```
   Data: Solution s, Neighborhood η
1  begin
2      for r1 ← 1 until r do
3          for r2 ← 1 until r (or r2 ← r1 + 1 until r for Swap(1,1)) do
4              if (r1 ≠ r2) then
5                  foreach Group i ∈ r1 (or subset of consecutive groups of r1) do
6                      foreach Group j' ∈ r2 (or subset of consecutive groups of r2)
                         do
7                          s' ← computeMovement(s, η)
8                          if (f(s') < f(s̄) e s' is feasible) then
9                              f(s̄) ← f(s')
10                             s̄ ← s'
11     if f(s̄) < f(s) then
12         s ← s̄
13     return s
```

3.3.3 Perturbation mechanisms

Four perturbation mechanisms based on inter-period neighborhood structures were developed. It is important to note that all neighborhoods do not take into account the improvement of the quality of the solution. At each execution of the perturbation procedure, one of the following structures is randomly selected and executed *maxPerturb* times: *Swap(1, 1)*, *Shift(1, 0)*, *Shift(2, 0)* and *Trade*.

3.3.4 Proposed algorithm

Algorithm 3.6, presents the pseudocode for the developed algorithm. In each of the $MaxIterMS$ iterations (lines 3 - 13), a solution is generated using the constructive heuristic (line 5). Next, the `LocalSearch` is applied to refine the current solution (line 7). After the local search, a post-optimization method is executed (line 8). It consists of two steps, the first run through the routes and performs the trade of bus stops of the same working group if the total travel distance is decreased. The second step is to optimize the use of vehicles by changing, wherever possible, the vehicle type associated with the route by a lower-cost one since the number of available seats is enough. If the current solution is improved, the iterator $iterILS$ is reset (line 11). Finally, the best solution is perturbed (line 12) and iterator $iterILS$ is incremented (line 13).

Algoritmo 3.6: `MILS-RVND` algorithm

```
   Data: MaxIterMS, MaxIterILS, TimeLimit, MaxPerturb
1  begin
2      f(s*) ← ∞
3      for iterMS ← 0  until  (MaxIterMS / TimeLimit) do
4          iterILS ← 0
5          s ← BuildInitialSolution()
6          for iterILS ← 0  until  MaxIterILS do
7              s ← LocalSearch(s)
8              s ← PostOptimization(s)
9              if f(s) < f(s*) then
10                 s* ← s
11                 iterILS ← 0
12             s ← Perturb(s*, MaxPerturb)
13             iterILS ← iterILS + 1
14     return s*
```

3.4 Experiments with real scenarios

In this section is presented the numerical results for the VRPTE. The proposed algorithm was coded in C++, and computational tests were performed on an Intel Core i7 3.40 GHz PC with 16 GB of RAM running Ubuntu 14.04 OS with just one thread.

3.4.1 New benchmark set

A set of 432 artificial instances were generated based on highly realistic scenarios[1] and divided into six datasets according to the number of passengers. More details about the characteristics of each dataset can be seen in Table 3.1. Due to confidentiality terms, the real data has been preserved, and a new dataset generator was designed and implemented based on characteristics of the real scenarios. Thus, since all instances must be similar to the workload of the company, the number of workdays is set to 30. In the same direction, the number of shifts per day is equal to three, and the number of working teams is fixed to either one (single-shift) or five (multi-shift).

[1] Real data is confidential, but the company provided approximate values for the instance parameters.

Moreover, based on the practical application, all maps dimensions ranges from five to 100 kilometers. Finally, the number of working groups varies from 10% of the number of passengers to one passenger per group. As mentioned in Section 3.2, is usual in real scenarios, the situation where a group has nearly one passenger.

Table 3.1: Basic characteristics for datasets

Dataset	Number of Instances	Number of Passengers	Number of Groups	
			Min	Max
i100	72	100	10	100
i250	72	250	25	250
i500	72	500	54	497
i750	72	750	75	740
i1000	72	1000	111	998
i1500	72	1500	152	1500
	432	-	10	1500

Two functions of probability distributions define the positioning for each bus stop [43]. In the former function, the continuous uniform distribution generates random floating-point values in an interval [a,b) with constant probability density. This distribution, also known as rectangular, produces random coordinates distributed evenly on the cartesian plane and is described by function p where $p(x)$ is equal $\frac{1}{b-a}$ for $a < x < b$, and 0 otherwise. The latter function, well-known as Gaussian distribution, is a symmetric distribution in which most-frequent values are concentrated around the mean and the values with lowest frequency taper off equally in both directions. The probability density function for Gaussian distribution is defined as $p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$, where μ and σ are, respectively, the mean and standard deviation of the distribution. This function creates clusters in such a way that the mean defines the location for centroid whereas the standard deviation measures the variation around the average (centroid), *i.e.*, the coefficient of dispersion ($D = \sigma^2/\mu$). When the standard deviation is low, the cluster is dense, concentrating the bus stops within a smaller area, similar to regions of high demographic density; otherwise, the cluster is sparse, suchlike the distribution of residents in planned cities.

Based on the above functions, the generator determines two types of positioning for bus stops: randomized (continuous uniform distribution) or cluster-based (Gaussian distribution). For the last one, the number of clusters is chosen exclusively between one or five and their respective centroids are arbitrarily positioned in regions away from the axes, in order to avoid a high concentration of bus stops at peripheral regions. Thus, the concentration for clusters is determined according to the coefficient of dispersion. Note that, due to QoS constraints, the bus stops associated with the same working group are at most 200 meters away. Likewise, the travel distance between bus stops i and j is defined by c_{ij} and is asymmetric in such a way that $c_{ji} = [c_{ij} - 0.5, c_{ij} + 0.5]$, *i.e.*, the outward and return distance differ no more than 500 meters.

The location of the workplace can be defined as follows. Let S be the set of points corresponding to the location of all bus stops in an instance. The convex hull is the set of all convex combinations of points in S, *i.e.*, the smallest convex polygon that encloses all points. In this sense, the workplace can be positioned in three ways: centralized, inside, or outside the polygon. When centralized, the workplace is defined as a centroid of the convex hull and is obtained in a $\mathcal{O}(n)$ time operation, whereas, for the other cases, it is randomly positioned inside or outside the polygon. Moreover, it is important to highlight that the Monotone Chain algorithm [2], which has a time complexity of $\mathcal{O}(n \log n)$ in the worst case, was used to compute the convex hull.

For each bus stop i and j, the distance d_{ij} is computed by $\sqrt{(x_j - x_i) + (y_j - y_i)}$ where x and y are the coordinates of the points in the cartesian plane. The maximum distance for a trip is randomly chosen between d_{min} and d_{max}, which are, respectively, the minimum and maximum distance from the workplace to an extreme point of the convex hull. On the other hand, for each roadway, the traveling time t_{ij} is calculated according to the average road speed s_{ij} which, in turn, is arbitrarily selected in a range from s_{min} to s_{max}. In this sense, s_{min} and s_{max} are the minimum and maximum average speed in km/h for all roadways and assume, respectively, the values 10 and 50, estimated through previous studies of mobility in a Brazilian metropolis [45]. Finally, the traveling time can be computed by $\frac{d_{ij}}{s_{ij}}$, whereas the maximum travel time is randomly selected in an interval from $\frac{d_{min}}{s_{min}}$ to $\frac{d_{max}}{s_{min}}$.

<table>
<tr><th rowspan="2">Category</th><th colspan="2">Fixed cost (R$)</th><th colspan="2">Variable cost</th><th rowspan="2">Type</th><th rowspan="2">Number of seats</th></tr>
<tr><th>Min</th><th>Max</th><th>Min</th><th>Max</th></tr>
<tr><td>Car</td><td>1500</td><td>2500</td><td>1.5</td><td>2.5</td><td>I</td><td>5</td></tr>
<tr><td rowspan="2">Minivan</td><td rowspan="2">3500</td><td rowspan="2">6000</td><td rowspan="2">3.0</td><td rowspan="2">4.0</td><td>I</td><td>7</td></tr>
<tr><td>II</td><td>8</td></tr>
<tr><td rowspan="3">Van</td><td rowspan="3">6000</td><td rowspan="3">9000</td><td rowspan="3">3.5</td><td rowspan="3">4.5</td><td>I</td><td>12</td></tr>
<tr><td>II</td><td>16</td></tr>
<tr><td>III</td><td>20</td></tr>
<tr><td rowspan="2">Microbus</td><td rowspan="2">8000</td><td rowspan="2">11000</td><td rowspan="2">4.0</td><td rowspan="2">5.5</td><td>I</td><td>22</td></tr>
<tr><td>II</td><td>27</td></tr>
<tr><td rowspan="3">Bus</td><td rowspan="3">12000</td><td rowspan="3">18000</td><td rowspan="3">4.5</td><td rowspan="3">6.0</td><td>I</td><td>32</td></tr>
<tr><td>II</td><td>42</td></tr>
<tr><td>III</td><td>55</td></tr>
</table>

Table 3.2: List of categories for all type of vehicles

Due to the lack of a fleet of vehicles, third-party transportation companies are responsible for meeting the demand through three types of vehicle lease agreements. In this way, each contract is linked to a specific type of vehicle which must belong to one of the categories described in Table 3.2. Each contract is generated arbitrarily following the restrictions of the category of vehicles linked to the respective lease agreement. The number of contracts created by instance is defined according to the sum of available seats for all lease agreements which, in turn, must be five to six times the number of passengers.

3.4.2 Analysis

Tables 3.3-3.8 present detailed results for each dataset. On these tables, column **Instance** depicts the name of the instance, in the following format: `(dataset)n(number of groups )_(instance ID)`. The following values depend on the number of executions, which was set to 30. Column **BKS Cost** (R$) presents the best-known solution cost for the instance; column **Avg Cost** (R$) presents the average solution cost for the instance; column **St. Dev. Cost** (R$) presents the standard deviation for solution cost on each instance; column **Avg Gap** (%) presents the average gap on each instance, where the gap is calculated as $gap = 100 \cdot (avg\ cost\ - BKS)/\ BKS$; column **Avg TB** (s) presents the average time (on 30 executions) to reach the best-known solution for the instance. In order to provide real-time decisions, a strict time limit of 60 seconds was imposed as the stopping criteria for the optimization process . The time limit for reach the BKS solution was extended for 300 seconds.

On Table 3.3, with 100 passengers and ordered by **Avg TB**, the first 16 instances with the smaller number of groups and few other instances present a zero standard deviation and zero gaps over the BKS, indicating a very stable search behavior. The most significant gap is 7.86% for instance 16 (number of groups 89), although the average gap for this dataset was superficial, around 0.91%. The average time to reach the best-known solution was 114 seconds.

On Tables 3.4-3.8, ranging from 250 to 1500 passengers, a similar behavior can be found, with increasing average gaps (ranging from 2.4%, 3.9%, 4.2%, 4.3% and 4.8%, respectively) as the number of passengers and groups are increased. As expected, due to the time limit set to 300 seconds, an average TB of 150 seconds is found on most classes.

In a preliminary phase, customers are clustered in groups according to QoS constraints related to the distance between their residences and the respective bus stop associated with the round-trip. Although this step is not in the scope of the proposed methodology, it directly affects the performance of the algorithm. This behavior can be seen in Table 3.3,

Table 3.3: Numerical results for dataset i100

Instance	BKS Cost (R$)	Avg Cost (R$)	St. Dev. Cost (R$)	Avg Gap (%)	Avg TB (s)
i100n10_38	59826.60	59826.601	0.000	0.0	0.0
i100n10_39	33066.92	33066.921	0.000	0.0	0.0
i100n14_41	28729.99	28729.998	0.000	0.0	0.0
i100n17_6	36052.92	36052.921	0.000	0.0	0.1
i100n13_57	14276.99	14276.999	0.000	0.0	0.2
i100n11_56	56735.75	56735.753	0.000	0.0	0.3
i100n13_60	44397.79	44397.796	0.000	0.0	0.3
i100n15_4	33929.00	33929.000	0.000	0.0	0.4
i100n12_24	18655.00	18655.000	0.000	0.0	0.4
i100n17_37	71974.22	71974.226	0.000	0.0	0.6
i100n22_23	23045.31	23045.314	0.000	0.0	1.1
i100n14_20	28852.00	28852.000	0.000	0.0	1.1
i100n17_5	45888.19	45888.199	0.000	0.0	1.6
i100n18_3	32628.99	32628.998	0.000	0.0	15.2
i100n16_22	19444.46	19444.466	0.000	0.0	25.9
i100n17_58	16847.69	16847.699	0.000	0.0	27.9
i100n18_55	37868.02	37875.401	40.412	0.0	47.3
i100n95_31	57334.30	57363.333	37.346	0.0	55.6
i100n49_29	65197.71	65197.714	0.000	0.0	65.9
i100n87_70	47743.83	47744.856	0.849	0.0	75.6
i100n22_59	30067.20	30068.426	4.647	0.0	110.0
i100n27_19	33211.53	33234.842	50.021	0.0	114.0
i100n56_8	68942.90	70938.717	737.407	2.8	121.1
i100n26_42	47418.30	47418.482	0.506	0.0	121.6
i100n61_26	89368.56	89386.111	18.258	0.0	123.1
i100n90_71	71649.25	72446.783	314.469	1.1	123.9
i100n87_72	53515.78	54228.305	211.920	1.3	126.8
i100n28_2	80767.27	80875.551	119.295	0.1	131.6
i100n64_43	85562.59	87623.287	976.888	2.4	133.3
i100n47_65	62547.53	62725.173	95.806	0.2	133.4
i100n84_15	37424.00	38133.164	397.918	1.8	137.5
i100n51_25	34564.42	34583.466	12.589	0.0	137.8
i100n61_9	43055.99	44414.131	647.015	3.1	138.6
i100n58_44	138320.56	139237.397	537.650	0.6	139.4
i100n85_17	51583.38	53534.188	1092.349	3.7	140.9
i100n54_11	40538.00	40750.749	40.469	0.5	141.7
i100n95_50	130825.72	134684.467	1619.179	2.9	143.2
i100n62_61	132082.68	132675.879	309.067	0.4	144.1
i100n50_12	45871.39	46122.928	170.273	0.5	144.5
i100n52_28	13506.00	13506.000	0.000	0.0	146.2
i100n51_66	54139.84	54504.706	199.927	0.6	147.5
i100n86_35	55108.16	56139.510	568.112	1.8	148.2
i100n92_51	71101.24	72128.317	400.443	1.4	148.4
i100n89_16	37815.23	40788.676	969.590	7.8	148.9
i100n63_47	61404.10	62873.789	670.429	2.3	150.1
i100n90_13	72540.53	75225.686	989.684	3.7	150.7
i100n85_68	87946.08	88123.433	110.410	0.2	153.6
i100n87_18	41009.99	41135.729	478.494	0.3	154.8
i100n58_10	45783.01	45819.105	42.691	0.0	157.2
i100n62_46	44201.99	45067.998	804.252	1.9	158.6
i100n84_14	91168.00	94266.647	1051.648	3.3	158.8
i100n82_53	47870.15	48545.492	524.037	1.4	159.0
i100n54_48	80584.55	81814.195	625.069	1.5	160.1
i100n49_7	62403.46	63835.854	730.230	2.2	161.8
i100n91_49	97483.26	99022.463	866.947	1.5	161.9
i100n89_32	81654.71	82515.664	381.488	1.0	162.3
i100n91_67	28038.34	28039.792	0.822	0.0	163.4
i100n91_52	59722.14	61392.038	913.036	2.7	163.8
i100n95_33	18608.00	18608.000	0.000	0.0	164.2
i100n22_40	38278.36	38278.367	0.000	0.0	165.5
i100n47_45	50551.05	51501.376	471.774	1.8	167.4
i100n100_54	58238.38	59116.356	410.116	1.5	168.1
i100n58_30	41905.49	41955.808	32.115	0.1	170.0
i100n27_21	15513.94	15809.937	64.695	1.9	170.5
i100n62_27	18295.00	18411.872	105.085	0.6	172.8
i100n53_63	34478.50	35114.549	360.464	1.8	181.9
i100n89_69	174937.94	174939.856	1.207	0.0	183.1
i100n92_34	18342.43	18346.818	2.929	0.0	185.1
i100n100_36	42887.84	43167.253	168.466	0.6	186.0
i100n48_64	93737.43	93747.682	7.311	0.0	249.4
i100n59_62	183123.29	183405.423	144.048	0.1	283.8
Average	-	-	-	0.912	114.7

Table 3.4: Numerical results for dataset i250

Instance	BKS Cost (R$)	Avg Cost (R$)	St. Dev. Cost (R$)	Avg Gap (%)	Avg TB (s)
i250n246_88	97846.00	103233.188	2011.395	5.5	112.5
i250n35_127	115394.57	115986.211	415.656	0.5	117.8
i250n213_124	94810.12	96334.774	803.128	1.6	118.9
i250n38_74	94261.63	95760.172	564.205	1.5	124.3
i250n135_99	33844.83	35122.248	633.854	3.7	126.9
i250n143_100	24590.00	24963.967	701.446	1.5	128.2
i250n139_137	111310.36	113477.925	1100.208	1.9	129.0
i250n156_82	110634.73	112918.778	1377.711	2.0	130.3
i250n142_81	104478.09	105736.207	795.179	1.2	131.4
i250n207_108	66416.71	68201.774	923.780	2.6	132.9
i250n68_114	95334.851	97571.375	906.992	2.3	133.1
i250n132_117	88974.00	90537.019	996.417	1.7	133.9
i250n37_78	69629.99	69629.992	0.000	0.0	135.6
i250n63_111	83411.99	85126.396	781.745	2.0	135.8
i250n210_85	150153.59	155428.864	2250.437	3.5	136.6
i250n35_110	95459.21	96148.016	371.050	0.7	136.8
i250n151_84	87932.00	92481.159	1668.226	5.1	136.8
i250n58_112	82132.73	83715.843	778.739	1.9	137.6
i250n212_103	108995.69	110832.594	777.877	1.6	139.0
i250n67_128	165031.35	166132.574	549.394	0.6	139.9
i250n210_121	251596.96	256121.246	2249.147	1.7	140.2
i250n146_115	185776.57	190139.446	1881.044	2.3	140.4
i250n153_79	149845.75	152805.769	1417.238	1.9	140.5
i250n250_125	120773.12	126332.538	2064.979	4.6	141.2
i250n46_76	85090.92	86314.022	536.693	1.4	142.2
i250n214_105	28765.09	31925.056	1895.074	10.9	142.9
i250n229_143	165933.23	169576.746	1608.978	2.1	143.1
i250n115_116	147822.35	150968.927	1504.120	2.1	143.8
i250n209_126	170273.98	174710.648	2135.595	2.6	144.4
i250n130_138	98745.39	102243.098	1200.552	3.5	145.1
i250n52_91	74302.87	75291.543	445.944	1.3	145.2
i250n56_109	114149.13	115878.716	891.853	1.5	145.3
i250n64_93	30262.99	31756.998	506.513	4.9	145.5
i250n25_96	35364.41	35971.145	261.761	1.7	145.9
i250n151_133	237409.14	240192.527	1241.197	1.1	145.9
i250n132_101	62349.54	64037.452	851.067	2.7	146.1
i250n60_132	74946.79	76850.528	618.872	2.5	146.5
i250n125_80	119281.25	121539.301	1177.010	1.8	146.7
i250n226_107	143843.95	146099.222	1171.341	1.5	146.8
i250n53_113	98885.77	100045.666	739.130	1.1	146.9
i250n208_90	124985.91	130871.894	1948.833	4.7	147.5
i250n40_75	76360.54	76827.654	460.437	0.6	148.9
i250n31_94	27167.00	27167.000	0.000	0.0	149.0
i250n115_83	104423.09	107958.163	1475.016	3.3	151.0
i250n239_139	293156.37	298051.838	2016.749	1.6	151.8
i250n148_102	60570.32	62231.233	731.484	2.7	153.4
i250n228_122	283032.25	288741.865	3434.015	2.0	153.7
i250n56_77	81216.44	82964.172	833.878	2.1	154.7
i250n46_95	64901.90	66174.122	457.741	1.9	155.0
i250n241_106	35860.28	37987.072	1208.664	5.9	155.5
i250n226_141	65675.17	68786.386	1415.620	4.7	156.2
i250n63_130	25881.99	26402.815	201.911	2.0	156.2
i250n120_134	163651.56	167157.774	963.349	2.1	156.3
i250n37_73	79491.89	79811.672	203.183	0.4	157.2
i250n31_131	46431.33	46588.677	96.254	0.3	157.8
i250n126_118	90193.10	90994.920	621.010	0.8	159.7
i250n117_97	90331.46	92732.402	979.084	2.6	160.3
i250n208_86	196490.37	201828.720	1910.416	2.7	161.4
i250n148_120	134288.50	138375.138	1583.798	3.0	163.0
i250n238_123	129039.78	131766.794	1470.426	2.1	164.0
i250n236_87	111311.41	113791.340	1317.243	2.2	164.8
i250n244_142	44227.87	45997.344	772.951	4.0	166.8
i250n131_119	122259.65	125190.344	1081.034	2.3	167.8
i250n67_92	58882.76	60733.811	724.501	3.1	169.0
i250n240_104	128824.56	131996.695	1295.471	2.4	170.5
i250n160_136	44708.49	46224.710	860.285	3.3	170.9
i250n148_135	49107.00	52198.110	1267.957	6.2	174.0
i250n133_98	116417.63	119051.986	889.485	2.2	176.1
i250n238_89	94996.80	100546.491	1532.034	5.8	177.5
i250n41_129	31669.99	32246.463	373.890	1.8	179.4
i250n245_140	280559.87	289562.754	2443.825	3.2	179.7
i250n219_144	284901.74	284903.627	0.975	0.0	295.0
Average	-	-	-	2.496	150.1

Table 3.5: Numerical results for dataset i500

Instance	BKS Cost (R$)	Avg Cost (R$)	St. Dev. Cost (R$)	Avg Gap (%)	Avg TB (s)
i500n496_193	392826.06	402318.098	4115.552	2.4	116.6
i500n428_176	194569.57	199266.181	2207.929	2.4	121.0
i500n120_163	129388.29	132526.018	1643.092	2.4	125.7
i500n117_201	57160.00	59800.690	1376.427	4.6	127.9
i500n471_212	525443.43	532294.552	4345.211	1.3	132.0
i500n255_206	298393.59	302872.957	2444.850	1.5	132.1
i500n94_147	158208.00	162124.389	1683.061	2.4	132.1
i500n260_170	191448.64	195768.710	2293.806	2.2	132.4
i500n285_188	290745.18	297546.269	3139.201	2.3	133.1
i500n314_190	150353.01	156887.532	2475.801	4.3	133.7
i500n114_204	117677.82	120856.839	1261.334	2.7	133.7
i500n92_165	51039.00	56140.311	1677.905	9.9	135.0
i500n412_194	467706.75	477714.865	4697.403	2.1	135.1
i500n424_161	169356.00	184049.806	4596.633	8.6	138.1
i500n490_213	87929.00	96283.326	2526.187	9.5	138.6
i500n239_171	57376.00	62339.810	2119.512	8.6	139.5
i500n421_215	233532.75	239694.381	2489.706	2.6	139.8
i500n453_195	198958.98	202345.829	2660.219	1.7	140.9
i500n470_158	292161.37	301773.896	4376.617	3.2	141.4
i500n314_173	122108.36	126112.401	2012.775	3.2	141.5
i500n236_153	164149.15	172900.088	3751.514	5.3	142.1
i500n249_151	208232.03	211812.667	1866.708	1.7	143.6
i500n76_150	132556.01	135624.616	1165.628	2.3	144.1
i500n450_196	203268.00	214425.851	4069.588	5.4	144.7
i500n88_167	97371.45	100527.419	861.380	3.2	145.7
i500n68_149	145536.82	148689.907	1350.303	2.1	145.8
i500n60_181	202134.07	205503.442	1726.512	1.6	145.8
i500n302_210	203294.23	207964.574	2050.909	2.2	146.0
i500n54_184	143283.29	145736.214	1073.517	1.7	146.7
i500n283_191	191543.68	200210.797	3495.234	4.5	146.9
i500n71_200	176799.79	179969.926	1269.810	1.7	148.5
i500n109_183	154128.76	156524.781	1470.650	1.5	148.9
i500n105_185	166600.42	171812.397	1791.500	3.1	149.9
i500n277_187	340644.28	348437.826	3646.622	2.2	150.1
i500n464_198	254342.54	265668.973	4148.769	4.4	150.3
i500n302_189	182106.00	192202.953	4008.417	5.5	151.1
i500n69_148	143294.79	146640.847	1498.307	2.3	152.1
i500n435_160	190485.29	196199.490	2374.991	3.0	152.6
i500n124_202	58838.00	61296.216	1259.436	4.1	153.5
i500n98_168	75428.78	78781.701	1498.615	4.4	155.2
i500n281_152	238348.40	246034.580	3525.362	3.2	155.3
i500n466_175	212269.90	217262.078	2666.093	2.3	155.5
i500n417_214	86564.46	90579.037	2076.344	4.6	156.0
i500n469_159	199342.71	211810.755	3488.388	6.2	156.5
i500n300_172	66309.00	70755.590	2176.696	6.7	156.8
i500n489_178	76428.99	83336.907	2489.020	9.0	157.2
i500n413_157	254066.34	266799.214	3475.600	5.0	157.5
i500n80_146	174841.46	178490.259	1244.109	2.0	158.7
i500n235_156	171461.48	177653.982	2213.543	3.6	159.4
i500n248_174	111827.71	115655.593	1925.454	3.4	160.2
i500n125_182	273394.81	277662.963	2153.558	1.5	161.0
i500n414_179	133567.04	140418.216	2368.510	5.1	161.4
i500n278_209	190607.34	201994.065	3841.761	5.9	162.2
i500n314_169	211096.59	217306.381	2162.810	2.9	163.3
i500n446_177	75599.37	83612.566	3520.655	10.6	164.0
i500n116_145	186374.14	191676.760	2328.638	2.8	164.4
i500n470_216	254543.89	263913.395	4351.971	3.6	164.9
i500n119_199	196056.34	199581.626	1577.939	1.7	166.0
i500n257_155	180627.01	185420.707	2362.331	2.6	168.1
i500n497_180	109297.14	116633.252	3312.330	6.7	169.7
i500n238_207	80400.00	84978.902	2131.388	5.6	169.7
i500n240_192	210607.96	217512.721	2977.767	3.2	170.2
i500n230_205	246899.71	250260.776	1997.169	1.3	171.1
i500n443_162	201071.14	216036.654	7034.093	7.4	174.3
i500n75_186	139522.98	141337.733	1338.323	1.3	175.2
i500n57_203	99352.44	103733.546	1713.082	4.4	178.1
i500n491_211	455518.18	462251.110	4544.743	1.4	178.6
i500n75_164	110174.26	113933.215	1426.570	3.4	179.4
i500n250_154	143392.01	153320.717	3385.355	6.9	182.3
i500n94_166	48839.00	50615.049	1023.652	3.6	184.2
i500n230_208	65244.99	69357.230	1957.429	6.3	184.6
i500n484_197	238082.85	248468.354	5079.207	4.3	192.5
Average	-	-	-	3.913	152.7

Table 3.6: Numerical results for dataset i750

Instance	BKS Cost (R$)	Avg Cost (R$)	St. Dev. Cost (R$)	Avg Gap (%)	Avg TB (s)
i750n642_229	349930.62	360897.876	4551.591	3.1	112.4
i750n197_236	222012.64	225833.640	1369.165	1.7	115.2
i750n200_220	201908.98	207947.626	3074.242	2.9	122.8
i750n151_222	225919.89	230980.595	3038.368	2.2	127.2
i750n672_251	262371.59	270613.763	3671.150	3.1	127.6
i750n618_230	339914.56	348644.475	3904.885	2.5	127.7
i750n623_270	315107.75	324900.441	5135.637	3.1	127.7
i750n393_279	115588.29	121796.139	2668.283	5.3	130.9
i750n103_256	214103.71	219310.752	2045.595	2.4	130.9
i750n82_276	122865.13	125954.585	1449.926	2.5	132.3
i750n112_258	248083.32	253847.189	2290.119	2.3	133.1
i750n206_274	81311.42	88680.176	3449.549	9.0	134.4
i750n365_246	147198.68	154379.878	2810.552	4.8	134.6
i750n145_237	70431.00	77147.174	2536.574	9.5	135.5
i750n709_283	508411.46	520053.717	5063.670	2.2	135.8
i750n623_285	157158.00	169608.063	3998.861	7.9	135.9
i750n618_232	274691.00	293891.032	7698.585	6.9	136.4
i750n114_235	185984.45	189453.729	1817.036	1.8	137.6
i750n720_249	110663.61	124235.448	5515.759	12.2	138.2
i750n457_242	328967.28	337624.741	3859.204	2.6	138.4
i750n674_266	605454.93	626043.029	8213.319	3.4	138.5
i750n110_272	290089.78	295890.989	1916.940	2.0	139.2
i750n625_269	298970.59	312270.832	5881.748	4.4	140.0
i750n185_217	270848.09	276575.565	2762.858	2.1	140.7
i750n739_233	293983.06	308698.427	6343.748	5.0	141.1
i750n637_252	208952.82	218214.791	4093.392	4.4	142.8
i750n205_254	403614.71	428751.286	6835.381	6.2	143.2
i750n399_244	79747.00	86490.984	2978.753	8.4	145.6
i750n351_260	440280.62	451567.286	5120.073	2.5	146.8
i750n694_265	538233.12	555521.423	7214.145	3.2	146.8
i750n402_281	249623.34	262246.688	4563.741	5.0	147.1
i750n351_282	266050.93	278613.854	4082.782	4.7	147.9
i750n368_223	308710.90	320108.993	4304.928	3.6	148.2
i750n185_255	234509.89	241570.149	2575.853	3.0	148.4
i750n458_261	246020.98	255374.003	4312.071	3.8	148.6
i750n713_286	190711.71	205593.112	5944.904	7.8	151.6
i750n425_278	466528.81	476717.062	3746.441	2.1	153.0
i750n439_259	466023.25	478671.443	5782.316	2.7	153.2
i750n388_245	174170.50	179719.179	2834.437	3.1	153.3
i750n154_240	110682.67	117262.000	2385.579	5.9	154.5
i750n475_228	245757.18	266313.614	8501.002	8.3	154.8
i750n354_226	258863.57	266118.422	3928.250	2.8	155.3
i750n669_268	263448.34	283131.299	7937.474	7.4	155.3
i750n451_224	308879.31	317699.541	3681.249	2.8	156.2
i750n434_227	265142.18	274216.342	3320.892	3.4	156.3
i750n141_271	270870.40	274518.135	1484.983	1.3	156.3
i750n121_253	293086.31	298947.909	2545.104	2.0	156.5
i750n460_241	285799.25	293983.069	2934.032	2.8	156.6
i750n740_234	259783.98	278498.658	6920.227	7.2	157.5
i750n617_247	358842.21	365524.335	2981.231	1.8	158.7
i750n103_275	131943.79	136418.559	1869.622	3.3	158.8
i750n648_287	300810.53	310650.515	4472.940	3.2	159.1
i750n389_263	304099.40	314084.838	3458.338	3.2	159.6
i750n734_248	386491.90	393817.555	4205.486	1.8	162.0
i750n76_239	82227.71	86953.467	1672.371	5.7	162.7
i750n380_264	309256.18	324877.530	5778.394	5.0	162.8
i750n392_243	90860.00	99568.053	4323.627	9.5	164.7
i750n151_221	217885.81	224059.244	2648.043	2.8	165.5
i750n438_277	440442.75	451884.111	3763.757	2.5	168.4
i750n709_284	595658.18	616878.502	9248.193	3.5	168.4
i750n733_288	312539.28	321705.511	3756.546	2.9	168.5
i750n166_257	217703.90	226227.364	2839.961	3.9	170.1
i750n210_273	97699.36	104707.278	2394.894	7.1	170.8
i750n634_250	104740.99	113798.427	4274.314	8.6	172.0
i750n693_231	293846.09	300624.746	4423.161	2.3	172.5
i750n75_219	203575.87	209222.245	2361.568	2.7	172.9
i750n385_225	236057.98	243751.849	2745.052	3.2	173.1
i750n202_238	69539.00	74616.264	2388.051	7.3	173.5
i750n125_218	289212.31	293642.917	2352.335	1.5	176.4
i750n345_280	99804.99	104618.038	2472.821	4.8	180.1
i750n726_267	322078.21	329211.243	3452.877	2.2	180.8
i750n420_262	236202.00	245042.964	3745.630	3.7	185.2
Average	-	-	-	4.236	150.5

Table 3.7: Numerical results for dataset i1000

Instance	**BKS** **Cost** (R$)	**Avg** **Cost** (R$)	**St. Dev.** **Cost** (R$)	**Avg** **Gap** (%)	**Avg** **TB** (s)
i1000n584_315	103846.10	120798.492	7148.908	16.3	118.0
i1000n201_308	213939.17	222094.870	2897.508	3.8	120.0
i1000n213_312	169773.57	177833.598	3733.112	4.7	120.0
i1000n831_301	494790.71	518018.760	9675.074	4.6	121.3
i1000n196_289	330957.46	340336.775	4356.152	2.8	122.6
i1000n628_351	151414.06	164311.857	5184.798	8.5	125.8
i1000n174_293	302896.40	308517.227	2952.849	1.8	126.6
i1000n174_347	240430.06	251517.534	3938.317	4.6	128.1
i1000n532_336	401316.12	420284.726	9049.702	4.7	129.6
i1000n226_291	274275.15	286317.724	5530.707	4.3	130.9
i1000n134_307	223772.10	228042.591	2209.834	1.9	131.0
i1000n500_299	378615.46	391221.910	5833.482	3.3	131.3
i1000n140_309	84300.00	87955.863	2030.279	4.3	134.5
i1000n111_326	338258.40	346208.621	2811.455	2.3	135.0
i1000n172_327	311506.96	318961.016	2731.940	2.3	135.1
i1000n922_337	753397.31	788132.052	14877.907	4.6	135.3
i1000n590_295	430717.90	445341.073	8080.990	3.3	136.5
i1000n984_339	369521.43	391176.174	9314.094	5.8	137.2
i1000n922_302	497321.65	509489.736	6302.289	2.4	137.2
i1000n504_334	313249.96	322207.748	4237.556	2.8	137.7
i1000n277_310	113283.00	119569.906	3690.949	5.5	138.6
i1000n469_352	143174.00	147953.315	2604.713	3.3	138.9
i1000n894_342	564665.93	582290.208	9825.743	3.1	139.7
i1000n829_305	367000.15	375598.095	5367.534	2.3	140.8
i1000n911_357	165790.98	182893.003	8225.754	10.3	141.0
i1000n938_304	343192.06	364186.733	10365.687	6.1	141.7
i1000n852_355	684743.75	702376.333	7562.330	2.5	142.0
i1000n569_314	346481.78	356266.455	4800.690	2.8	143.4
i1000n480_300	281973.00	292542.203	5762.274	3.7	143.8
i1000n851_340	361945.78	379256.095	7010.054	4.7	144.2
i1000n494_317	195202.54	205262.722	3266.548	5.1	144.2
i1000n555_331	560741.75	580139.314	6425.428	3.4	144.4
i1000n146_294	307259.56	313875.931	3411.633	2.1	146.5
i1000n887_321	164279.70	178613.976	9683.670	8.7	146.8
i1000n265_329	319232.00	325411.745	3376.147	1.9	146.8
i1000n506_296	444190.53	455723.187	4506.744	2.5	146.9
i1000n460_354	344788.46	358597.414	4486.500	4.0	147.7
i1000n877_319	446957.03	456934.431	4621.502	2.2	147.7
i1000n231_343	372888.56	385692.411	5048.428	3.4	149.2
i1000n470_318	221216.76	230454.639	3874.001	4.1	150.3
i1000n895_324	323030.78	334804.641	5200.211	3.6	152.5
i1000n135_345	98958.17	104917.128	2775.538	6.0	154.2
i1000n998_360	382422.75	397535.757	5795.342	3.9	154.5
i1000n483_349	531910.50	539716.214	4056.568	1.4	154.6
i1000n833_322	129730.00	152690.573	7739.423	17.6	155.9
i1000n551_313	359995.93	369863.454	4673.647	2.7	157.0
i1000n893_356	866952.68	887793.985	8098.567	2.4	157.1
i1000n874_338	718579.31	739422.425	9786.524	2.9	157.6
i1000n877_358	158135.62	168536.455	5018.141	6.5	158.0
i1000n470_316	96261.47	103961.326	3928.521	7.9	159.2
i1000n887_303	371910.03	390455.277	7923.368	4.9	159.3
i1000n935_323	267255.96	275399.639	4539.150	3.0	159.5
i1000n202_290	352864.90	363238.371	3185.720	2.9	160.3
i1000n136_348	190323.39	194992.238	2549.237	2.4	160.9
i1000n495_350	579323.06	592588.964	4792.853	2.2	161.6
i1000n640_353	330879.59	341319.495	3698.120	3.1	162.0
i1000n178_344	365239.43	373580.427	4594.822	2.2	162.8
i1000n231_325	393177.56	407441.378	5465.008	3.6	163.0
i1000n844_320	386959.56	403380.709	5444.793	4.2	163.2
i1000n520_298	315002.96	333447.651	8190.985	5.8	163.9
i1000n524_332	615773.12	644944.514	11615.810	4.7	165.0
i1000n130_292	291243.37	296381.357	2611.441	1.7	166.9
i1000n593_297	347089.65	357382.481	6305.732	2.9	166.9
i1000n578_333	333311.31	352112.691	5834.845	5.6	167.5
i1000n983_341	420938.59	437363.160	6630.649	3.9	167.9
i1000n971_306	399629.03	416300.512	6873.460	4.1	168.2
i1000n157_328	312533.21	319860.982	3688.713	2.3	169.6
i1000n133_311	100201.65	106278.427	2455.938	6.0	170.1
i1000n143_346	103527.00	108346.125	2371.819	4.6	170.6
i1000n607_335	375781.53	396860.935	6236.309	5.6	171.5
i1000n138_330	315405.37	325317.651	4452.980	3.1	173.7
i1000n834_359	321677.37	328162.808	3516.884	2.0	191.2
Average	-	-	-	4.304	148.7

Table 3.8: Numerical results for dataset i1500

Instance	BKS Cost (R$)	Avg Cost (R$)	St. Dev. Cost (R$)	Avg Gap (%)	Avg TB (s)
i1500n1436_392	588057.68	602547.631	6616.289	2.4	105.8
i1500n1237_428	855928.25	878306.956	9205.395	2.6	113.7
i1500n359_417	159624.00	171944.324	4791.510	7.7	117.8
i1500n1382_394	220386.98	235760.914	8181.773	6.9	120.8
i1500n810_422	689613.06	723101.762	8381.196	4.8	124.8
i1500n1500_412	527732.18	566396.079	15419.807	7.3	126.0
i1500n701_371	492255.96	510164.636	8958.454	3.6	126.5
i1500n154_419	268512.43	283117.558	4970.388	5.4	127.8
i1500n164_361	454355.56	462039.728	5065.846	1.6	129.1
i1500n175_416	387083.71	399209.988	5198.811	3.1	129.6
i1500n231_381	127321.99	139893.551	4820.481	9.8	130.2
i1500n776_405	509448.59	536708.126	7827.955	5.3	135.1
i1500n720_403	697186.31	711523.693	7736.403	2.0	136.1
i1500n249_363	395976.46	416172.947	7857.249	5.1	136.6
i1500n1268_414	586186.62	621043.946	10670.764	5.9	136.7
i1500n1449_432	568523.56	583058.077	8141.843	2.5	138.0
i1500n395_420	520826.84	530796.090	4703.471	1.9	140.0
i1500n1409_431	605039.18	623692.396	8491.025	3.0	140.1
i1500n894_407	632259.93	660261.948	10883.564	4.4	140.3
i1500n822_385	553996.75	571324.685	8238.134	3.1	141.2
i1500n1282_373	627133.31	649269.662	10477.017	3.5	141.3
i1500n1416_410	978566.50	1010568.375	14204.618	3.2	142.2
i1500n780_369	468106.21	494165.208	9326.483	5.5	142.8
i1500n1257_413	620087.50	649426.166	13020.993	4.7	143.2
i1500n1476_411	496764.03	560999.112	23523.234	12.9	143.9
i1500n875_421	951573.87	982254.750	12209.598	3.2	144.0
i1500n1456_391	649861.18	664270.373	7923.973	2.2	145.9
i1500n1465_396	442565.12	453877.520	6436.416	2.5	147.7
i1500n1334_429	248781.09	272216.998	11553.349	9.4	149.1
i1500n322_379	348538.87	361334.776	4095.052	3.6	150.7
i1500n1496_395	375608.06	392078.683	7585.180	4.3	150.8
i1500n330_382	135038.00	147465.248	4252.332	9.2	151.2
i1500n745_367	554962.31	589578.575	12593.826	6.2	151.6
i1500n947_390	312968.53	336837.905	9203.090	7.6	151.8
i1500n1264_393	197082.98	221713.684	11296.639	12.4	152.2
i1500n1458_377	534376.25	587854.435	14675.563	10.0	153.7
i1500n1274_430	265470.90	279450.413	7447.223	5.2	154.3
i1500n837_386	560770.00	571439.375	5435.250	1.9	156.6
i1500n760_406	473239.06	500158.032	9756.454	5.6	156.7
i1500n1372_427	1031352.37	1064228.810	12176.147	3.1	157.0
i1500n352_401	543289.68	554404.050	4700.135	2.0	157.1
i1500n833_423	201776.46	218292.802	8593.518	8.1	157.2
i1500n302_384	257955.06	268023.242	5206.117	3.9	157.6
i1500n1299_375	547669.68	574942.706	14507.250	4.9	157.7
i1500n720_388	154442.00	165817.908	7115.172	7.3	159.6
i1500n195_415	372204.25	382577.646	5017.059	2.7	159.8
i1500n169_383	145354.00	154978.936	4136.897	6.6	160.1
i1500n833_408	588760.43	609885.806	10095.182	3.5	160.2
i1500n152_365	434991.78	450059.665	6491.701	3.4	160.4
i1500n403_380	371117.71	382428.509	4718.174	3.0	161.1
i1500n1473_378	598230.50	620545.158	11196.337	3.7	161.6
i1500n953_370	479559.31	504452.775	10402.167	5.1	162.1
i1500n1443_409	1122089.62	1156107.254	15673.498	3.0	165.5
i1500n319_366	453851.31	463969.569	5655.177	2.2	166.0
i1500n902_426	605301.81	630999.908	10371.279	4.2	166.5
i1500n764_424	213537.98	224803.192	6727.560	5.2	167.0
i1500n809_425	550661.50	562254.629	7074.753	2.1	167.1
i1500n411_397	650765.62	659628.998	5706.160	1.3	167.3
i1500n773_389	255295.45	264612.906	4943.827	3.6	169.1
i1500n157_364	430385.93	444923.858	5210.977	3.3	169.3
i1500n1444_376	555882.43	592756.298	17051.777	6.6	169.6
i1500n861_372	480640.71	506534.439	9809.659	5.3	170.7
i1500n926_387	201040.98	219575.299	6726.582	9.2	170.7
i1500n281_362	514404.53	526532.410	7058.332	2.3	170.9
i1500n951_404	828581.87	876753.937	16124.919	5.8	171.6
i1500n241_399	431769.37	450623.064	7039.546	4.3	172.2
i1500n298_402	478027.06	499608.966	7794.579	4.5	173.7
i1500n358_398	629980.31	649244.575	7280.632	3.0	175.6
i1500n910_368	562545.00	590334.589	11127.738	4.9	177.0
i1500n393_400	438174.28	462404.352	10895.204	5.5	178.1
i1500n1309_374	761489.00	794780.081	12688.487	4.3	179.3
i1500n173_418	132383.98	141090.670	4198.966	6.5	195.2
Average	-	-	-	4.852	151.9

where instances with smaller group sizes tend to be solved much faster than the others.

A Pearson Correlation test was performed on Table 3.9, indicating a strong correlation between group count and average TB for 100 passenger instance class. For dataset $i100$, a `p-value` of 2.67×10^{-11} was obtained on Pearson Correlation test, showing a correlation value of 0.6870 for the number of groups and the average time to reach the best solution. It is worth mentioning that this correlation does not occur in the other instance classes with more groups because a slightly more significant number of groups already pushes the average TB over the maximum time limit. Column **Uniform TB** indicates the result of a Kolmogorov-Smirnov test [13], checking if the TB values (time to best) belong to a uniform distribution between the minimum (zero) and maximum times (time limit set to 300). Only for the dataset $i100$, it was possible to discard the null hypothesis, indicating that this is not a uniform distribution.

Table 3.9: Summary for results on each dataset

Dataset	**Avg Gap (%)**	**Avg TB (%)**	**Avg Cost Coef. Var. (%)**	**Pearson Correlation**	**Uniform TB** `p-value`
i100	0.912	114.7	0.437	0.687	$< 2.2 \times 10^{-16}$
i250	2.496	150.1	1.111	-	≥ 0.05
i500	3.913	152.7	1.574	-	≥ 0.05
i750	4.236	150.5	1.676	-	≥ 0.05
i1000	4.304	148.7	1.808	-	≥ 0.05
i1500	4.852	151.9	1.959	-	≥ 0.05

Figure 3.3 also indicates that the TB value (time to best) tends not to follow a uniform distribution for dataset $i100$, that comprises the smaller number of groups. The situation is very different for bigger datasets, where time to best is uniformly distributed between 0 and the maximum time limit of 300 seconds (see Figure 3.4 for dataset $i250$).

3.4.3 Visualization of routes

The cost between a given pair of nodes is usually modeled by a simplification of the reality, considering the Euclidean distance of two points or a time to move through a given distance in a constant speed. However, route planning can become a hard task when access ways, and street directions are also considered. Several tools are found with the aim of solving distinct problems related to route planning, acquisition of geographic informations, route visualization, and real-time navigation. Among the existing services, the module developed in this paper also focused on the presentation of the route for users, data collection and route planning[2].

[2]This visualization service integration was developed using open-source libraries, as an alternative to the existing proprietary system on the company.

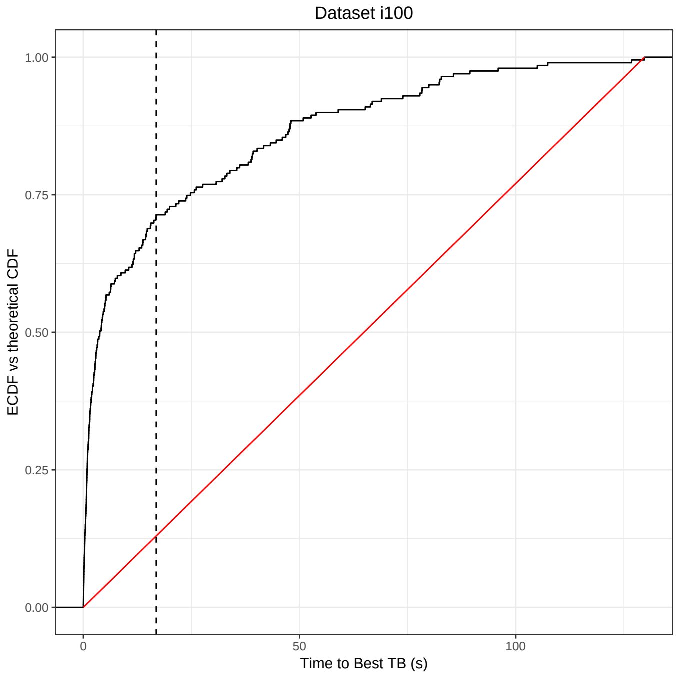

Figure 3.3: ECDF versus theoretical CDF (in red) for Instance $i100$. Dotted line indicates the maximum difference between the empirical and theoretical distributions. Time to Best TB does not form a uniform distribution over the available execution time.

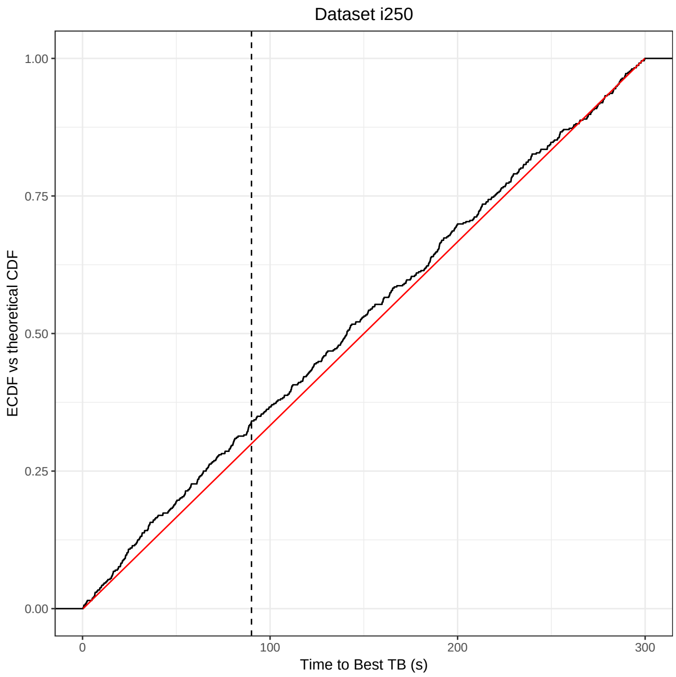

Figure 3.4: ECDF versus theoretical CDF (in red) for Instance $i250$. Dotted line indicates the maximum difference between the empirical and theoretical distributions. Time to Best TB forms a uniform distribution over the available execution time.

In order to provide route visualization easily, a novel system which integrates the mentioned services is developed and incorporated to the routing module. The service is implemented through the Open Source Routing Machine (OSRM). In turn, the OSRM acts on routing planning using an algorithm based on Contraction hierarchies [38], which uses open data provided by users of OpenStreetMap. Some techniques are used to improve the performance of routing for the shortest path using precomputed versions of the graph. Figure 3.5 depicts an artificial instance involving the transportation of employees to a workplace in Niteroi. The idea is to illustrate the output of the system when Euclidean data is considered. The tool is an extension of the previous work on the `cvrp-draw` library [50] [3].

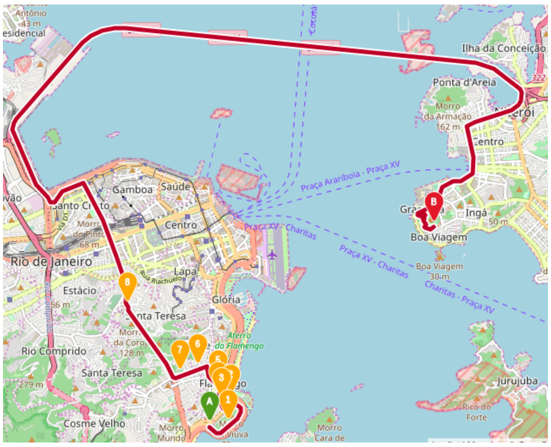

Figure 3.5: Outward route visualization for an instance. The green and red markers denote, respectively, a fake starting point and the workplace, while the yellow markers represent the pickup bus stops.

3.5 Conclusions

In this paper is presented an optimization module for dealing with a Vehicle Routing Problem for Transportation of Employees (VRPTE) of an energy industry, which can be seen as a School Bus Routing Problem (SBRP) variant considering real-time decisions. Additionally, Quality of Service (QoS) demands denoted by the company are formulated as a Logistic Service Level Agreement (LSLA), imposing restrictions such as time constraints for picking up/delivering employees; a restriction to impose the same order of pickups and deliveries in outward and return routes; and a radial constraint for avoiding passengers to be "upset" by getting close to the company and then having to wait for more pickups. It is also proposed an integration of the routing module with a system based on the tools *Open Source Routing Machine* and *OpenStreetMaps*, in order to develop a free of charge visualization mechanism for present graphically the solutions.

[3] cvrp-draw on GitHub: https://github.com/hugbro/cvrp-draw

The developed heuristic is based on the state-of-the-art ILS-RVND algorithm that managed to solve several vehicle routing problems with a heterogeneous fleet in the literature. The computational performance of the algorithm is tested on instances created artificially by a novel dataset generator based on real data provided by the company. Numerical experiments with the proposed algorithm indicated that it was capable of finding reasonable solutions that minimized operational costs for a significant number of employees (up to 1500). Future perspectives include validating the mathematical programming model and accelerating the algorithm within the given time limit.

Chapter 4

Conclusions

In this work, two different logistics problems were presented, both associated with applications in smart cities. For the Prize-Collecting Path Problem, a new NP-completeness proof was introduced, as well as an in-depth analysis was performed to understand some aspects for special graph classes. Moreover, a polynomial FPT-algorithm for graphs with bounded treewidth was presented. The ongoing investigations will consist of the development of the proposed algorithm and the generation of datasets based on the complexity of the analyzed graph classes.

At the same time, for the Vehicle Routing Problem for Transportation of Employees, a multi-start ILS-RVND algorithm was developed to solve a logistic problem in an energy industry. A mathematical model was formulated, and a new dataset generator, based on characteristics of real scenarios, was proposed. In order to investigate the performance of the algorithm, numerical experiments were performed for each dataset. The results indicated the capacity of the algorithm in finding good solutions in a reasonable time. Future works will lead to the improvement of the visualization module for routes, and the development of new neighborhoods to be incorporated into the local search phase of the algorithm.

Acknowledgment

The authors thank the Brazilian funding agencies CNPq and FAPERJ for supporting the development of this work. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] Akhmedov, M.; Kwee, I.; Montemanni, R. A matheuristic algorithm for the Prize-collecting Steiner Tree Problem. In *2015 3rd International Conference on Information and Communication Technology, ICoICT 2015* (may 2015), IEEE, pp. 408–412.

[2] Andrew, A. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters 9*, 5 (dec 1979), 216–219.

[3] Arnborg, S.; Corneil, D. G.; Proskurowski, A. Complexity of Finding Embedding in a k-Tree. *SIAM J Algebra Discr 8*, 2 (apr 1987), 277–284.

[4] Bienstock, D.; Goemans, M. X.; Simchi-Levi, D.; Williamson, D. A note on the prize collecting traveling salesman problem. *Mathematical Programming 59*, 1-3 (1993), 413–420.

[5] Bodlaender, H.; Jansen, K. On the complexity of the maximum cut problem. In *Stacs 94*, vol. 3075. 1994, pp. 769–780.

[6] Bodlaender, H. L. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM Journal on Computing 25*, 6 (dec 1996), 1305–1317.

[7] Bodlaender, H. L.; Drange, P. G.; Dregi, M. S.; Fomin, F. V.; Lokshtanov, D.; Pilipczuk, M. An O(ckn) 5-approximation algorithm for treewidth. In *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS* (oct 2013), IEEE, pp. 499–508.

[8] Canuto, S. A.; Resende, M. G. C.; Ribeiro, C. C. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks 38*, 1 (2001), 50–58.

[9] Cattaruzza, D.; Absi, N.; Feillet, D.; Vigo, D. An iterated local search for the multi-commodity multi-trip vehicle routing problem with time windows. *Computers & Operations Research 51* (nov 2014), 257–267.

[10] Chapovska, O.; Punnen, A. P. Variations of the prize-collecting steiner tree problem. *Networks 47*, 4 (jul 2006), 199–205.

[11] da Cunha, A. S.; Lucena, A.; Maculan, N.; Resende, M. G. C. A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs. *Discrete Applied Mathematics 157*, 6 (mar 2009), 1198–1217.

[12] Dantzig, G. B.; Ramser, J. H. The Truck Dispatching Problem. *Management Science 6*, 1 (oct 1959), 80–91.

[13] Darling, D. A. The Kolmogorov-Smirnov, Cramer-von Mises Tests. *The Annals of Mathematical Statistics 28*, 4 (dec 1957), 823–838.

[14] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 1 (dec 1959), 269–271.

[15] Dourisboure, Y.; Gavoille, C. Tree-decompositions with bags of small diameter. *Discrete Mathematics 307*, 16 (jul 2007), 2008–2029.

[16] Duffin, R. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications 10*, 2 (apr 1965), 303–318.

[17] E_K, B. *The design and analysis of computer algorithms*. Addison-Wesley, Reading, Mass., 1974.

[18] Fomin, F. V.; Grandoni, F.; Kratsch, D. Faster steiner tree computation in polynomial-space. In *Algorithms - ESA 2008* (Berlin, Heidelberg, 2008), D. Halperin and K. Mehlhorn, Eds., Springer Berlin Heidelberg, pp. 430–441.

[19] Gavril, F. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B 16*, 1 (feb 1974), 47–56.

[20] Goemans, M. X.; Williamson, D. P. A General Approximation Technique for Constrained Forest Problems. *SIAM Journal on Computing 24*, 2 (apr 1995), 296–317.

[21] Gutiérrez, A. M.; Cassales Marquezan, C.; Resinas, M.; Metzger, A.; Ruiz-Cortés, A.; Pohl, K. Extending WS-Agreement to Support Automated Conformity Check on Transport and Logistics Service Agreements. In *Proceedings of the 11th International Conference on Service-Oriented Computing - Volume 8274* (New York, NY, USA, 2013), ICSOC 2013, Springer-Verlag New York, Inc., pp. 567–574.

[22] Hasle, G.; Kloster, O. Industrial Vehicle Routing. In *Geometric Modelling, Numerical Simulation, and Optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 397–435.

[23] Hung, R. W.; Chang, M. S. Linear-time algorithms for the Hamiltonian problems on distance-hereditary graphs. *Theoretical Computer Science 341*, 1-3 (sep 2005), 411–440.

[24] Itai, A.; Papadimitriou, C. H.; Szwarcfiter, J. L. Hamilton Paths in Grid Graphs. *SIAM Journal on Computing 11*, 4 (nov 1982), 676–686.

[25] Johnson, D. S.; Minkoff, M.; Phillips, S. The prize collecting Steiner tree problem: theory and practice. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms* (2000), D. B. Shmoys, Ed., ACM/SIAM, pp. 760–769.

[26] Jones, M.; Lokshtanov, D.; Ramanujan, M. S.; Saurabh, S.; Suchý, O. Parameterized complexity of directed steiner tree on sparse graphs. In *Algorithms – ESA 2013* (Berlin, Heidelberg, 2013), H. L. Bodlaender and G. F. Italiano, Eds., Springer Berlin Heidelberg, pp. 671–682.

[27] Karak, A.; Abdelghany, K. The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transportation Research Part C: Emerging Technologies 102*, September 2018 (2019), 427–449.

[28] Karp, R. M. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer US, Boston, MA, 2010, pp. 219–241.

[29] Keshavarz-Kohjerdi, F.; Bagheri, A.; Asgharian-Sardroud, A. A linear-time algorithm for the longest path problem in rectangular grid graphs. *Discrete Applied Mathematics 160*, 3 (feb 2012), 210–217.

[30] Kinnersley, N. G. The vertex separation number of a graph equals its path-width. *Information Processing Letters 42*, 6 (jul 1992), 345–350.

[31] Laporte, G. Fifty years of vehicle routing. *Transportation Science 43*, 4 (2009), 408–416.

[32] Leksakul, K.; Smutkupt, U.; Jintawiwat, R.; Phongmoo, S. Heuristic approach for solving employee bus routes in a large-scale industrial factory. *Advanced Engineering Informatics 32* (apr 2017), 176–187.

[33] Lenstra, J. K.; Kan, A. H. G. R. Complexity of vehicle routing and scheduling problems. *Networks 11*, 2 (1981), 221–227.

[34] Li, L. Y. O.; Fu, Z. The school bus routing problem: a case study. *Journal of the Operational Research Society 53*, 5 (may 2002), 552–558.

[35] Liebig, T.; Piatkowski, N.; Bockermann, C.; Morik, K. Dynamic route planning with real-time traffic predictions. *Information Systems 64* (mar 2017), 258–265.

[36] Lourenço, H. R.; Martin, O. C.; Stützle, T.; Stutzle, T.; Stützle, T. Iterated Local Search: Framework and Applications. In *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds., vol. 146. Springer US, Boston, MA, 2010, pp. 363–397.

[37] Lucena, A.; Resende, M. G. C. Strong lower bounds for the prize collecting Steiner problem in graphs. *Discrete Applied Mathematics 141*, 1-3 (may 2004), 277–294.

[38] Luxen, D.; Vetter, C. Real-time routing with OpenStreetMap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '11* (New York, New York, USA, 2011), ACM Press, p. 513.

[39] Marquezan, C. C.; Metzger, A.; Franklin, R.; Pohl, K. Runtime Management of Multi-level SLAs for Transport and Logistics Services. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 560–574.

[40] Nederlof, J. Fast polynomial-space algorithms using möbius inversion: Improving on steiner tree and related problems. In *Automata, Languages and Programming* (Berlin, Heidelberg, 2009), S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. Nikoletseas, and W. Thomas, Eds., Springer Berlin Heidelberg, pp. 713–725.

[41] Palhazi Cuervo, D.; Goos, P.; Sörensen, K.; Arráiz, E. An iterated local search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research 237*, 2 (sep 2014), 454–464.

[42] Park, J.; Kim, B.-I. I. The school bus routing problem: A review. *European Journal of Operational Research 202*, 2 (apr 2010), 311–319.

[43] Pei, Y.; Zaiane, O. A Synthetic Data Generator for Clustering and Outlier Analysis. Tech. rep., University of Alberta, Edmonton, CA, 2006.

[44] Penna, P. H. V.; Subramanian, A.; Ochi, L. S.; Vidal, T.; Prins, C. A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. *Annals of Operations Research 273*, 1-2 (feb 2019), 5–74.

[45] Rolnik, R.; Klintowitz, D. (Im)Mobility in the city of São Paulo. *Estudos Avançados 25*, 71 (apr 2011), 89–108.

[46] Schittekat, P.; Kinable, J.; Sorensen, K.; Sevaux, M.; Spieksma, F.; Springael, J. A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research 229*, 2 (sep 2013), 518–528.

[47] Segev, A. The nodeâweighted steiner tree problem. *Networks 17*, 1 (1987), 1–17.

[48] Silva, K. S.; Lopes, A. d. S.; Silva, R. C. R. d.; Costa, F. F.; Assis, M. A. A. d.; Nahas, M. V. Time spent by Brazilian students in different modes of transport going to school: changes over a decade (2001-2011). *Cadernos de Saúde Pública 30* (11 2014), 2471 – 2476.

[49] Silva, M. M.; Subramanian, A.; Ochi, L. S. An iterated local search heuristic for the split delivery vehicle routing problem. *Computers & Operations Research 53* (jan 2015), 234–249.

[50] Soares, H. D.; Coelho, I.; Santos, E.; Viterbo, J.; Fernandes, H. Visualizaçâo de dados de roteamento para cidades inteligentes (in portuguese). In *I Workshop on Computational Intelligence and Smart Cities* (2017), pp. 48–57.

[51] Souza, M.; Coelho, I.; Ribas, S.; Santos, H.; Merschmann, L. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research 207*, 2 (dec 2010), 1041–1051.

[52] Souza Lima, F. M.; Pereira, D. S.; Conceição, S. V.; Ramos Nunes, N. T. A mixed load capacitated rural school bus routing problem with heterogeneous fleet: Algorithms for the Brazilian context. *Expert Systems with Applications 56* (sep 2016), 320–334.

[53] Vansteenwegen, P.; Mateo, M. An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. *European Journal of Operational Research 237*, 3 (sep 2014), 802–813.

[54] Yan, J.; Liu, J.; Tseng, F.-M. An evaluation system based on the self-organizing system framework of smart cities: A case study of smart transportation systems in China. *Technological Forecasting and Social Change*, 1 (jul 2018), 1–12.

[55] Ãlvarez Miranda, E.; Ljubic, I.; Toth, P. Exact approaches for solving robust prize-collecting Steiner tree problems. *European Journal of Operational Research 229*, 3 (sep 2013), 599–612.