Universidade Estadual de Campinas
Instituto de Computação

Julia Ramos Beltrão

The Single Line Planning Problem: Optimization Approaches to Public Transit Planning

Problema de Planejamento de Linha Única: Abordagens de Otimização para o Planejamento de Transporte Público

CAMPINAS
2022

Julia Ramos Beltrão

The Single Line Planning Problem: Optimization Approaches to Public Transit Planning

Problema de Planejamento de Linha Única: Abordagens de Otimização para o Planejamento de Transporte Público

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestra em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Fábio Luiz Usberti

Este exemplar corresponde à versão final da Dissertação defendida por Julia Ramos Beltrão e orientada pelo Prof. Dr. Fábio Luiz Usberti.

CAMPINAS
2022

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Beltrão, Julia Ramos, 1991-
B419s The single line planning problem : optimization approaches to public transit planning / Julia Ramos Beltrão. – Campinas, SP : [s.n.], 2022.

Orientador: Fábio Luiz Usberti.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Otimização combinatória. 2. Programação linear inteira. 3. Algoritmos genéticos. 4. Engenharia de transportes. I. Usberti, Fábio Luiz, 1982-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações Complementares

Título em outro idioma: Problema de planejamento de linha única : abordagens de otimização para o planejmento de transporte público
Palavras-chave em inglês:
Combinatorial optimization
Integer linear programming
Genetic algorithms
Transportation engineering
Área de concentração: Ciência da Computação
Titulação: Mestra em Ciência da Computação
Banca examinadora:
Fábio Luiz Usberti [Orientador]
Laura Silva de Assis
José Frederico Vizcaino González
Data de defesa: 19-12-2022
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-6486-812X
- Currículo Lattes do autor: http://lattes.cnpq.br/4726158152982763

Universidade Estadual de Campinas
Instituto de Computação

Julia Ramos Beltrão

The Single Line Planning Problem: Optimization Approaches to Public Transit Planning

Problema de Planejamento de Linha Única: Abordagens de Otimização para o Planejamento de Transporte Público

Banca Examinadora:

- Prof. Dr. Fábio Luiz Usberti
  Universidade Estadual de Campinas (UNICAMP)
- Profa. Dra. Laura Silva de Assis
  Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
- Prof. Dr. José Federico Vizcaino González
  Universidade Estadual Paulista Júlio de Mesquita Filho (UNESP)

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 19 de dezembro de 2022

Acknowledgments

First, I would like to thank my advisor, Fábio, who always made himself present, was always patient and understanding, encouraged me to submit articles to and attend conferences, and always believed in me.

I would also like to thank my husband, Pedro, who was by my side supporting me during the entirety of this Masters program; my parents, Renata and Minoru, who were by my side since the very beginning of my life, and whose lifetime support and encouragement made it possible for me to get into and through this program; my dear sister, Adriana, lifetime friend and companion, and a source of inspiration for her dedication to her career; and, Martini, my everyday companion, always eager to take a nap under my desk to keep me company while I worked on this dissertation.

I also thank the professors of the Institute of Computing, in special Professor Christiane, whom I had the pleasure to work with during my Undergrad Final Project, who helped me immensely, both academically and personally, and who introduced me to my advisor; the student government body of the Institute of Computing, CACo, which provided me with valuable knowledge and long-term friendships; the staff of both the undergrad and grad administrative offices of the Institute of Computing, always very supportive and efficient, in special, Daniel, Flávio and Wilson; the staff of IT services of the Institute of Computing, in special, the very friendly André and William; the Institute of Computing as a whole; the University's swimming team, USSR, where I was able to meet people from different schools and with diverse backgrounds; and the University of Campinas as a whole.

Finally, I would like to thank the Brazilian National Council for Scientific and Technological Development. This study was financed in part by the Brazilian National Council for Scientific and Technological Development (CNPq), grant 134626/2018-4.

Resumo

Em um mundo altamente urbanizado, o transporte público parece ter um papel mais importante do que nunca, uma vez que contribui para cidades mais sustentáveis e acessíveis. Com o objetivo de contribuir com o planejamento de sistemas de transporte mais eficientes, propomos um modelo de programação linear inteira pra projetar linhas de transporte público com rotas que apresentem atendimento ótimo de demanda de usuários, dado um limite de orçamento para construir a infraestrutura necessária. Diversas instâncias foram geradas, e um *solver* comercial reconhecido e um algoritmo genético, o BRKGA-MP-IPR (uma adaptação da metaheurística Biased Random-key Genetic Algorithm), foram utilizados para buscar soluções para o problema em estudo. Os resultados para ambas as abordagens são apresentados, analizados e comparados neste trabalho. Foi possível concluir que, enquanto o modelo exato obteve soluções ótimas para instâncias menores, ele não se demonstrou competitivo para instâncias maiores. Em contrapartida, o BRKGA conseguiu obter boas soluções para as instâncias maiores, tornando-o uma alternativa interessante para tratar cenários reais.

Abstract

In an exceptionally urbanized world, public transport seems to be more important than ever as it contributes for more sustainable, accessible cities. Aiming to contribute to the development of efficient transportation systems, we propose an integer linear programming model to design a demand-optimal route for a line of public transit, regarding a budget constraint for building necessary infrastructure. We generated several instances, and used a renowned commercial solver and a genetic algorithm, BRKGA-MP-IPR (a variation of the Biased Random-key Genetic Algorithm metaheuristic), to search for solutions for the studied problem. We present the results for both approaches, analyze and compare them. We observed that, while the exact model obtained optimal solutions for smaller instances, it was not competitive for larger instances. On the other hand, BRKGA attained good solutions for larger instances, proving to be a promising alternative to address real scenarios.

List of Acronyms

BRKGA: Biased Random-key Genetic Algorithm

BRKGA-MP-IPR: Multi Parent Biased Random-key Genetic Algorithm with Implicit Path Relinking

ILP: Integer Linear Program/Programming

LB: Lower Bound

OD: Origin-destination

SLPDP: Single Line Planning Decision Problem

SLPP: Single Line Planning Problem

UB: Upper Bound

Contents

Chapter 1

Introduction

Transportation has played an important role in human agglomerations throughout history. From being a determining factor in city locations - cities were usually formed on the seaside or by navigable rivers, where harbors could be built - to a limiting factor in city sizes - ineffective intraurban transport made difficult the distribution of food and other basic supplies.

But it was not until the 17^{th} century that the first organized forms of public transportation systems emerged. Vehicles which could be hired for intraurban trips, known as coaches - the predecessors of modern taxis -, first appeared in London in 1600. In 1634, coach owners obtained permission to ply the streets for hire; by 1694 there were around 700 licensed coaches in London. In 1612 public-hire sedan chairs - a chair mounted on wooden poles carried by a pair of "chair-men" - first appeared in Paris; in 1634 they were introduced in London. In 1662, a public coach service started operating on fixed routes in Paris. Vehicles could carry eight passengers at a time and were horse-drawn. However, those forms of transport were accessible only to those of the upper classes as they were very costly.

Transport systems with higher capacity began to emerge at the end of the 18^{th} century: horse-drawn omnibuses were operating in the outskirts of London as early as 1798. But it was in France the omnibus became more popular and where they were first used in the inner-city areas. Shortly after, omnibuses services were created in the United States and other European cities. Subsequently, horse-drawn tramway - which was basically an omnibus on rails - services commenced in the US and later on in Europe. The low rolling resistance of the vehicle allowed for more efficient use of horsepower and a higher passenger capacity.

The first mechanized transport services appeared in the 19^{th} century as early as 1833. However, most of those fail to attain commercial success and public approval due to factors such as unreliability and being slow and uncomfortable. It was with the emergence of electric tramways that transport services with mechanized vehicles became successful. A regular electric tramway service initiated its operation in Cleveland in 1884 and several cities in the US had this service in the middle 1880s. In Europe, electric tramway services

developed more slowly.

It was also in the 19^{th} century that occurred the development of high-speed railway transit modes. Suburban railways, with vehicles based on steam engines, had a first large-scale development in London; the first line started operating in 1838. Electric interurban railways - which consist of large, high-speed single car or short trains powered by electricity and typically connecting a group of cities that are not too far apart - were established in Northern Ireland in 1883. The rapid-transit or metro occurrence in London: the Metropolitan Line, operated with steam powered locomotives was inaugurated in 1863; 30 years later the first electric powered vehicles started operating also in London.

The 20^{th} century saw further development of electric tramway technology as well as the invention of new transport modes: the motorbus (based on an internal combustion engine), which would soon replace the horse omnibus and the trolleybus, an electric powered vehicle[1].

In Brazil, transportation was not a matter of public policy until 1808, when the Portuguese Royal Family moved to Rio de Janeiro, which was then turned into the capital of the Kingdom of Portugal, Brazil and Algarves. The first documentation concerning transport regulation was issued in 1812 by Dom João VI. However, it was limited to military functions [13].

The first public transport system, however, was not created until 1817. The conception of such system was also induced by the arrival of the Portuguese Royal Family in Brazil, when the hand-kissing ceremony was established, which subjects had to travel long distances to attend. So, in 1817, D. João VI signed a decree granting permission to an employee of the Royal Court to run a carriage transportation service between the Imperial Square, in the center of Rio de Janeiro and one of the Royal Family's dwellings, around 50 kilometers apart from each other [18].

While the remaining of the 19^{th} century saw investments in railroads increase, the 1900s took a sharp turn to road-oriented transportation. During his term as president (1926-1930), Washington Luís went as far as claim that "to govern is to build roads". Nationwide road planning did not commence, however, until 1937, when a National Road Plan was designed. Later on, in 1945, Law Joppert was passed, setting up the National Fund for Roads. During the 1950s, roads are further confirmed as the main mode of integration of Brazil's territory. Juscelino Kubitschek, president between 1956 and 1960, with his motto "50 years in 5", invested heavily in road building and provided huge incentives to the automobile industry. He also tried to rescue aquatic and rail transportation systems, but with little success [13].

Nowadays, transportation, in particular public transportation, remains as important as

[1]The first paragraphs of this text are based on the book *Urban Transit Systems and Technology* by Vuchic [23]

ever. The number of trips carried out on public transportation had an 18% increase between the years 2000 and 2015 [22] and public transport is known to be the answer to several problems large cities face in the present days, such as traffic congestion, air pollution, and energy consumption [14]. Moreover, transportation systems may affect land use and real estate prices, as is in the case of São Paulo, one of the largest cities in the world [1].

With the relevance of public transit in mind, we propose to optimize a single line of a public transportation system by maximizing the number of trips (demand) served. Note that, although we have so far provided a very practical motivation, in this work we study the problem of planning transportation systems and networks under quite a few layers of abstraction. Ultimately, we simplify this as a graph optimization problem. In order to evade ambiguity with terminology, Section 2.1 presents definitions of graph theory, Section 2.2 introduces concepts of transportation planning, and Section 2.3 cites definitions of combinatorial optimization. Finally, Section 3.1 displays the formalization of the Single Line Planning Problem (SLPP) as an Integer Linear Programming model.

Several optimization models to address transportation planning problems have been proposed in the literature. Laporte et al. [16] proposed a model based on linear programming to design networks of rapid-transit modes - which are commonly segregated and independent of the street network. The model is rather complex taking into account travel demand, how this demand is distributed between private and public transport, the decision of constructing both stations and alignments connecting stations - as well as the cost of constructing these.

Borndörfer et al. [7] proposed a column-generation approach to network design tied to the problem of finding the frequencies of each line (this combination is known as the line planning problem) in order to satisfy a given travel demand. They detected two objectives: minimizing operating costs for the company and minimizing traveling times for passengers. A multi-commodity flow model was introduced and results for the data of the city of Postdam, Germany, were reported.

Bussieck et al. [8] also investigated the line planning problem. They used both nonlinear and mixed integer optimization to produce models for the problem and considered two objectives: minimizing operating costs and passenger convenience, which is modeled as a minimum for the frequency. Results for practical data of the Dutch Railways were presented.

Marín and Jaramillo [26] presented a model for urban rapid transit network design, which comprises the location of stations and the alignments between stations. The model aims at maximizing travel demand supplied under cost constraints and it also takes into account that users can choose their transportation modes and trips. As the model cannot be solved efficiently by Branch and Bound due to a large number of variables and constraints, algorithms based on Benders decomposition were proposed. The two methods were compared using fabricated networks.

Zarrinmehr et al. [25] proposed a multi-objective bi-level optimization model for the network design problem. The model takes into consideration (maximizing) transit ridership, (minimizing) agency operating cost, system performance, and an elastic demand. The authors conceived a greedy algorithm to solve the model.

After preliminary experiments (Section 5.4), which showed the ILP formulation had low scalability, we hypothesized that the SLPP was NP-hard and were to draw a proof for such hypothesis, presented in Section 3.2 - definitions provided in Section 2.4. As a consequence, we decided to test a heuristic approach. The chosen metaheuristic was the Multi Parent Biased Random-key Genetic Algorithm with Implicit Path Relinking (BRKGA-MP-IPR) [3] - terminology is presented in Section 2.5.

A fairly new method, BRKGA-MP-IPR has few applications reported in the literature. Kummer et. al [15] used BRKGA-MP-IPR to address the home health routing and scheduling problem. Londe et. al [17] studied the Root Sequence Index (RSI) allocation problem[2] using exact and heuristic methods, the BRKGA-MP-IPR being one of them. Andrade et. al [2] adapted the BRKGA-MP-IPR framework to solve the physical cell identity assignment problem. In all cases, BRKGA-MP-IPR is used in order to improve the scalability of optimization problems.

Our contributions encompass the proposition of a generic model which can adapt to support transport modes other than interurban trains and should be easier to solve when compared to the existing ones, and the development of a heuristic method to overcome the low scalability limitation of the ILP formulation.

[2]RSI is used to allocate uplink channels between user equipment and a base station.

Chapter 2

Definitions and Terminology

2.1 Graphs

The definitions and notations presented in this section are based on Diestel's *Graph Theory* [9] and Bondy's and Murty's book, also named *Graph Theory* [6].

Definition 1: Graph

A *graph* is an ordered pair $G = (V, E)$ of disjoint sets such that $E \subseteq [V]^2$ - elements of V are called *vertices* of the graph and elements of E are called *edges* - along with an incidence function ϕ_G that associates each edge in E to an unordered (not necessarily distinct) pair of vertices in V.

The vertex set of a graph $G = (V, E)$ is denoted by $V(G)$, and its edge set by $E(G)$. We may not always make a strict distinction between a graph and its vertex and edge set. For instance, we may say that a vertex $v \in G$, or that an edge e is in G.

For the following definitions we will assume graphs are non-empty, unless stated otherwise.

Definition 2: Incidence

A vertex v is *incident* to an edge e if $v \in e$. In this case, we say e is an edge *at* v and v is an *end vertex* or *end* of e; also, we say an edge *joins* its ends.

An edge with identical ends is named a *loop*. Two or more edges with the same pair of distinct ends are called *parallel edges*.

Definition 3: Simple Graph

A graph is said to be *simple* if it has no loops or parallel edges.

Definition 4: Adjacency

Two distinct vertices u, v of G are *adjacent*, or neighbors, if $\{u, v\}$ is an edge of G. The set of neighbors of a vertex v in G is denoted by $N_G(v)$.

Drawing Graphs
Graphically, graphs are usually represented by drawing a dot for each vertex and a line for each edge between a pair of dots if there is an edge that joins the vertices that those dots represent.
The above convention will be used throughout this work.

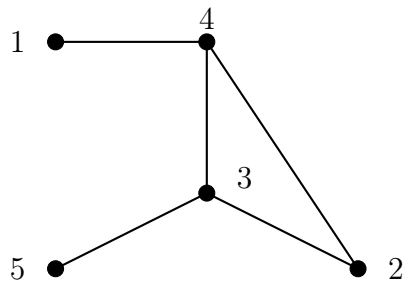

Figure 2.1: The graphic representation of the graph $G = (\{1, 2, 3, 4, 5\}, \{\{1, 4\}, \{4, 3\}, \{3, 2\}, \{2, 4\}, \{5, 3\}\})$.

Definition 5: Subgraph

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. Then, if $V' \subseteq V$ and $E' \subseteq E$, G' is said to be a *subgraph* of G (and G a supergraph of G') or $G' \subset G$. Less formally, we say that G contains G'.

If $G' \subset G$ and $G' \neq G$, then G' is called a proper *subgraph* of G.

Definition 6: Directed Graph

A *directed graph* O is a pair (V, A) of disjoint sets of vertices and arcs accompanied of an *incidence function* ψ_O, which associates each arc of A with an ordered pair of (not necessarily distinct) vertices of V. Let a be an arc in A and $\psi_O(a) = (u, v)$, then u and v are the *ends* of e and e is said to be *oriented* from u to v.

Drawing Directed Graphs

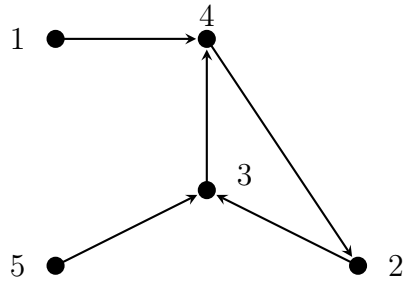

Figure 2.2: The graphic representation of the directed graph $O = (\{1,2,3,4,5\}, \{(1,4),(3,4),(2,3),(4,2),(5,3)\})$.

Definition 7: Vertex Degree

Graph
Let $G = (V, E)$ be a graph. Then the *degree* of a vertex v in G, denoted by $d_G(v)$, is equal to the number of edges at v.

Directed Graph
Let $D = (V, E)$ be a directed graph and v a vertex of D. The vertices of D that *dominate* v are called its *in-neighbors*; the vertices that are *dominated* by v are its *out-neighbors*. A vertex u of D is called an *in-neighbor* of v if there is an edge joining u to v; symmetrically, u is called an *out-neighbor* of v if there is an edge joining v to u. These sets of vertices are denoted by $N_D^-(v)$ and $N_D^+(v)$, respectively. We define the *in-degree* of a vertex v as $d_D^-(v) = |N_D^-(v)|$ and the *out-degree* as $d_D^+(v) = |N_D^+(v)|$.

Definition 8: Weighted Graph

A (directed) *weighted graph* W is a (directed) graph accompanied of a *cost function* $w(e) : E \to \mathbb{R}$ for the (arcs) edges, which associates a cost to each (arc) edge of W.

Drawing Weighted Graphs

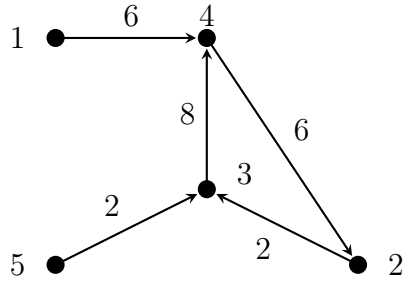

Figure 2.3: The graphic representation of the weighted graph $W = (\{1,2,3,4,5\}, \{(1,4),(3,4),(2,3),(4,2),(5,3)\})$.

Definition 9: Path

A *path* is a (directed/weighted) graph $P = (V, E)$ of the form

$$V = \{x_0, x_1, \ldots, x_k\}, E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$$

where the $x_i, i = 0, 1, \ldots, k-1, k$ are all mutually distinct.
The *length* of a path is defined as

- the number of (arcs) edges in it, for graphs and directed graphs;
- the sum of the weights of the edges in it, for weighted graphs.

We shall hereafter refer to paths by the natural sequence of their vertices. For instance, we shall write $P = (x_0, \ldots, x_k)$, and call P a path *between* x_0 and x_k.

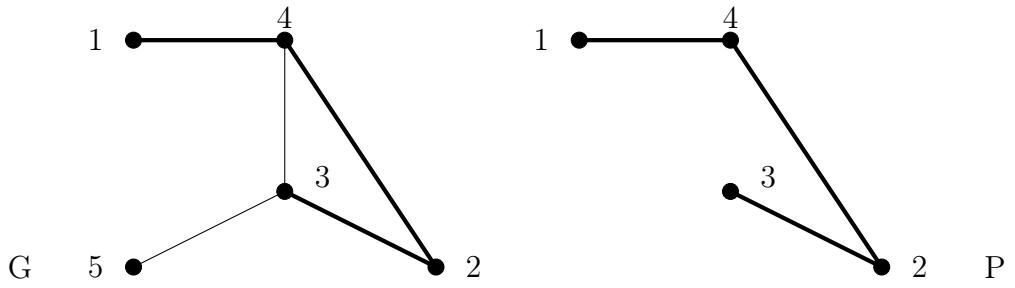

Figure 2.4: $P = (1, 4, 2, 3)$ is a path of length 3 between vertices 1 and 3 in the graph G.

Definition 10: Shortest Path

A shortest path (or geodesic) between two vertices u and v is a path between u and v of minimum length.

Note there could be more than one shortest path between a pair of vertices.

Definition 11: Hamiltonian Path

A *hamiltonian path* of a given graph G is a path that visits all vertices of G.

Definition 12: Distance

Let G be a (directed/weighted) graph and u and v be vertices of G. Then the *distance* between u and v is the length of a shortest path between u and v and it is denoted by $dist(u, v)$.

Definition 13: Connectivity

A (directed/weighted) graph G is said to be connected if for any two vertices u and v of G there is a path between u and v.

Definition 14: Cartesian Product

Let $G = (V_G, E_G)$, $V_G = \{u_1, ..., u_m\}$, and $H = (V_H, E_H)$, $V_H = \{v_1, ..., v_n\}$, be simple graphs.
The *cartesian product* of G and H is the graph $G \times H = (V, E)$, $V = V_G \times V_H$ and $E = \{(u_i v_j, u_k v_l) : (u_i, u_k) \in E_G \text{ and } v_j = v_l \text{ or } (v_j, v_l) \in E_H \text{ and } u_i = u_k\}$.

An example of a cartesian product is shown in Figure 2.5.

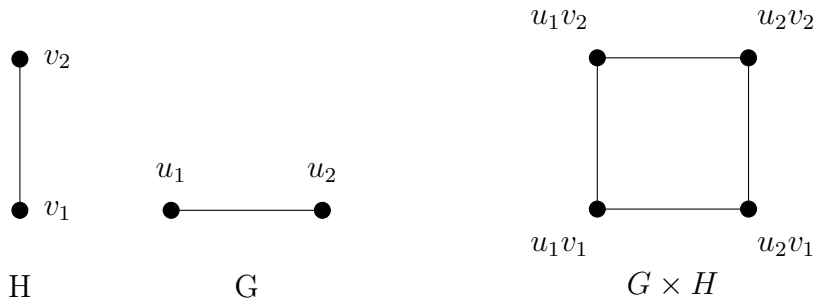

Figure 2.5: Graphs G and H and their cartesian product $G \times H$.

Definition 15: Two-Dimensional Grid Graph

Let P_m and P_n be path graphs with m and n vertices, respectively.
A *two-dimensional grid graph* is the cartesian product $P_m \times P_n$ [24].

We say $P_m \times P_n$ is a grid graph of dimension $m \times n$.

The two following definitions will not likely be found in graph theory literature and have been added here to help describing instances in Section 5.2.

Definition 16: Center and Border Vertices

Let G be a grid graph with dimension $m \times n$, $n, m \geqslant 3$. Then all vertices $v \in G$ such that $d_G(v) = 4$ are called *center vertices*, and all vertices $v \in G$ such that $d_G(v) < 4$ are called *border vertices*.

Definition 17: Density

Let $G = (V, E)$ be a grid of dimension $m \times n$ and $G' = (V', E')$ a subgraph of G such that $V' = V$ and $E' \subseteq E$. The *density* of G' is defined as the ratio $r = \frac{|E'|}{|E|}$.

Sometimes, we may refer to the subgraph of a grid with density r as grid with density r.

2.2 Transport-Related Concepts

Origin-Destination Matrix

An *origin-destination matrix*, or *OD matrix*, summarizes information about trips a given population makes on a daily bases. Usually, these matrices represent a given period of time, for instance, the one hour period between 13:00 and 14:00; weekdays and weekends are usually represented separately[1]. Let $D = d_{ij}$ be an $n \times n$ origin-destination matrix for a given period, then $d[i][j]$ is the travel demand - the number of daily trips - from location i to location j, and $d[j][i]$ is the travel demand from location j to location i, $i, j \in \mathbb{Z}_+, i, j < n$. Note that $d[i][j]$ is not necessarily equal to $d[j][i]$; $d[i][j] = 0$ for $i = j$.

Line of a Transportation System

A *line of a transportation system* is a fixed itinerary between two fixed locations through which vehicles of the given system drive with a predefined frequency. In this text we shall refer to a line of a transportation system simply as line.

Street Network

The *street network* of a given area is the network formed by the streets encompassed in this area. Also referred as road network.

Street Segment

A *street segment* is a section of a given street. In this text, we use street segment to refer to a section of street between any two consecutive intersections of the street. Also referred as road segment.

[1] In this work, each matrix is considered a different instance of the problem.

2.3 Combinatorial Optimization and Linear Programming

Definition 18: Optimization Problem

Let X be a set and $f : X \longrightarrow \mathbb{R}$ be a function.
Then, finding $x^* \in X$ such that

$$f(x^*) \leqslant f(x) \forall x \in X$$

is an (minimization) optimization problem.
Generally, X is called the search space, each element of X is called a feasible solution, and the point x^* is called an optimal solution.

Optimization problems can be divided into two categories: problems with continuous variables and problems with discrete variables; the latter are also known as combinatorial optimization problems.
Linear programming is an optimization problem in which X is a polyhedron formed by the intersection of semispaces and objective function f is a linear function of the decision variables. More formally, we may define linear programming as follows.

Definition 19: Linear Programming

Let f and g_i, $i = 1, ..., m$ be linear functions.
A linear programming problem is finding $x \in \mathbb{R}^n_+$ to

$$\text{minimize } f(x)$$

subject to

$$g_i(x) \geqslant 0, i = 0, ..., m$$

Definition 20: Linear Integer Programming

An *integer linear programming problem* is a linear programming problem with the additional restriction that all variables must take integer values.

Definition 21: Binary Integer Programming

A *binary integer programming problem* is a linear programming problem such that all variables are restricted to take the values 0 or 1.

2.4 NP-completeness

This section was based on Garey and Johnson [10].

Definition 22: Decision Problem

A decision problem ρ consists of a set of instances I_ρ, a *yes-no* question P_ρ, asked in terms of a generic instance, and a subset $Y_\rho \subseteq I_\rho$ of *yes-instances*, that is, instances which the answer to P_ρ is *yes*.

Definition 23: Class P

The class P is defined as the set of all decision problems ρ for which there is a deterministic algorithm α with **polynomial running time** that solves ρ.

Definition 24: Class NP

The class NP is defined as the set of all decision problems ρ for which there is a non-deterministic algorithm β with **polynomial running time** that solves ρ.

Note that $P \subseteq NP$, and whether $P = NP$ remains an open question.

Definition 25: Polynomial Transformation

A polynomial transformation of a decision problem ρ_1 to a decision problem ρ_2 is an algorithm T that runs in polynomial time and can transform every instance $i \in I_{\rho_1}$ in an instance $j \in I_{\rho_2}$.

Definition 26: Class NP-complete

The class NP-complete is defined as the set of all decision problems $\rho \in NP$ such that problem $\rho' \in NP$ can be reduced in polynomial time to ρ.

Definition 27: NP-hard Problem

An NP-hard problem ρ is such that there exist a polynomial transformation from a problem $\rho' \in$ NP-complete to ρ.

2.5 Metaheuristics

A *metaheuristic* is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithm [21]. Generic

algorithms, local search and path-relinking are examples of metaheuristics.

Genetic Algorithms

Genetic algorithms apply the concept of survival of the fittest to search for optimal or near-optimal solutions to combinatorial problems. Each individual of a group, or population of individual, is mapped to a solution of the optimization problem being studied. Each individual is represented by a chromosome, which consists of an array of genes. Each gene has an associated value, called an allele. Each chromosome has a fitness value which corresponds to the objective function value of the solution it is mapped to.
Given a population of individuals, this metaheuristic performs a set of operations on it - or evolves it - for a number of iterations, called generations. At each generation a new population is created by combining individuals of the current population. These individuals are chosen at random but those with higher fittest are preferred. Random mutation also takes place in order to avoid being trapped in local minima.

Random-key Genetic Algorithm

In *Random-key Genetic Algorithms* (*RKGA*), alleles taken on values from the real interval $[0, 1]$, generated independently at random. The mapping of a chromosome to a solution is made by a deterministic algorithm, called decoder, which is specific to the problem being studied.
RKGA starts with an initial population of p randomly generated individuals (chromosomes). The metaheuristic then proceeds to repeat some steps for g generations - note that the size of the population remains the same during the evolution process. In a given generation ζ, the chromosomes are mapped into solutions, those being used to separate the population into two different groups: elite and non-elite. The elite group holds the p_e, $p_e < p - p_e$, fittest individuals, that is, the chromosomes that map to the p_e best solutions - which are passed on to generation $\zeta + 1$. The population of generation $\zeta + 1$ is completed by adding p_m mutants (randomly generated individuals) and $p - p_e - p_m$ individuals are generated through crossover, which produces a new individual by sampling genes from two individuals of generation ζ, called parents; those are drawn randomly from the entire population.

Biased Random-key Genetic Algorithm[2]

Proposed by Gonçalves and Resende [12], the *Biased Random-Key Genetic Algorithm* (*BRKGA*) differs from RKGA in the way parents ara chosen. In BRKGA, one parent is drawn randomly from the elite group, and the other from the non-elite group, also at random. During the crossover, an allele for the resulting chromosome is chosen from the elite parent with probability φ_e.
An evolution cycle is shown in Figure 2.6[12]. The terms TOP and BOT may be ignored for the purpose of this text.
In BRKGA, there is a clear division between problem-dependent and problem-indepent parts. The problem-dependent part comprises only the decoder, and, therefore, in order to

[2]The Figures used in this Section appeared originally in *Biased random-key genetic algorithms for combinatorial optimization* by Gonçalves and Resende [12].

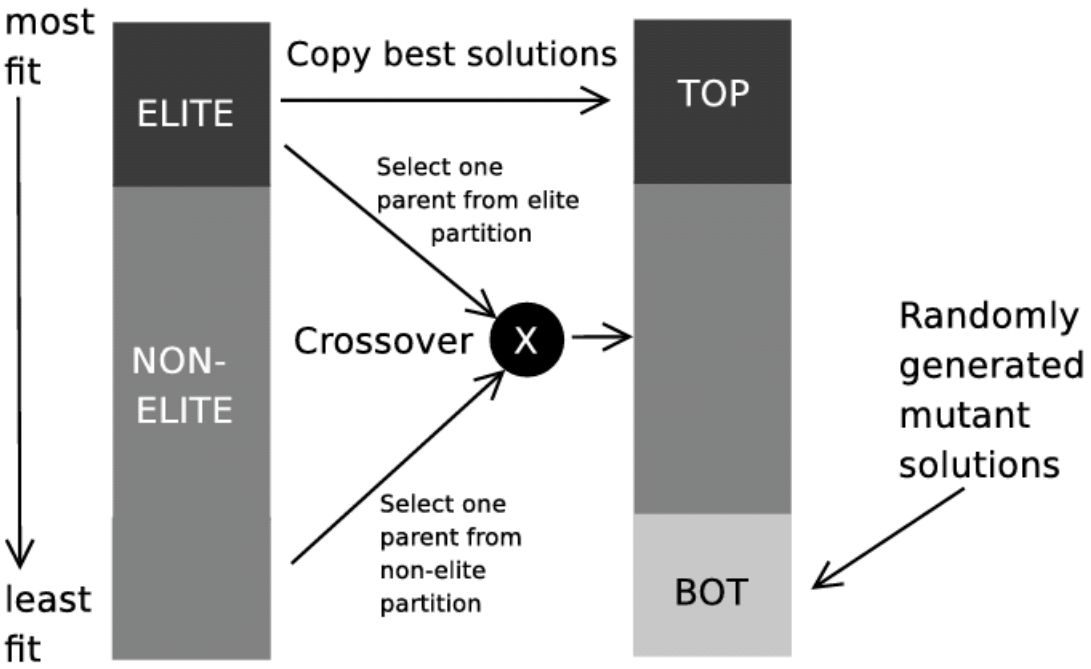

Figure 2.6: Transition from generation ζ to $\zeta + 1$ in BRKGA.

define a BRKGA heuristic, one needs only to specify the chromosome representation and the decoder. Everything else is problem-independent. Figure 2.7[12] presents a flowchart for BRKGA and emphasize this separation.

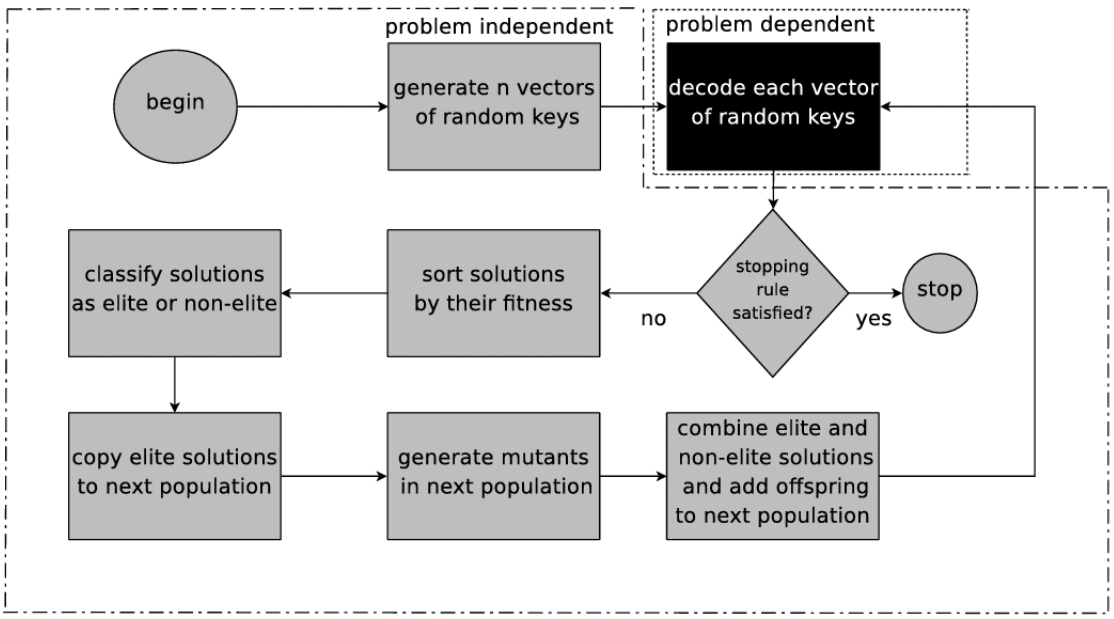

Figure 2.7: Flowchart for BRKGA.

BRKGA With Multiple Parents

Proposed by Lucena et al. [19], *Multi-Parent Biased Random-Key Genetic Algorithm* or *BRKGA-MP* is a variation of the BRKGA metaheuristic in which $n, n > 2$, parents are used to generate a new individual. The number of parents n is set prior to running the heuristic and remains fixed, as well as the number of parents to be drawn from the elite group.

Path-Relinking

The idea behind the Path-Relinking approach is to explore the neighborhood of the path between two feasible solutions. Two feasible solutions - a base and a guide solution - are sampled from the pool of known feasible solutions. These solutions need to have a sufficient amount of different components between them; such amount is fixed beforehand. A component of the base solution is then selected and replaced with a component from the guide solution. This process is repeated until all components of the base solution have been substituted by components from the guide solution. All the intermediate solutions are evaluated and the best one is returned [11].
The drawback of this approach is that it is strongly dependent of the specific problem at hand. The use of Path-Relinkig combined with BRKGA was first proposed by Ribeiro et al. [20].

Implicit Path-Relinking

Andrade et al.[3] proposed a variation to the Path-Relinking strategy, which they named *Implicit Path-Relinking*. This variation takes advantage of the already implemented chromosome decoder. Two different versions of Implicit Path-Relinking were proposed, *Direct Implicit Path-Relinking* and *Permutation-Based Implicit Path-Relinking*. For this work, we are interested in the latter.

Permutation-Based Implicit Path-Relinking

Crafted to address problems where the order of the genes in the chromosome is used by the decoder to create a solution. This version of Implicit Path-Relinking, instead of exchanging keys between the base and guide solutions, uses the permutation from the guide solution to switch the keys of the base solution in such a way that they induce the same "sub-permutation" on the base solution [3].

Local Search

Local search is a tool to improve results by searching the neighborhood of the solutions found by a heuristic. A formal definition follows next.

Definition 28: Local Search

Given an instance (F, c) of an optimization problem, where F is the feasible set and c is the cost mapping, we choose neighborhood

$$N : F \to 2^F$$

which is searched at point $t \in F$ for improvements by the subroutine

$$\text{improve}(t) = \begin{cases} \text{any } s \in N(t) \text{ with } c(s) \leqslant c(t) \text{ if such an } s \text{ exists;} \\ \text{“no” otherwise.} \end{cases}$$

Chapter 3

Formulation and Complexity

3.1 The Single Line Planning Problem

The Single Line Planning Problem (SLPP) is defined as follows.

Let $G = (V, E)$ be the weighted graph representing the road network of a given urban area. Each edge represents a road segment, each vertex represents a road intersection, and each edge weight c_{ij} denotes the cost to realize the system infrastructure on that road segment. Let D be the OD matrix, a square matrix of dimension $|V|^2$ where d_{ij} is the travel demand from vertex i to vertex j, $i, j \in V$. Note that for $i = j$ $d_{ij} = 0$.

We construct a directed weighted graph $O = (V', A)$ such that $V' = V \cup \{s, t\}$, where s and t are artificial source and target locations, respectively, and $A = \{(u, v), (v, u) : \forall\{u, v\} \in E\} \cup \{(s, i), (i, t) : \forall i \in V\}$, with arc eights $c'_{uv} = c'_{vu} = c_{uv}, \forall\{u, v\} \in E$ and $c'_{si} = c'it = 0, \forall i \in V$. An example of this construction is presented in Figure 3.1.

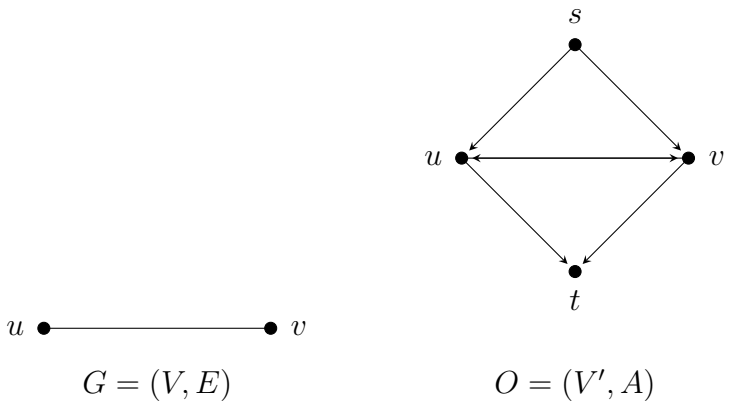

Figure 3.1: Example of graph and the derived directed graph (costs were omitted).

We formulate a binary integer program that receives an directed weighted graph O, OD matrix D, and budget constraint b as inputs. A solution can be represented by a subgraph $O_L = (V_L, A_L)$, $O_L \subseteq O$, which serves demand d_{ij}, if $y_{ij} = 1$, or not, if $y_{ij} = 0$. The

adjacency matrix of O_L is given by $X = (x_{ij}), x_{ij} \in \{0,1\}, i,j \in V$.

The decision variables are formally presented next.

$$x_{ij} = \begin{cases} 1, & \text{if } (i,j) \in O_L. \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{ij} = \begin{cases} 1, & \text{if demand } d_{ij} \text{ is served.} \\ 0, & \text{otherwise.} \end{cases}$$

The ILP formulation for the SLPP is formally presented next.

$$\text{Max} \sum_{i,j \in V} y_{ij} d_{ij}$$

Subject to

$$\sum_{i,j \in V} x_{ij} c_{ij} \leqslant b \quad \text{(budget constraint)}$$

In order to guarantee O_L is connected, we must enforce that for every arc that reaches a given vertex i, $i \notin \{s,t\}$, there is an arc that leaves i, that is, the in-degree of i equals its out-degree. For s and t the out-degree and the in-degree, respectively, must be equal to 1. Hence, we add the following constraints.

$$\sum_{k \in V} x_{ki} = \sum_{l \in V} x_{il} \quad \forall i \in V$$

$$\sum_{i \in V} x_{si} = 1$$

$$\sum_{i \in V} x_{it} = 1$$

A demand d_{ij} is served ($y_{ij} = 1$) if and only if both i and j are in the path formed by O_L, that is, there is an arc $a \in O_L$ that leaves vertex $i \in O_L$ and an arc $a' \in O_L$ that reaches vertex $j \in O_L$. Thus, we add the following constraints.

$$y_{ij} \leqslant \sum_{k \in V} x_{ik} \quad \forall i,j \in V$$

$$y_{ij} \leqslant \sum_{k \in V} x_{kj} \quad \forall i, j \in V$$

In order to ensure that the path described by O_L does not go through the same vertex more than once, that is, there are no subcycles in G_L, we must add constraints to remove subcycles.

$$\sum_{i,j \in S} x_{ij} \leqslant |S| - 1 \quad \forall S \subseteq V$$

Note that these constraints also guarantee that O_L has a single connected component.

3.2 Complexity Analysis

In this section, we prove that the SLPP is an NP-hard problem, that is, it belongs in a class of problems to which no algorithm that solves them runs in polynomial time, unless P = NP. We show a polynomial reduction of a known NP-hard problem to the SLPP. Because polynomial reductions are usually done on decision problems rather than optimization problems, we compose a decision version of the SLPP.

This analysis is relevant because it supplies us with a sound justification for the metaheuristic approach also used to address the SLPP as defined in Section 3.1. In the remaining of this section, we present formal definitions for the concepts mentioned in the last paragraph, as well as the proof that the SLPP is NP-hard.

Lemma 1

If there is a polynomial transformation T which transforms a decision problem ρ_1 in a decision problem ρ_2, then $\rho_1 \in$ P implies $\rho_2 \in$ P [10].

Definition 29: Hamiltonian Path Problem (HPP)

Given a graph G, determine if a hamiltonian path exists in G.

Lemma 2

The HPP is NP-complete [10].

Definition 30: Single Line Planning Decision Problem (SLPDP)

Given a graph G, an OD matrix D, a demand value δ and budget value b, determine if a path in G, with cost c, $c \leqslant b$, that serves at least demand d exists.

Theorem 1

The SLPDP is NP-hard.

Proof

An instance of the HPP can be reduced to an instance of the SLPDP using a polynomial running time algorithm with the following steps.

- Convert $G = (V, E)$ in $O = (V', A)$, such that $V' = V \cup \{s, t\}$ and $A = \{(u, v), (v, u) : \forall\{u, v\} \in E\} \cup \{(s, i), (i, t) : \forall i \in V\}$, as described in Section 3.1;
- Set cost 0 to all arcs;
- Set $b = 0$;
- Build OD matrix D such that $d_{ij} = 1, i \neq j$, $d_{ij} = 0, i = j$;
- Set
$$\delta = \sum_{i,j \in V} d_{ij}$$
.

In order to serve demand d, the solution path has to visit all vertices of O, so that demand d_{ij} is served for all vertex pairs $0 \leqslant i, j < |V|$. But a path that visits all vertices of a graph is a hamiltonian path. Hence, we can use an algorithm that solves the SLPDP to solve the HPP - if the answer to the SLPDP is *yes*, then the answer to the corresponding HPP is also *yes*.
Furthermore, we may also affirm that if there is an algorithm that solves the SLPDP in polynomial time, then the HPP can also be solved in polynomial time.

Chapter 4

Metaheuristic Approach

As explained in Section 2.5, the BRKGA heuristic requires the user to define a decoder to map chromosomes into solutions. In the following Sections, we introduce three different decoders created and used for this work.

Note that these decoders receive as inputs an OD matrix, a budget limit and an undirected weighted graph.

4.1 Vertex-based Decoders

Let a chromosome be a sequence of genes $g_1, \ldots, g_n$, with alleles $k_1, \ldots, k_n$, where n is the number of nodes in the input graph G. For these decoders, we map each gene g_i of the chromosome to vertex v_i of G. We use this mapping to build a feasible solution for the SLPP. In the following subsections, we describe two different methods to build such a solution.

4.1.1 Decoder Without Local Search (Decoder 1)

We start by adding to the path the vertex mapped to the gene with the lowest allele. Let us call this vertex v. We continue building the path by adding to it the node among neighbors of v mapped to the lowest allele. Let us call this vertex u. We keep on growing the path by adding the vertex that has been mapped to the lowest allele among those in the union of neighbors of v and u minus the vertices that have already been added to the path. Let us call this vertex w. Note that if the path we have so far is (u, v) and w is a neighbor of v, the path resulting of the addition of w is the path (u, v, w); otherwise, the resulting path is (w, u, v). If w is neighbor to both v and u, we add w to the beginning of the path by default - as we did when adding u to the path. There is no particular reason for this default behavior.

More generally, given a path $p = (v_1, \ldots, v_n)$, we search among the neighbors of v_1 and v_n for the one associated to the lowest allele, let us call it z; z is added to p only if the cost of the resulting path does not surpasses the budget limit B, and we do so according to the following rules:

1. if z is neighbor of v_1, we add it at the begging of the path, regardless of whether z is also a neighbor of v_n or not;

2. if z is only neighbor of v_n, we add z at the end of the path.

Once we can no longer increase the path due to the budget limit, we use the OD matrix D to calculate the demand served.

4.1.2 Decoder With Iterative Local Search (Decoder 2)

This decoder operates in a similar fashion as the previous one. The difference is that, after adding a new vertex z at the beginning or the end of the existing path, the decoder checks whether more vertices can be added to the result without increasing its cost.

Generally, given a path $p = (v_1, \ldots, v_n)$ to which z was just added ($v_1 = z$ or $v_n = z$), we check, for each pair v_i, v_{i+1}, $i = 1, \ldots, n-1$, whether there is a path $p_{v_i \to v_{i+1}}$ between v_i and v_{i+1} which costs at most as much as the edge (v_i, v_{i+1}) and $p - \{v_i, v_{i+1}\} \cap p_{v_i \to v_{i+1}} = \{\varnothing\}$. If such path exists, we replace vertices v_i and v_{i+1} by $p_{v_i \to v_{i+1}}$. Note that this replacement guarantees there is no raise in cost of the resulting path and no decrease in demand, while having the potential of increasing the demand served.

4.2 Edge-based Decoder (Decoder 3)

The vertex-based decoders described above introduce a "locality" bias. The solution path is limited to the vertices that are in the same component as the first vertex added to the path. In an attempt to avoid this bias, we propose an edge-based decoder that sees the chromosome as a "*text*" to be *directly translated* into another language - a candidate solution -, rather than using it as a *guide* to *build* a viable solution. Such decoder is described next.

Let a chromosome be a sequence of genes $g_1, \ldots, g_m$, with alleles $k_1, \ldots, k_n$, where m is the number of edges in the input graph G. For this decoder, we map each gene g_i of the chromosome to edge e_i of G. We use this mapping to decode a chromosome into a candidate solution for the SLPP.

The decoding is done according to the following criteria. Each allele k_i is interpreted as an indicative of the presence or absence of edge e_i in the candidate solution. As alleles taken on values in the real range $[0, 1]$, the decoder associates values in the range $[0, 0.5)$ as the absence of the edge, and values in the range $[0.5, 1]$ as the presence of the edge in the solution. Once the candidate solution is transcribed, the decoder verifies whether it is a viable solution. In order to do so, the decoder must guarantee the candidate solution holds the following.

- Its total cost is smaller than or equal to the given budget limit;

- the solution has no cycles;
- the solution edges form a single path.

If the candidate solution proves to be feasible, the OD matrix is used to calculate the demand served. Otherwise, the solution is discarded.

Chapter 5

Computational Experiments

5.1 Computational Environment

All experiments were conducted on a headless machine with two Intel Xeon CPU E5-2630 v4 2.20GHz processors, with 10 cores each, running Ubuntu 18.04.3; the algorithms were developed in Python 3.7. Gurobi v9.0 - through its Python interface - was used as an integer linear programming solver, and a Python module named *brkga-mp-ipr* [3] was used to implement the BRKGA-MP-IPR based heuristic solution.

5.2 Instances

An instance for the SLPP is comprised of an weighted graph, an OD matrix and a budget limit. In this section we explain the methods used to generate each of those elements for the experiments described in this work.

5.2.1 Graphs

In order to emulate reticulated street networks, the graphs used are two-dimensional grids, with dimensions $n \times n$, $n = 4, 6, 8, 10, 20, 40, 60, 80, 100$. In an attempt to mimic real cities, which tend to have blocks of different sizes, we also use subgraphs of those grids with densities $r \approx 0.2, 0.4, 0.6, 0.8$.

Edge Weights

The costs of the edges were generated based on an uniform distribution and were drawn as follows. Let C_1 be a circle of radius 0.2 and center $(0, 0)$ in the Cartesian plane. Let C_2 be a circle with radius 0.2 and center $(0.6, 0)$. We draw a point $p_1 = (x_1, y_1)$ from C_1 and a point $p_2 = (x_2, y_2)$ from C_2 by sampling x_1, y_1 and y_2 from the uniform distribution $[-0.2, 0.2)$ and x_2 from the uniform distribution $[0.4, 0.8)$. We then calculate the Euclidean distance *dist* between p_1 and p_2, multiply *dist* by 1000, truncate the

result and attribute this value to the cost c_e of edge $e = (u, v)$, where v and u are vertices of a given grid. This process is repeated for each edge of the given grid. Note that if an edge e is in both a grid and the derived subgraph, then its cost c_e is the same in both.

Note that these weights presume the cost of a street segment is independent of its position and the cost of others segments.

5.2.2 OD Matrices

Two types of OD matrix were generated using two different distributions, the uniform distribution, which assumes individual demands do not depend on each other, and the clusterized distribution, which presumes the existence of interest points that concentrate demand between each other. For instance, these points could be a small square in a residential suburb and a bus station in the center of an area with a concentration of cultural facilities (museums, theaters, cinemas, etc). Three "flavors" of the clusterized distribution were designed.

- Interest points are sampled randomly from all the vertices of the grid,
- interest points are sampled randomly from the border vertices of the grid, and
- for each pair of interest points, one point is sampled randomly from the border vertices and the other from the central vertices of the grid.

The strategies to build these matrices is described below.

Uniform Distribution

Each element $m_{ij}, i \neq j$ was drawn uniformly from the discrete interval $[1, 100]$.

Clusterized Distribution

For each grid graph $G = (V, E)$, we choose $n_S = 0.1 \times |V|$ vertex pairs (u, v), $u \neq v$. We name each of these pairs as *seed*, and refer to the set of seeds as S. We only calculate clusterized distributed OD matrices for $n > 20$, so n_S is always a positive integer.

Seeds may be chosen according to three different strategies:

- both u and v are sampled randomly from all vertices of G,
- both u and v are sampled randomly from all border vertices of G,
- u is sampled randomly from border vertices of G and v is sampled randomly from all center vertices of G.

Once all n_S seeds have been chosen, we determine values d_{ij} of OD matrix D as follows. Let D_{min}, D_{max} and α be numbers such that $D_{min}, D_{max} \in \mathbb{N}$, $D_{min} < D_{max}$, $\alpha \in \mathbb{R}$, $\alpha \in (0, 1)$; $uniform\_sampling(interval)$ be a function that returns an integer value sampled uniformly from $interval$; $dfs(v, G)$ a function that returns the set of all vertices in the depth-first[1] tree of G from vertex v; $dist(v_1, v_2)$ is the distance, in number of edges, between v_1 and v_2; and, $ceil(f)$ returns the smallest integer greater than f. A pseudocode of the construction method based on a clusterized distribution is given next.

$d_{ij} \leftarrow 0, \forall i, j = 0, \ldots, |V| - 1$
for all $(u, v) \in S$ **do**
  $demand_{uv} \leftarrow uniform\_sampling([D_{min}, D_{max}])$
  for all $w \in dfs(v)$ **do**
    $i \leftarrow dist(v, w)$
    $d_{uw} \leftarrow d_{uw} + ceil(\alpha^i \times uniform\_sampling([D_{min}, demand_{uv}]))$
  end for
  $demand_{vu} \leftarrow uniform\_sampling([D_{min}, D_{max}])$
  for all $w \in dfs(u)$ **do**
    $i \leftarrow dist(u, w)$
    $d_{vw} \leftarrow d_{vw} + ceil(\alpha^i \times uniform\_sampling([D_{min}, demand_{vu}]))$
  end for
end for
We set $D_{min} = 100$, $D_{max} = 200$ and $\alpha = 0.8$.

5.2.3 Budget Limit

The budget limits are percentages of the total cost of the studied grid or subgraph. The percentages used were 25%, 50% and 75%.

5.3 Parameters

The time limit used for all experiments described in Sections 5.4 and 5.5 was 10 minutes. Parameters specific to the heuristic method (BRKGA-MP-IPR) are listed below.

- *Population size:* 2000;
- *number of independent populations:* 3;
- *mutant percentage in population:* 15%;
- *elite percentage in population:* 30%;

[1]The choice of depth-first over breadth-first was arbitrary.

- *number of parents to mate:* 3;
- *number of elite parents:* 2;
- *minimum distance between chromosomes to path-relink:* 0.15.

5.4 Preliminary Experiments

The results presented in this Section, as well as their analyses, have been published in the Proceedings of the Brazilian Symposium on Operations Research in 2020 [4] and 2022 [5]. For those experiments, we used grids $n \times n$, for $n = 4, 6, 8, 10$, their subgraphs with densities $r \approx 0.2, 0.4, 0.6, 0.8$, OD matrices with uniform distribution, and budget limit of 25%, 50% and 75% of the total cost of the grid or subgraph.

For the analysis of these results, we used the concepts of lower bound (LB), upper bound (UB) and gap, conventionally associated with exact methods for combinatorial optimization. However, we also adapted those to be used when comparing the exact and heuristic approaches. In order to differentiate the conventional use from the adapted concept, we use ILP and BRKGA[2] as subscriptions. Hence, we LB_{ILP}, UB_{ILP} and GAP_{ILP} to refer to the conventional meaning of lower and upper bounds, and gap respectively. We recall that

$$GAP_{ILP} = \frac{UB_{ILP} - LB_{ILP}}{LB_{ILP}}.$$

We define LB_{BRKGA} as the objective value of the best solution found by the BRKGA-MP-IPR heuristic, and GAP_{BRKGA} as

$$GAP_{BRKGA} = \frac{LB_{BRKGA} - LB_{ILP}}{LB_{ILP}};$$

UB_{BRKGA} is not defined.

5.4.1 Analyzing Results for the Exact Method

Tables 5.1-5.3 show the results of the ILP solution for instances with a budget constraint B of 25%, 50% and 75% with respect to the network total cost. The model obtained optimal solutions for most (49 out of 60) instances. Lower and upper bounds were obtained for all instances, except one, namely the instance with 100 vertices, 100% density and 25% budget constraint. The results also show that instances composed of sparse graphs were solved more often than those composed of dense graphs, as all instances with a density of 40% or less were solved to optimality.

Concerning optimality gaps, Figure 5.1 shows the average gap obtained by the methodology for every graph size and budget constraint. The figure clearly shows an exponential

[2]Note that the acronym BRKGA is used here for the sake of simplicity. The metaheuristic used for all experiments was BRKGA-MP-IPR.

Number of nodes	Results	Density				
		20%	**40%**	**60%**	**80%**	**100%**
16	LB_{ILP}	605	782	1,926	2,190	3,799
	UB_{ILP}	605	782	1,926	2,190	3,799
	GAP_{ILP}	0%	0%	0%	0%	0%
36	LB_{ILP}	970	3,767	8,088	11,588	18,581
	UB_{ILP}	970	3,767	8,088	11,588	22,326
	GAP_{ILP}	0%	0%	0%	0%	20.1550%
64	LB_{ILP}	2,391	3,002	23,647	37,720	54,944
	UB_{ILP}	2,391	3,002	23,647	60,509	96,674
	GAP_{ILP}	0%	0%	0%	60.4162%	75.9501%
100	LB_{ILP}	1,810	18,009	46,400	86,599	-
	UB_{ILP}	1,810	18,009	95,376	180,947	266,298
	GAP_{ILP}	0%	0%	105.5517%	108.9491%	-

Table 5.1: Test results for a budget constraint of 25% of the total cost of the network.

Number of nodes	Results	Density				
		20%	**40%**	**60%**	**80%**	**100%**
16	LB_{ILP}	984	2,379	4,420	6,911	10,459
	UB_{ILP}	984	2,379	4,420	6,911	10,459
	GAP_{ILP}	0%	0%	0%	0%	0%
36	LB_{ILP}	1,375	5,956	20,839	42,829	58,575
	UB_{ILP}	1,375	5,956	20,839	42,829	58,575
	GAP_{ILP}	0%	0%	0%	0%	0%
64	LB_{ILP}	3,133	3,002	55,835	127,169	200,134
	UB_{ILP}	3,133	3,002	55,835	137,879	205,621
	GAP_{ILP}	0%	0%	0%	8.4219%	2.7417%
100	LB_{ILP}	1,810	18,009	116,812	320,725	125,342
	UB_{ILP}	1,810	18,009	242,190	372,483	501,178
	GAP_{ILP}	0%	0%	107.3332%	16,1378%	299.8484%

Table 5.2: Test results for a budget constraint of 50% of the total cost of the network.

growth in the average gap with the increase of the graph size. For the instance in which no lower bound was attained, the optimality gap was assumed to be infinite. As for the instances budget, Figure 5.1 indicates that loose budget constraints proved to be the easier to solve, since all but one instance with 75% budget constraint were solved to optimality.

5.4.2 Comparing Exact and Heuristic Approaches

Tables 5.4- 5.6 present results for both methodologies. Taking a closer look at Table 5.4, one can observe that for smaller grids - with 16 and 36 nodes - the BRKGA-MP-IPR approach underperforms the exact method; that is, while the solver for the exact method is able to find the optimal solution, the best solution the metaheuristic obtains is far apart from the optimal one. For the large grids with low density - 20%, 40%, and 60%, and 20% and 40% for networks with 64 and 100 nodes, respectively - BRKGA-MP-IPR also was outperformed by the model. However, for the remaining networks, for which the exact method is not able to find the optimal solution - only lower and upper bounds - during the time limit, the BRKGA-MP-IPR method is able to decrease the gap between those bounds. More precisely, in the case of the 100-node network with 100% density, our

Number of nodes	Results	Density				
		20%	**40%**	**60%**	**80%**	**100%**
16	LB_{ILP}	984	2,379	4,974	11,926	13,342
	UB_{ILP}	984	2,379	4,974	11,926	13,342
	GAP_{ILP}	0%	0%	0%	0%	0%
36	LB_{ILP}	1,375	5,956	22,659	62,037	65,277
	UB_{ILP}	1,375	5,956	22,659	62,037	65,277
	GAP_{ILP}	0%	0%	0%	0%	0%
64	LB_{ILP}	3,133	3,002	55,835	192,993	206,005
	UB_{ILP}	3,133	3,002	55,835	192,993	206,005
	GAP_{ILP}	0%	0%	0%	0%	0%
100	LB_{ILP}	1,810	18,009	125,121	427,150	501,178
	UB_{ILP}	1,810	18,009	125,121	444,717	501,178
	GAP_{ILP}	0%	0%	0%	4.1126%	0%

Table 5.3: Test results for a budget constraint of 75% of the total cost of the network.

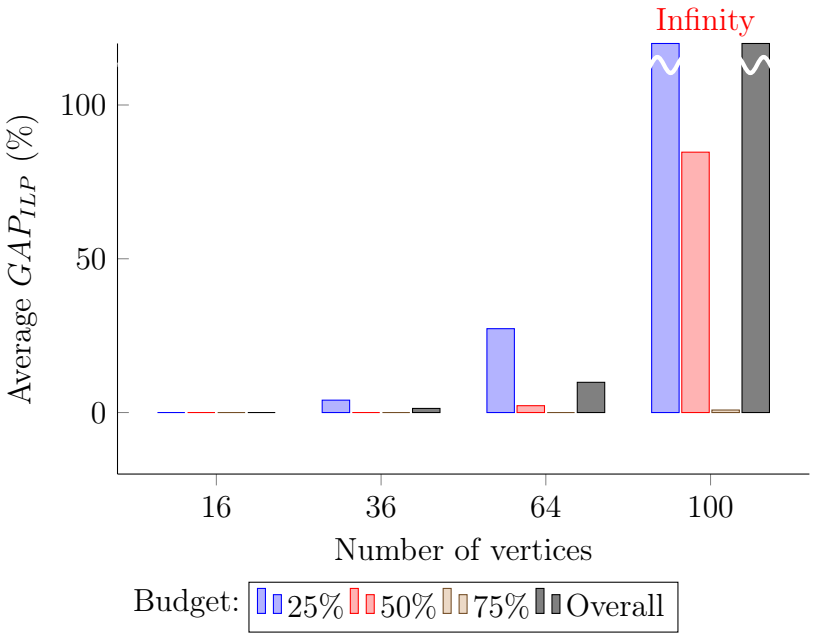

Figure 5.1: Average GAP_{ILP} obtained by the methodology with respect to the instances sizes and budgets.

Number of nodes	Results	Density				
		20%	**40%**	**60%**	**80%**	**100%**
16	LB_{ILP}	605	782	1,926	2,190	3,799
	UB_{ILP}	605	782	1,926	2,190	3,799
	Decoder 1	322	547	1,404	2,190	3,754
36	LB_{ILP}	970	3,767	8,088	11,588	18,581
	UB_{ILP}	970	3,767	8,088	11,588	22,326
	Decoder 1	472	2,845	7,099	11,588	18,357
64	LB_{ILP}	2,391	3,002	23,647	37,720	54,944
	UB_{ILP}	2,391	3,002	23,647	60,509	96,674
	Decoder 1	1,827	2,351	21,351	38,033	56,207
100	LB_{ILP}	1,810	18,009	46,400	86,599	-
	UB_{ILP}	1,810	18,009	95,376	180,947	266,298
	Decoder 1	1,400	16,521	47,784	90,151	135,451

Table 5.4: Test results for a budget constraint of 25% of the total cost of the network.

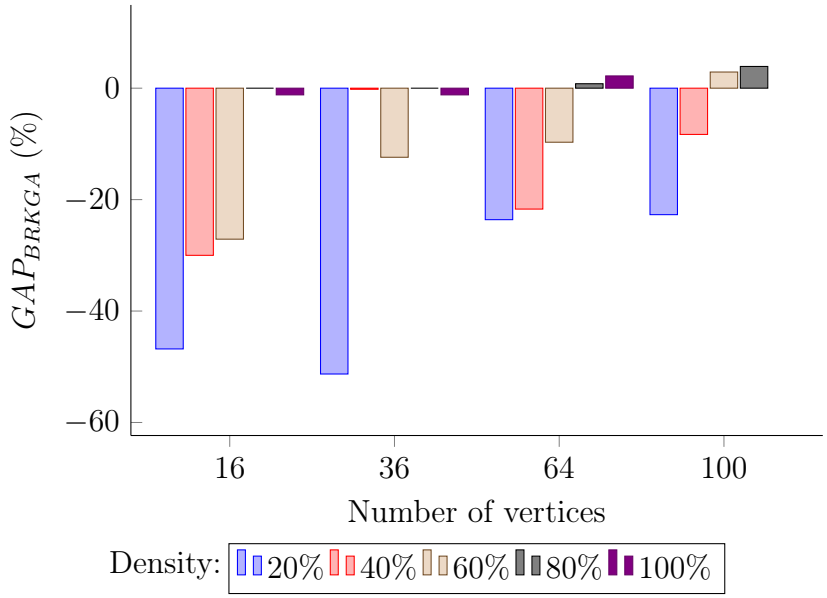

Figure 5.2: GAP_{BRKGA} for different network sizes and densities subjected to the 25% budget constraint.

heuristic is responsible for obtaining a lower bound, as the solver was not able to do so within the time limit imposed in the original work.

In Figure 5.2, we summarize that information by computing the gap between the best feasible solution found by the BRKGA-MP-IPR heuristic and the lower bound found by the exact method - if found - for all networks, under the 25% budget constraint. Figures 5.3 and 5.4 do the same for the other two budget constraints. Bars below the *x-axis* indicate that the BRKGA-MP-IPR has done worse than the exact method; bars above it mean the metaheuristic was able to achieve better results than the solver.

For the 100-node network with 100% density subjected to a budget constraint of 50% of the total network cost, BRKGA-MP-IPR is also able to decrease the gap, very significantly in this case (see Table 5.5 and Figure 5.3).

Number of nodes	Results	Density				
		20%	**40%**	**60%**	**80%**	**100%**
16	LB_{ILP}	984	2,379	4,420	6,911	10,459
	UB_{ILP}	984	2,379	4,420	6,911	10,459
	Decoder 1	716	1,861	3,459	6,911	10,459
36	LB_{ILP}	1,375	5,956	20,839	42,829	58,575
	UB_{ILP}	1,375	5,956	20,839	42,829	58,575
	Decoder 1	804	4,940	18,885	36,536	55,708
64	LB_{ILP}	3,133	3,002	55,835	127,169	200,134
	UB_{ILP}	3,133	3,002	55,835	137,879	205,621
	Decoder 1	2,497	2,351	52,257	121,844	181,094
100	LB_{ILP}	1,810	18,009	116,812	320,725	125,342
	UB_{ILP}	1,810	18,009	242,190	372,483	501,178
	Decoder 1	1,400	16,532	115,055	275,214	414,478

Table 5.5: Test results for a budget constraint of 50% of the total cost of the network.

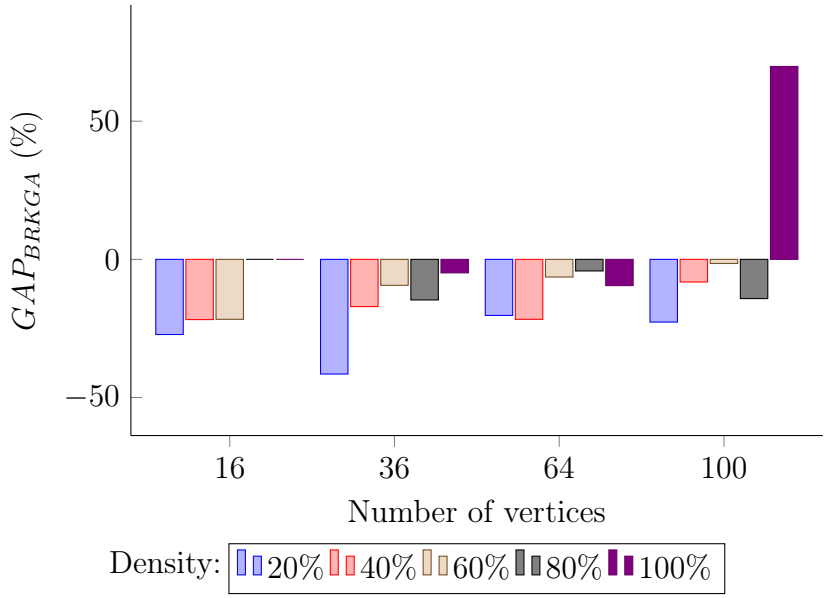

Figure 5.3: GAP_{BRKGA} for different network sizes and densities subjected to the 50% budget constraint.

For the 75% budget constraint, the BRKGA-MP-IPR approach was outperformed in every scenario. However, it is likely that, for larger instances than the ones used in these preliminary experiments, the solver would struggle to find solutions.

This behavior can be explained by the fact the model can easily solve the problem for small instances and "loose" budget constraints. The solver starts struggling as we decrease the budget limit and increase the size of the networks - by either nodes or edges. In those cases, the exact method is not able to find an optimal solution within the time limit, only accomplishing to find upper and lower bounds - except for the network with 100 nodes and 100% density subjected to a budget constraint of 25% of the cost of the network, for which no lower bound was found.

Number of nodes	Results	Density				
		20%	**40%**	**60%**	**80%**	**100%**
16	LB_{ILP}	984	2,379	4,974	11,926	13,342
	UB_{ILP}	984	2,379	4,974	11,926	13,342
	Decoder 1	716	1,861	3,986	11,796	13,342
36	LB_{ILP}	1,375	5,956	22,659	62,037	65,277
	UB_{ILP}	1,375	5,956	22,659	62,037	65,277
	Decoder 1	804	4,940	20,670	62,037	65,277
64	LB_{ILP}	3,133	3,002	55,835	192,993	206,005
	UB_{ILP}	3,133	3,002	55,835	192,993	206,005
	Decoder 1	2,497	2,351	52,257	175,319	199,784
100	LB_{ILP}	1,810	18,009	125,121	427,150	501,178
	UB_{ILP}	1,810	18,009	125,121	444,717	501,178
	Decoder 1	1,400	16,532	115,055	313,025	416,069

Table 5.6: Test results for a budget constraint of 75% of the total cost of the network.

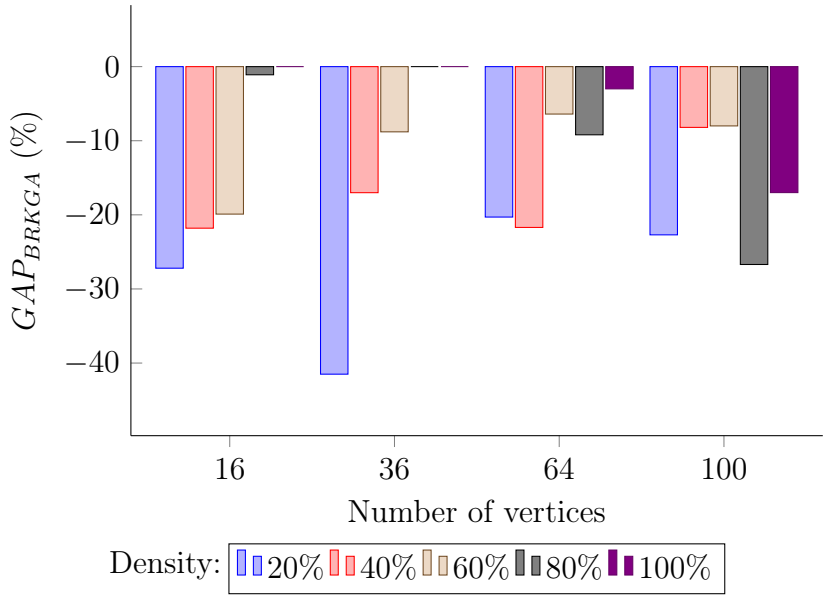

Figure 5.4: GAP_{BRKGA} for different network sizes and densities subjected to the 75% budget constraint.

Based on these experiments, we hypothesize that the two methods - namely, the SLPP ILP model and BRKGA-MP-IPR - complement each other. In order to test this hypothesis, we run more experiments on larger instances. The results are presented in Section 5.5.

5.5 Experiments

For these experiments, we used grids $n \times n$, for $n = 20, 40, 60, 80, 100$, their subgraphs with densities $r \approx 0.2, 0.4, 0.6, 0.8$, OD matrices with uniform distribution and clusterized matrices generated from all three types of seeds, and budget limit of 25%, 50%, and 75%, adding up to 300 instances. Each was solved by the SLPP ILP model and three versions of the BRKGA-MP-IPR metaheuristic.

The SLPP ILP model was unable to solve any of the mentioned instances to optimality within the given time limit of 10 minutes, with most gaps being infinite. This result confirms our hypothesis that the model does not scale well as expected, considering the SLPP is an NP-hard problem (Section 3.2). In the following tables, we summarize and highlight some of the results.

For comparison between the two methods, both the "easiest" and the "hardest" instances have been selected - subgraph with dimension 20×20 and $r \approx 0.2$ with a 75% budget limit (Table 5.7), and grid with dimension 100×100 combined with a 25% budget limit (Table 5.8), as pointed out by the preliminary experiments reported in Section 5.4. For these instances, we chose to only compare the ILP model with the first decoder described in Section 4 as done in the preliminary experiments. Note that, even for the "easiest" instance, the ILP model is outperformed by BRKGA-MP-IPR.

For the "hardest", the solver runs out of memory before being able to complete the first iteration[3], while BRKGA-MP-IPR is able to find feasible solutions.

Matrix Distribution	**ILP model**			**BRKGA-MP-IPR**	
	LB_{ILP}	UB_{ILP}	GAP_{ILP}	LB_{BRKGA}	GAP_{BRKGA}
Uniform	100	438,160	4,380.6%	4,314	42.1%
Clusterized (random)	1,396	10,791,079	7,728.9%	125,208	88.6%
Clusterized (border-center)	1,405	10,852,834	7,723.4%	127,035	89.4%
Clusterized (border-border)	0	10,782,764	∞	129,690	∞

Table 5.7: Results for the "easiest" instance, namely subgraph with dimension 20×20 with $r \approx 0.2$ and a 75% budget limit.

[3]Note that a 100×100 grid has 10,000 nodes, hence, its associated OD matrix has dimension $10,000 \times 10,000$.

Matrix Distribution	ILP model			BRKGA-MP-IPR	
	LB_{ILP}	UB_{ILP}	GAP_{ILP}	LB_{BRKGA}	GAP_{BRKGA}
Uniform	-	-	-	20,453,890	-
Clusterized (random)	-	-	-	1,315,470,195	-
Clusterized (border-center)	-	-	-	1,324,480,950	-
Clusterized (border-border)	-	-	-	1,933,826,486	-

Table 5.8: Results for the "hardest" instance, namely grid with dimension 100×100 combined with a 25% budget limit.

The model was also unable to solve all instances with 80×80 grid or subgraph. For instances with grids of dimensions 40×40 and 60×60, in most cases, the model was able to find a lower bound, but not an upper bound.

In order to illustrate the difference in performance of the three decoders, we chose 36 different instances. These instances are listed next; they all have a 25% budget.

- $n = 20$, $r = 1$, combined with all four matrix types;
- $n = 40$, $r = 1$, combined with all matrix types;
- $n = 60$, $r = 1$, combined with all matrix types;
- $n = 80$, $r = 1$, combined with all matrix types;
- $n = 100$, $r \approx 0.8$, combined with all matrix types;
- $n = 100$, $r \approx 0.6$, combined with all matrix types;
- $n = 100$, $r \approx 0.4$, combined with all matrix types;
- $n = 100$, $r \approx 0.2$, combined with all matrix types.

Matrix Distribution	n					Decoder
	20	**40**	**60**	**80**	**100**	
Uniform	1,953,549	8,977,808	12,454,531	19,109,046	16,289,779	**Decoder 1**
	1,996,527	9,390,787	13,600,347	21,172,823	18,391,160	**Decoder 2**
	0	0	0	0	0	**Decoder 3**
Clusterized (random)	49,891,800	480,872,350	604,703,669	1,126,491,810	1,315,470,195	**Decoder 1**
	51,538,229	502,030733	660,336,407	1,1,252,658,890	1,493,058,670	**Decoder 2**
	0	0	0	0	0	**Decoder 3**
Clusterized (border-center)	49,780,490	374,300,955	858,289,224	1,129,587,105	1,324,480,950	**Decoder 1**
	51,672,148	396,384,711	943,259,857	1,253,841,690	1,500,636,920	**Decoder 2**
	0	0	0	0	0	**Decoder 3**
Clusterized (border-border)	49,521,420	361,276,367	773,557,533	1,135,571,075	1,370,069,085	**Decoder 1**
	51,106,105	385,843,160	845,498,384	1,277,534,600	1,534,477,385	**Decoder 2**
	0	0	0	0	0	**Decoder 3**

Table 5.9: Results obtained by the three different decoders for instances with $r = 1$, a 25% budget, and n as indicated on the Table.

Matrix Distribution	r					Decoder
	0.2	**0.4**	**0.6**	**0.8**	**1**	
Uniform	16,683	122,427	527,889	4,729,811	16,289,779	**Decoder 1**
	16,683	122,427	527,889	4,971,031	18,391,160	**Decoder 2**
	0	0	0	0	0	**Decoder 3**
Clusterized (random)	1,212,804	6,376,443	37,017,684	340,069,293	1,315,470,195	**Decoder 1**
	1,212,804	6,376,443	37,017,684	347,890,887	1,493,058,670	**Decoder 2**
	0	0	0	0	0	**Decoder 3**
Clusterized (border-center)	1,188,918	6,331,220	40,411,600	443,715,624	1,324,480,950	**Decoder 1**
	1,188,918	6,331,220	40,411,600	457,470,808	1,500,636,920	**Decoder 2**
	0	0	0	0	0	**Decoder 3**
Clusterized (border-border)	1,198,008	8,534,784	61,943,242	288,078,036	1,370,069,085	**Decoder 1**
	1,198,008	8,534,784	61,943,242	296,144,221	1,534,477,385	**Decoder 2**
	0	0	0	0	0	**Decoder 3**

Table 5.10: Results obtained by the three different decoders for instances with $n = 100$, a 25% budget, and r as indicated on the Table.

The first thing that comes to attention is the fact that decoder 3 is not successful in finding a feasible solution. The probable cause is that most chromosomes do not translate to viable solutions. For instance, half of the existing chromosomes translate to a graph with no edges; from the other half, graphs that have more than one component are not considered a valid solution. Thus, the decoder struggle to find feasible solutions, causing it to converge at a much slower rate than the other two. We conclude that, although the decoder theoretically removes bias, it is not worth the resulting slow convergence rate. A possible solution would be to increase the probability of an edge being present, that is, increase the size of the interval that is connected to the presence of edges - consequently decreasing the interval linked to the absence of edges -, and considering only the largest component of a graph with multiple components as a candidate solution.

Note that, in Table 5.10, decoders 1 and 2 obtained the same solution for $r \approx 0.2, 0.4, 0.6$. This likely occurred due to the fact that the absence of edges limited the options of paths between two vertices such that the paths $p = (v_1, \ldots, v_n)$ found by decoder 1 were already the shortest paths between v_1 and v_n. This hypothesis is further sustained by the fact that decoder 2 was able to attain better results than decoder 1 for all grids with $r = 1$.

Chapter 6

Final Remarks

We would like to start these final remarks by reiterating that, although the motivation for this work is very practical, our model simplifies and abstracts the addressed problem. We are aware that there are other variables when planning a public transit line that have not been accounted for by our model. For instance, the length of the line and the time to travel between its two ends. It does not sound practical to have an intraurban bus line that is 100 km long and takes five hours to be traversed, for example.

Furthermore, even if our instances do a fair job of mimicking real street networks and real OD matrices, they will never be as good as data from real cities, particularly where network size is concerned. It is important to notice, however, that both methodologies can handle real instance data - although the ILP model will not cope well with large instances, as we have shown in this work.

Moreover, we finish this dissertation with quite a few ideas for future work. First and foremost, we would like to experiment with real data. It is worth noting however that those are not easy to obtain, specially OD matrices. We also wish to experiment with new decoders and improving the current ones. For instance, the edge-based decoder can be modified to guarantee that all chromosomes are transcribed to feasible solutions. Additionally, we would like to test combining the two techniques, for instance, by providing the solutions obtained heuristically as a warm start for the solver and observe how it impacts its performance.

Finally, we conclude by stating that our main hypothesis has been confirmed and the two approaches are in fact complimentary. However, we would like to highlight that, due to its low scalability, the ILP model cannot realistically be used for real world data, as shown in Section 5.5.

Bibliography

[1] Roads, transit, and the denseness of são paulo's urban development. *MIT Center for Real Estate Research Paper*, 2021.

[2] Carlos E Andrade, Luciana S Pessoa, and Slawomir Stawiarsk. The physical cell identity assignment problem: a practical optimization approach. *IEEE Transactions on Evolutionary Computation*, 2022.

[3] Carlos E. Andrade, Rodrigo F. Toso, José F. Gonçalves, and Mauricio G.C. Resende. The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. *European Journal of Operational Research*, 289(1):17–30, 2021.

[4] Julia R. Beltrão and Fábio L. Usberti. Using optimization to design public urban transportation networks. In *Proceedings of the Brazilian Symposium on Operations Research*, 2020.

[5] Julia R. Beltrão and Fábio L. Usberti. An evolutionary approach for the optimal design of public transportation networks. In *Proceedings of the Brazilian Symposium on Operations Research*, 2022.

[6] Adrian Bondy and U.S.R. Murty. *Graph theory*. Graduate texts in mathematics 244. Springer, 3rd corrected printing. edition, 2008.

[7] Ralf Borndörfer, Martin Grötschel, and Marc E. Pfetsch. A column-generation approach to line planning in public transport. *Transportation Science*, 41(1):123–132, 2007.

[8] Michael R. Bussieck, Thomas Lindner, and Marco E. Lübbecke. A fast algorithm for near cost optimal line plans. *Mathematical Methods of Operations Research*, 59(2):205–220, June 2004.

[9] Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics 173. Springer-Verlag Berlin Heidelberg, 5 edition, 2017.

[10] Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman, 1979.

[11] Fred Glover, Manuel Laguna, and Rafael Martí. Fundamentals of scatter search and path relinking. *Control and cybernetics*, 29(3):653–684, 2000.

[12] José Fernando Gonçalves and Mauricio GC Resende. Biased Random-Key Genetic Algorithms for Combinatorial Optimization. *Journal of Heuristics*, 17(5):487–525, 10 2011.

[13] Margardida C. Gordinho. *Transportes no Brasil*. Editora Marca d'Água, 2003.

[14] Konstantinos Kepaptsoglou and Matthew Karlaftis. Transit route network design problem: Review. *Journal of Transportation Engineering*, 135:491–505, 08 2009.

[15] Alberto Kummer, Olinto de Araújo, Luciana Buriol, and Mauricio Resende. A biased random-key genetic algorithm for the home health care problem. *arXiv preprint arXiv:2206.14347*, 2022.

[16] Gilbert Laporte, Ángel Marín, Juan A. Mesa, and Francisco A. Ortega. An integrated methodology for the rapid transit network design problem. In *Algorithmic Methods for Railway Optimization*, pages 187–199. Springer Berlin Heidelberg, 2007.

[17] Mariana A Londe, Carlos E Andrade, and Luciana S Pessoa. An evolutionary approach for the p-next center problem. *Expert Systems with Applications*, 175:114728, 2021.

[18] Marcus Lopes. Como nasceu o primeiro sistema de transporte coletivo do mundo. *BBC News Brasil*. Available: `https://www.bbc.com/portuguese/geral-45587611` [Last access: 12 Dec 2022].

[19] Marina L. Lucena, Carlos E. Andrade, Mauricio GC Resende, and Flávio K. Miyazawa. Some extensions of biased random-key genetic algorithms. In *Proceedings of the 46th Brazilian Symposium of Operational Research*, pages 1–12, 2014.

[20] Celso C. Ribeiro, Jose A. Riveaux, and Julliany S. Brandao. Biased random-key genetic algorithms using path-relinking as a progressive crossover strategy. In *2021 5th International Conference on Intelligent Systems, Metaheuristics amp; Swarm Intelligence*, ISMSI 2021, page 28–36, New York, NY, USA, 2021. Association for Computing Machinery.

[21] Kenneth Sörensen and Fred W. Glover. *Metaheuristics*. Springer US, Boston, MA, 2013.

[22] UITP. Urban public transport in the 21st century. *uitp.org/*. Available: `http://www.uitp.org/urban-public-transport-21st-century` [Last access: 29 Jan 2020]., 2017.

[23] Vukan R. Vuchic. *Urban Transit Systems and Technology*. Wiley, 2007.

[24] Eric W. Weisstein. Grid graph. *From MathWorld–A Wolfram Web Resource*. Available: `https://mathworld.wolfram.com/GridGraph.html` [Last access: 4 Dec 2022].

[25] Amirali Zarrinmehr, Mahmoud Saffarzadeh, Seyedehsan Seyedabrishami, and Yu Marco Nie. A path-based greedy algorithm for multi-objective transit routes design with elastic demand. *Public Transport*, 8(2):261–293, September 2016.

[26] Ángel Marín and Patricia Jaramillo. Urban rapid transit network design: accelerated Benders decomposition. *Annals of Operations Research*, 169(1):35–53, July 2009.