**UNIVERSIDADE FEDERAL DO CEARÁ**

**CAMPUS DE RUSSAS**

**CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**YAN RODRIGUES DA SILVA**

# A APLICAÇÃO DO PROBLEMA DE BALANCEAMENTO DE BICICLETAS COMPARTILHADAS PARA RESOLUÇÃO DE UM CASO REAL DO BICICLETAR

**RUSSAS**

**2025**

YAN RODRIGUES DA SILVA

# A APLICAÇÃO DO PROBLEMA DE BALANCEAMENTO DE BICICLETAS COMPARTILHADAS PARA RESOLUÇÃO DE UM CASO REAL DO BICICLETAR

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em ciência da computação do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em ciência da computação.

Orientadora: Prof. Dr. Tatiane Fernandes Figueredo.

RUSSAS

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

S584a Silva, Yan Rodrigues da Silva.
A aplicação do problema de balanceamento de bicicletas compartilhadas para resolução de um caso real do Bicicletar / Yan Rodrigues da Silva Silva. – 2025.
56 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2025.
Orientação: Profa. Dra. Tatiane Fernandes Figueredo.

1. Sistemas de compartilhamento de bicicletas.. 2. Programação Linear Inteira.. 3. Problema de rebalanceamento.. I. Título.

CDD 005

---

YAN RODRIGUES DA SILVA

# A APLICAÇÃO DO PROBLEMA DE BALANCEAMENTO DE BICICLETAS COMPARTILHADAS PARA RESOLUÇÃO DE UM CASO REAL DO BICICLETAR

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em ciência da computação do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em ciência da computação.

Aprovada em: 05/03/2025.

BANCA EXAMINADORA

---

Prof. Dr. Tatiane Fernandes Figueredo (Orientadora)
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Eurinardo Rodrigues Costa
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Cristiano Martins Monteiro
Universidade Federal de Minas Gerais (UFMG)

À minha família, por acreditar em mim mesmo quando eu duvidava. Mãe, seu cuidado foi meu alento. Pai, sua presença, meu abrigo. A vocês, raiz e sustento, dedico este trabalho e cada passo dessa jornada.

## AGRADECIMENTOS

Agradeço, em primeiro lugar, à Profa. Dra. Tatiane Fernandes Figueiredo, por sua orientação atenta, apoio constante e contribuições ao longo deste trabalho.

À minha família, em especial Lúcia de Fátima e Jorge Luís, por serem fonte de força, amor e incentivo durante toda a minha caminhada acadêmica.

Aos professores que fizeram parte da minha formação, pelo compromisso com o ensino e por cada lição transmitida dentro e fora da sala de aula.

Aos amigos que caminharam comigo nesta jornada: Samuel Jonas, por dividir não só a casa, mas também os desafios e aprendizados da graduação; Alysson Lucas, Paulo Henrique, Naum Josafá, Mateus Daniel, Pedro Ítalo e Jaziel Loureiro, por suas amizades sinceras, companheirismo e apoio nos momentos em que mais precisei.

A todos vocês, minha sincera gratidão.

“Eu falhei. Eu falhei outra vez. Não importa. Tente de novo. Falhe de novo. Falhe melhor.”
(Samuel Beckett)

## RESUMO

Os impactos ambientais significativos causados pelos meios de transporte público e privado em regiões metropolitanas têm motivado a busca por alternativas mais sustentáveis de mobilidade urbana. Uma das soluções adotadas em diversas cidades ao redor do mundo é a implementação de sistemas de compartilhamento de bicicletas. Nesse modelo, os usuários podem retirar bicicletas em estações distribuídas pela cidade e devolvê-las em qualquer outra estação disponível, promovendo flexibilidade no deslocamento. No entanto, essa característica gera um desafio operacional relacionado ao desbalanceamento entre as estações, resultando em excesso de bicicletas em algumas localidades e escassez em outras. Diante desse cenário, este trabalho propõe a aplicação de dois modelos de Programação Linear Inteira ao Bicicletar, sistema de compartilhamento de bicicletas da cidade de Fortaleza. Além disso, realiza-se uma análise comparativa dos resultados obtidos com a aplicação dos modelos, buscando avaliar sua eficácia no processo de rebalanceamento das estações.



**Palavras-chave**: sistemas de compartilhamento de bicicletas; programação linear inteira; problema de rebalanceamento

## ABSTRACT

The significant environmental impacts caused by public and private transportation in metropolitan areas have motivated the search for more sustainable urban mobility alternatives. One widely adopted solution in cities around the world is the implementation of bicycle-sharing systems. In this model, users can pick up bicycles at designated stations distributed throughout the city and return them at any other available station, allowing flexible travel. However, this feature introduces an operational challenge related to the imbalance between stations, often resulting in an excess of bicycles in some areas and shortages in others. In this context, this study proposes the application of two Integer Linear Programming models to Bicicletar, the bicycle-sharing system in the city of Fortaleza, Brazil. Furthermore, a comparative analysis of the results obtained from the application of the proposed models is conducted, aiming to assess their effectiveness in the station rebalancing process.



**Keywords**: bike sharing systems; integer linear programming; rebalancing problem

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ALGORITMOS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AMC | Autarquia Municipal de Trânsito e Cidadania |
| B&B | *Branch-and-Bound* |
| B&C | *Branch-and-Cut* |
| CSV | *Comma-separated values* |
| JSON | *JavaScript Object Notation* |
| mTSP | *The Multiple Traveling Salesman Problem* |
| PBESCB | Problema de Balanceamento Estático em Sistemas de Compartilhamento de Bicicletas |
| PL | Programação Linear |
| PLI | Programação Linear Inteira |
| PO | Pesquisa Operacional |
| PRV | Problema de Roteirização de Veículos |
| PRVCE | Problema de Roteirização de Veículos com Coleta e Entrega |
| SCB | Sistema de Compartilhamento de Bicicletas |
| SCSP | Secretaria Municipal da Conservação e Serviços Públicos |

**SUMÁRIO**

# 1 INTRODUÇÃO

Com a crescente preocupação em relação aos impactos ambientais, especialmente aqueles gerados pelo setor de transporte, várias cidades ao redor do mundo têm apresentado soluções inovadoras para sistemas de transporte sustentáveis. Uma alternativa notável para esse contexto são os Sistemas de Compartilhamento de Bicicletas (SCBs).

Conforme destacado por Mora e Moran (2022), os SCBs oferecem aos usuários a possibilidade de alugar temporariamente uma bicicleta, com a possibilidade de devolvê-la em diferentes pontos da cidade. Essa abordagem elimina as preocupações relacionadas à posse da bicicleta, ao mesmo tempo em que reduz problemas como roubo. A adoção desse sistema tem impactos significativos ao promover a utilização de meios de transporte alternativos, contribuindo para a diminuição das emissões de gases do efeito estufa e na melhoria da saúde dos usuários (DEMAIO, 2009).

A proposta central dos SCBs é estabelecer uma rede de estações distribuídas em vários pontos de uma cidade. Cada estação possui um número fixo de vagas onde os usuários podem retirar ou devolver bicicletas. De maneira geral, os usuários iniciam e finalizam seus percursos em diferentes estações. A localização das estações leva em consideração diversos fatores, como centros comerciais, áreas de grande circulação de pessoas, estações de metrô, entre outros.

Contudo, apesar da utilidade e dos benefícios associados ao uso do sistema, surge uma problemática intrínseca ao SCB. Durante um determinado período de uso, é esperado que algumas estações possuam mais bicicletas do que outras, fazendo com que certas estações precisem ser reabastecidas. Essa operação de reabastecimento, chamada de rebalanceamento, é realizada por um veículo ou uma frota de veículos capacitados. É de extrema importância encontrar rotas que sejam economicamente mais vantajosas e eficientes para o(s) veículo(s) capacitado(s), considerando as demandas por bicicletas nas diferentes estações, a fim de garantir a usabilidade e disponibilidade do sistema.

Buscando abordar a problemática citada, este trabalho apresenta duas formulações matemáticas, além de aplicar um algoritmo para resolver essas formulações no SCB de Fortaleza, o sistema Bicicletar. Implementado em 2014, o Bicicletar é fruto de uma parceria público-privada entre a Prefeitura de Fortaleza e a Unimed e segue em expansão, contando atualmente com mais de 200 estações.

Para uma compreensão mais aprofundada, esta monografia está estruturada da se-

guinte maneira: o Capítulo 2 apresenta os objetivos gerais e específicos deste trabalho; o Capítulo 3 aborda as teorias essenciais para a compreensão do tema; no Capítulo 4 são discutidos os trabalhos relacionados ao problema em questão; o Capítulo 5 detalha as etapas metodológicas seguidas na realização do trabalho proposto; o Capítulo 6 apresenta as soluções obtidas e as análises decorrentes delas; por fim, o Capítulo 7 discute as conclusões e trabalhos futuros.

# 2 OBJETIVOS

## 2.1 Objetivo geral

Aplicar técnicas de Programação Linear Inteira para solucionar o Problema de Rebalanceamento em Sistemas de Compartilhamento de Bicicletas no Bicicletar, gerando rotas para veículos capacitados e garantindo o rebalanceamento das estações.

### *2.1.1 Objetivos específicos*

- Estudar os modelos de Programação Linear Inteira aplicados ao problema de rebalanceamento em sistemas de compartilhamento de bicicletas existentes na literatura.
- Apresentar os modelos de Programação Linear Inteira especificamente para o problema de rebalanceamento em sistemas de compartilhamento de bicicletas.
- Apresentar um algoritmo de resolução para as formulações matemáticas propostas.
- Gerar uma base de dados utilizando os dados obtidos a partir do sistema Bicicletar.
- Aplicar os modelos sobre a base de dados gerada.
- Analisar as rotas obtidas e gerar *insights* significativos sobre as soluções propostas.

# 3 FUNDAMENTAÇÃO TEÓRICA

Com o intuito de apresentar uma fundamentação matemática básica para uma melhor compreensão do problema abordado neste trabalho, este capítulo apresenta alguns conceitos das áreas de Teoria dos Grafos e Pesquisa Operacional.

## 3.1 Teoria dos grafos

A Teoria dos Grafos se originou a partir dos trabalhos de L. Euler. Em 1736, Euler apresentou um problema conhecido como o "problema das pontes de Königsberg". Königsberg era uma cidade que possuía sete pontes, as quais atravessavam um rio, conectando vários locais da cidade. O problema se tratava de encontrar um percurso que permitisse atravessar a cidade, começando e terminando no mesmo local, enquanto cruzava cada ponte uma única vez.

A Teoria dos Grafos tem uma ampla gama de aplicações em campos como física, química e diversas outras áreas do conhecimento. De maneira geral, essa teoria nos permite modelar diversos problemas por meio de conceitos relacionados a grafos (PRESTES, 2020).

Um grafo, denotado por $G = (V(G), E(G))$, é definido como um par ordenado que consiste em dois componentes principais: um conjunto finito não vazio de vértices $(V(G))$ e um conjunto de pares não ordenados de elementos distintos de vértices, chamado de arestas $(E(G))$, isto é, $E(G) \subseteq \{\{u,v\} \mid u,v \in V(G), u \neq v\}$. Para simplificar a notação, adota-se $V$ e $E$ para $V(G)$ e $E(G)$ respectivamente. Além disso, quando se refere a uma aresta $e = \{u,v\}$, diremos que "$e$ incide em $u$ e $v$" ou que "$u$ e $v$ são adjacentes" e, simplificamos a notação descrevendo $e = \{u,v\}$ como simplesmente $uv$.

Pode-se definir também a noção de um grafo orientado (digrafo), denotado por $D = (V,A)$, que consiste em um par ordenado contendo um conjunto de vértices $V(D)$ e um conjunto de arcos ou arestas orientadas $A(D)$, onde os arcos são pares ordenados que conectam vértices distintos em $V$, isto é, $A \subseteq \{(u,v) \mid u,v \in V(D), u \neq v\}$. É comum usar a notação $V$ e $A$ assim como em um grafo não orientado. Se $a = (u,v)$ é um arco, diremos que "$a$ incide em $u$ e $v$" ou que "$u$ e $v$ são adjacentes". É usual referir-se ao vértice $u$ como origem e $v$ como destino ou término do arco $a$.

A *adjacência* ou *vizinhança* de um vértice $v$ de um grafo $G$, denotado por $N_G(v)$, é o conjunto de todos os vértices que possuem uma aresta ligada a $v$. Utilizamos o conceito de vizinhança para definir o grau de um vértice, denotado por $d_G(v)$, onde $d_G(v) = |N_G(v)|$.

Um *passeio* é definido como uma sequência $(v_0, e_1, v_1, e_2, \cdots, e_n, v_n)$ de vértices e arestas, onde, para $1 \leq i \leq n$ a aresta $e_i$ conecta os vértices $v_{i-1}$ e $v_i$. Tanto as arestas quanto os vértices podem ou não ser distintos. Se apenas as arestas forem distintas, então o passeio é chamado de *trilha*. Um *caminho* é um caso especial de *trilha* onde tanto as arestas e vértices são distintos. Portanto, podemos estabelecer a seguinte hierarquia:

$$caminho \subset trilha \subset passeio$$

Quando um *passeio* ou uma *trilha* termina nos mesmos vértices de origem e destino, isto é, $v_0 = v_n$, ele é denominado *passeio fechado* ou *trilha fechada*, respectivamente. Uma *trilha fechada* é conhecida como *circuito*, que é um *passeio fechado* que não repete nenhuma aresta. Por fim, um *circuito* que não repete nenhum vértice é referido como *ciclo*, exceto $v_0$ e $v_n$.

Com a noção de *passeios*, *trilhas* e *ciclo* é possível definir a noção de conectividade em um grafo. Um grafo *G* é dito *conexo* quando há um caminho entre cada par de vértices.

Por fim, outra noção necessária de Teoria dos Grafos é de *grafos ponderados*. Um grafo *G* ou dígrafo *D* é considerado *ponderado* quando cada uma de suas arestas ou arcos possui um valor real associado a ela. Isso significa que mapeamos cada aresta a valores reais por meio de uma função w, onde $w : \mathbb{E}(\mathbb{G}) \to \mathbb{R}$ ou $w : \mathbb{E}(\mathbb{D}) \to \mathbb{R}$.

## 3.2 Pesquisa operacional

A Pesquisa Operacional (PO), como o nome sugere, trata-se de "pesquisa sobre operações", o que significa que essa abordagem pode ser aplicada a uma variedade de situações que envolvem a gestão e coordenação de atividades em diferentes contextos organizacionais. A pesquisa operacional tem encontrado aplicações em diversos setores, abrangendo desde a indústria manufatureira, passando pelo transporte, construção, telecomunicações, planejamento financeiro, assistência médica, até mesmo questões militares e serviços públicos, entre muitos outros (HILLIER; LIEBERMAN, 2013).

Existem várias categorias em que um problema de PO pode ser categorizado, como, problemas de Programação Linear, Programação Não-Linear e Teoria dos jogos. Contudo, nesse trabalho, o foco principal está na aplicação de técnicas de Programação Linear Inteira (PLI).

### *3.2.1 Programação linear*

Segundo Dantzig (2002), a Programação Linear (PL) é um pilar na revolução e no progresso tecnológico que permitiu à humanidade definir objetivos gerais e traçar um caminho de decisões detalhadas a serem tomadas a fim de alcançar, de maneira ótima, os objetivos perante a situações práticas de grande complexidade. Esse progresso é possível por meio da formulação de problemas do mundo real em termos matemáticos (modelos), do desenvolvimento de técnicas para resolver esses modelos (algoritmos) e de um ambiente computacional para a execução dos algoritmos (computadores e softwares).

De acordo com Corrar e Garcia (2001), a formulação de um modelo matemático de um problema de programação linear possui a seguinte estrutura:

Maximizar ou minimizar Z, onde:

$$
\begin{gathered}
\mathrm{Z} = \sum_{i=1}^{n} C_i X_i \\
\text{Sujeito a} \\
\mathrm{a}_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n < b_1 \\
\mathrm{a}_{21} x_1 + a_{22} x_2 + \cdots + a_{2n} x_n < b_2 \\
\cdots \\
\mathrm{a}_{m1} x_n + a_{m2} x_n + \cdots + a_{mn} x_n < b_n \\
\mathrm{x}_i \geq 0, \quad \forall i = 1, 2, \ldots, n \\
\mathrm{b}_j \geq 0, \quad \forall j = 1, 2, \ldots, m
\end{gathered}
$$

De forma que, Z representa a função objetivo que deverá ser maximizada ou minimizada, $x_i$ são as variáveis de decisão que representam os recursos a serem determinados $(x_i \in \mathbb{R})$, $C_i$ são os coeficientes de custos associados a cada variável $x_i$, geralmente atribuídos na formulação do problema, $b_j$ denota a quantidade disponível de recursos e $a_{ij}$ a quantidade de recurso associada a cada variável de decisão.

O método Simplex, desenvolvido por Dantzig (1953), é a técnica utilizada para solucionar problemas que podem ser formulados como modelos lineares (problemas de PL).

### *3.2.2 Programação linear inteira*

Qualquer problema de otimização no qual as variáveis de decisão são discretas pode ser categorizado como um problema de programação linear inteira (PLI). A otimização

inteira e suas técnicas para a resolução de problemas do mundo real têm evoluído em conjunto com o desenvolvimento da área de PO, especialmente quando se trata de programação linear (TAHA, 1975). Há dois casos particulares da PLI que são amplamente utilizados na resolução de problemas. Onde o primeiro acontece quando apenas algumas variáveis de decisão são inteiras (Programação Linear Inteira Mista). O segundo caso ocorre quando as variáveis de decisão assumem apenas valores 0 ou 1 (Programação Linear Inteiro Binário) (WOLSEY, 2020).

Contudo, apesar do uso da PLI nos variados tipos de problemas, de acordo com Karp (1972), a programação linear inteira está na classe de problemas NP-Completos. Isso implica que, dada uma instância de entrada suficientemente grande, os algoritmos, conhecidos atualmente, que se propõem a resolver problemas de PLI necessitam, no pior caso, de tempo exponencial.

## 3.3 Problema de balanceamento estático em sistemas de compartilhamento de bicicletas

O Problema de Balanceamento Estático em Sistemas de Compartilhamento de Bicicletas (PBESCB), conforme definido por Raviv *et al.* (2013), consiste em determinar rotas para uma frota de veículos capacitados que parte de um depósito central, sejam eles carregados ou não com bicicletas. Esse processo leva em conta a necessidade de remover ou adicionar bicicletas entre as estações, considerando as capacidades individuais de cada veículo, assim como o retorno obrigatório ao depósito central.

O problema é estático, pois se refere ao fato do rebalanceamento ocorrer durante o período noturno, quando há uma diminuição dos usuários do sistema de compartilhamento de bicicletas, ou mesmo quando o sistema não está disponível para uso. A versão dinâmica do problema leva em consideração o rebalanceamento durante o dia, enquanto o fluxo de bicicletas e usuários muda rapidamente.

### *3.3.1 Classe de problema do PBESCB*

O PBESCB está inserido em classes de problemas amplamente abordadas na literatura, destacando-se principalmente na classe de problemas de roteirização de veículos. O problema de rebalanceamento é, fundamentalmente, baseado no Problema de Roteirização de Veículos (PRV), que se concentra na otimização de rotas para veículos, visando minimizar custos operacionais e maximizar a eficiência logística.

O PRV foi inicialmente abordado por Dantzig e Ramser (1959) no problema conhecido como *Truck Dispatching Problem*. Nesse problema, os autores desenvolveram um modelo no qual uma frota de caminhões homogêneos busca atender à demanda por óleo de diversos postos de gasolina. Essa tarefa é realizada a partir de uma central, visando minimizar a distância total percorrida. Posteriormente, o PRV foi formalizado por Clarke e Wright (1964), dando origem à noção de roteirização de veículos.

Ao longo do tempo, diversos aspectos teóricos relacionados ao PRV foram abordados. Em especial, Lenstra e Kan (1981) mostraram que o problema pertence à classe de problemas NP-Difícil. Isso implica que, algoritmos exatos conhecidos atualmente são eficientes apenas para instâncias pequenas, o que vai de contramão à escala dos problemas do cotidiano. Nesse sentido, as abordagens utilizando heurísticas e meta-heurísticas se tornam relevantes para soluções possivelmente aproximadas.

Especialmente, o problema de rebalanceamento está inserido em uma classe específica de problema conhecida como Problema de Roteirização de Veículos com Coleta e Entrega (PRVCE). O PRVCE é uma variante do tradicional PRV, onde os veículos não se limitam apenas à entrega de mercadorias, mas também realizam a coleta de itens nos locais dos clientes (NAGY; SALHI, 2005). No contexto do PBESCB, a frota de veículos atende às demandas das estações, onde alguns nós da rede necessitam coletar uma certa quantidade de bicicletas, enquanto outros nós exigem a entrega de bicicletas.

Por fim, seguindo a notação proposta por Berbeglia *et al.* (2007), o PBESCB pode ser classificado como um problema de coleta e entrega de muitos para muitos (M-M). Problemas M-M não possuem distinção entre os vértices de coleta e entrega, em outras palavras, todos os nós podem desempenhar tanto o papel de ponto de coleta quanto de ponto de entrega.

# 4 TRABALHOS RELACIONADOS

Neste capítulo é apresentada uma revisão dos trabalhos relevantes existentes na literatura de modo a contextualizar o problema abordado.

## 4.1 PBESCB com frota homogênea de veículos capacitados

(DELL'AMICO *et al.*, 2014) introduz o PBESCB, onde uma frota homogênea de veículos capacitados é responsável por redistribuir bicicletas entre várias estações, visando minimizar o custo total das rotas tomadas pelos veículos. A princípio, o trabalho foi conduzido utilizando dados obtidos do sistema de compartilhamento de bicicletas da cidade de Reggio Emilia, Itália.

Primeiramente, os autores estiveram em contato com uma análise preliminar de um conjunto de dados de sete meses para entender as particularidades do sistema, assim como o fluxo de saídas e chegadas nas diferentes estações, em toda a cidade. Em seguida, foram apresentadas quatro formulações de programação linear inteira mista com um número exponencial de restrições.

Todas as formulações têm como base o trabalho anterior de Bektas (2006), que abordou o problema em inglês como *The Multiple Traveling Salesman Problem* (mTSP), onde, no máximo, M veículos não capacitados precisam partir de um depósito central e visitar um conjunto de vértices de maneira a visitá-los apenas uma vez. As formulações acrescentam e/ou modificam algumas restrições originais.

Por fim, os autores propõem o uso do método de *Branch-and-Cut* (B&C) nas formulações desenvolvidas. Após obterem resultados na cidade de Reggio Emilia, a pesquisa foi expandida para diversas cidades ao redor do mundo.

## 4.2 PBESCB com frota heterogênea de veículos capacitados e diferentes tipos de bicicletas

Há uma variedade de sistemas de compartilhamento de bicicletas, onde cada um adapta particularidades e necessidades da cidade onde é implementado. Nesse contexto, o estudo de Chastre e Andrade (2023) foca na aplicação do PBESCB na cidade de Lisboa.

O trabalho desenvolvido por Chastre e Andrade (2023) utiliza uma das formulações propostas por Dell'Amico *et al.* (2014) e, portanto, tem como função objetivo minimizar o custo total das rotas tomadas pelos veículos. Entretanto, a fim de adaptar o modelo para o SCB de

Lisboa, diversas alterações foram feitas na formulação final. O modelo proposto pelos autores leva em consideração uma frota heterogênea de veículos capacitados, incluindo veículos elétricos e sua autonomia (tempo de utilização do automóvel) e diferentes tipos de bicicletas a serem transportadas.

Um aspecto importante abordado pelos autores são as características de seleção de estações e preempção. A seleção de nós refere-se à possibilidade de relaxar a visita obrigatória de algumas estações, uma vez que algumas estações não necessitam de uma operação de balanceamento. A preempção, por outro lado, refere-se à possibilidade de um veículo deixar temporariamente bicicletas em uma estação intermediária para coleta futura. No trabalho proposto, os autores, por meio de restrições, permitem que estações sem demandas não sejam visitadas, mas não excluem a possibilidade de a estação ser visitada mesmo nessa condição.

Por fim, os autores utilizaram o método de B&C para resolver o modelo em conjunto com diversos procedimentos de aceleração. O algoritmo de B&C foi executado durante um período de 30 minutos, usando 5 instâncias reais do SCB de Lisboa. As rotas obtidas apresentaram uma redução significativa, variando de 30% a 75%, em comparação com as rotas existentes.

## 4.3 PBESCB com único veículo capacitado e minimização do tempo de desbalanceamento entre as estações

Na literatura, é possível encontrar diferentes funções objetivo para o PBESCB. Kadri *et al.* (2016) propõem um modelo cujo objetivo é minimizar o tempo em que as estações permanecem em um estado de desbalanceamento, e não necessariamente o custo da rota tomada pelo veículo.

Diferentemente das abordagens propostas por Dell'Amico *et al.* (2014) e Chastre e Andrade (2023), o modelo apresentado por Kadri *et al.* (2016) utiliza um único veículo capacitado para redistribuir as bicicletas entre as estações. Para a formulação matemática do problema, os autores introduzem um modelo de programação linear inteira e, para distinguir as estações em estado de desbalanceamento, é introduzida a noção de intervalo de confiança do número de bicicletas requerido por uma estação.

A resolução do problema é realizada por meio do método de *Branch-and-Bound* (B&B), em conjunto com diferentes limitantes inferiores e superiores. Entre os limitantes inferiores, há o uso dos métodos de relaxação lagrangiana e da regra de Smith *et al.* (1956) para certas restrições do modelo. A pesquisa apresenta três limitantes superiores que fazem o uso de

algoritmos genéticos, busca gulosa e busca do vizinho mais próximo.

## 4.4 Tabela de relacionamentos

A tabela 1 apresenta as principais diferenças entre os trabalhos citados anteriormente e compara-os com o trabalho proposto.

Tabela 1 – Relacionamento entre trabalhos

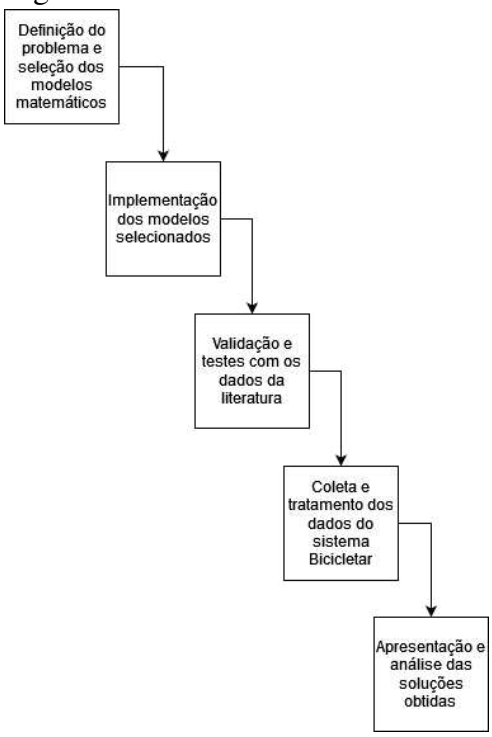
| **Características** | **(DELL'AMICO *et al.*, 2014)** | **(CHASTRE; ANDRADE, 2023)** | **(KADRI *et al.*, 2016)** | **Autor** |
|---|---|---|---|---|
| **Função Objetivo (min)** | Custo total das rotas | Custo total das rotas | Tempo em estado de desbalanceamento | Custo total das rotas |
| **Múltiplos veículos** | Sim | Sim | Não | Sim |
| **Frota homogênea** | Sim | Não | Único veículo | Sim |
| **Múltiplas bicicletas** | Não | Sim | Não | Não |

**Fonte:** Elaborado pelo autor.

# 5 METODOLOGIA

Este capítulo é responsável por propor as etapas metodológicas que serão utilizadas para a realização do trabalho. Abaixo há um diagrama que mostra, em detalhes, os passos a serem tomados.

Figura 1 – Etapas da metodologia




Fonte: elaborado pelo autor.

## 5.1 Definição do problema e seleção dos modelos matemáticos

Conforme definido na seção 3.3.1, o PBESCB envolve a otimização de rotas para veículos que partem de um depósito central, considerando a adição ou remoção de bicicletas entre as estações. O problema é estático, ocorrendo durante a noite com menor demanda, ao contrário da versão dinâmica que lida com mudanças rápidas durante o dia. Considerar as capacidades dos veículos e o retorno ao depósito central são fundamentais na resolução do problema.

Para este trabalho, será utilizada a definição formal proposta por Dell'Amico *et al.* (2014). O PBESCB é formalmente definido abaixo:

**Problema 5.1.0.1** *Problema de Balanceamento Estático em Sistemas de Compartilhamento de Bicicletas (PBESCB)*

***Entrada:*** *Uma tupla* $\vartheta = (G, m, Q, q, C)$*, onde:*

- *Digrafo* $G = (V, A)$*, em que:*
  - $V = \{0, 1, \ldots, n\}$ *onde, o vértice 0 representa o depósito central e os demais vértices representam as estações.*
  - $A = \{(i, j) \mid i, j = 0, \ldots, n, i \neq j\}$ *representam as ruas ou um caminho no mapa.*
- *m representa a quantidade de veículos capacitados.*
- *Q representa a capacidade de transporte de cada veículo.*
- $q = (q_1, q_2, \ldots, q_n)$ *representa as demandas de cada estação, onde o i-ésimo elemento corresponde à demanda da estação i.*
  - *Dado uma estação* $i \in V \setminus \{0\}$ *se* $q_i < 0$*, então i é uma estação de entrega e deve ser suprida com* $q_i$ *bicicletas.*
  - *Caso contrário, se* $q_i \geq 0$ *então, i é uma estação de coleta e* $q_i$ *bicicletas devem ser removidas.*
- $C = \left[c_{ij}\right]$ *representa a matriz de custos de percurso entre as estações, onde o elemento (i, j) representa o custo, em metros, de viajar da estação i para a estação j.*

***Questão:*** *Determinar m rotas para os veículos, visando minimizar o tamanho das rotas tomadas pelos veículos, garantindo as seguintes condições:*

1. *Cada veículo atende a uma rota que inicia e finaliza no depósito central.*
2. *Cada veículo inicia sua rota com carga vazia ou com uma carga inicial, variando de 0 a Q.*
3. *Cada estação* $i \in V \setminus \{0\}$ *é visitada exatamente uma vez por algum veículo, e a demanda da estação é completamente atendida pelo veículo que a visita.*
4. *A soma das demandas das estações visitadas mais a carga inicial nunca é negativa e nunca ultrapassa Q na rota realizada por um veículo.*

### *5.1.1 Seleção dos modelos matemáticos*

A primeira etapa deste trabalho consistiu em uma revisão da literatura sobre o problema de balanceamento em sistemas de compartilhamento de bicicletas, com o objetivo de identificar os estudos mais relevantes para a resolução do problema. Dentre os trabalhos analisados, as formulações propostas por Dell'Amico *et al.* (2014) destacaram-se por serem amplamente citadas e aplicadas em diversos contextos. Este artigo foi selecionado como base para a implementação dos modelos matemáticos, assim como para a aplicação do algoritmo de

*Branch-and-Cut*, heurísticas e procedimento de separação.

Dentre as formulações apresentadas no estudo, duas foram escolhidas para serem implementadas neste trabalho, conforme descrito a seguir.

#### *5.1.1.1 Formulação $F_1$*

A formulação $F_1$ utiliza como base o problema dos múltiplos caixeiros viajantes (mTSP), proposto por Bektas (2006). Nesse modelo, é utilizado uma variável binária $X_{ij}$ para representar a escolha de uma rota. A variável $X_{ij}$ assume valor 1 se o arco (i, j) é utilizado por algum veículo e 0 caso contrário. Ou seja, $X_{ij} = 1$ indica que o veículo se desloca da estação i para a estação j.

$$X_{ij} = \begin{cases} 1, & \text{se o arco } (i,j) \text{ é utilizado por um veículo,} \\ 0, & \text{caso contrário.} \end{cases}$$

Formalmente, o problema é modelado da seguinte maneira:

$$\text{Min} \quad \sum_{i \in V} \sum_{j \in V} C_{ij} X_{ij} \tag{5.1}$$

S.A:

$$\sum_{i \in V} X_{ij} = 1, \quad \forall j \in V \setminus \{0\} \tag{5.2}$$

$$\sum_{i \in V} X_{ji} = 1, \quad \forall j \in V \setminus \{0\} \tag{5.3}$$

$$\sum_{j \in V} X_{0j} \leq m \tag{5.4}$$

$$\sum_{j \in V \setminus \{0\}} X_{0j} = \sum_{j \in V \setminus \{0\}} X_{j0} \tag{5.5}$$

$$\sum_{i \in S} \sum_{j \in S} X_{ij} \leq |S| - 1, \quad S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{5.6}$$

$$X_{ij} \in \{0,1\}, \quad i,j \in V \tag{5.7}$$

A função objetivo (5.1) minimiza o custo total em metros do percurso. As restrições (5.2) e (5.3) impõem que cada estação, exceto o depósito, é visitada exatamente uma vez. A restrição (5.4) garante que no máximo $m$ veículos partem do depósito, enquanto a restrição (5.5) assegura que todos os veículos utilizados retornam ao depósito ao final de suas rotas. Por fim, as

restrições (5.6) são restrições de eliminação de subciclos utilizadas para garantir a conectividade da solução.

É importante ressaltar que as restrições de eliminação de subciclos podem crescer exponencialmente. Portanto, é necessário implementar um procedimento de separação que identifique quais restrições são violadas. Apenas as restrições violadas devem ser adicionadas ao modelo de maneira iterativa.

Para garantir a corretude da solução, é fundamental incluir restrições adicionais na formulação que considerem a capacidade dos veículos e a demanda das estações. Na formulação $F_1$, introduzimos uma variável extra $\theta_j$, que representa a carga do veículo após visitar a estação j, para $j \in V$. A carga deve ser atualizada ao longo da rota percorrida pelo veículo, de modo que, ao percorrer o arco $(i, j)$, a carga $\theta_j$ deve ser igual a $\theta_i + q_j$, onde $q_j$ representa a demanda da estação $j$. Inicialmente, podemos modelar as restrições adicionais citadas, onde:

$$\theta_j \geq (\theta_i + q_j)X_{ij}, \quad i \in V, j \in V \setminus \{0\} \tag{5.8}$$

$$\theta_i \geq (\theta_j - q_i)X_{ij}, \quad i \in V \setminus \{0\}, j \in V \tag{5.9}$$

$$\max\{0, q_j\} \leq \theta_j \leq \min\{Q, Q + q_j\}, \quad j \in V \tag{5.10}$$

As restrições (5.8) e (5.9) impõem que, se o arco $(i, j)$ é escolhido, então $\theta_j = \theta_i + q_j$, modelando assim a conservação do fluxo, independente do sinal de $q_j$. Além disso, a restrição (5.10) fornece limites superiores e inferiores para a carga.

Para serem incorporadas ao modelo, as restrições (5.8) e (5.9) precisam ser linearizadas. Podemos linearizá-las utilizando o método *big M*, resultando nas seguintes restrições finais:

$$\theta_j \geq \theta_i + q_j - M(1 - X_{ij}), \quad i \in V, j \in V \setminus \{0\} \tag{5.11}$$

$$\theta_i \geq \theta_j - q_i - M(1 - X_{ij}), \quad i \in V \setminus \{0\}, j \in V \tag{5.12}$$

Por fim, para finalizar a formulação, definimos o valor do *big M* como $M = \min\{Q, Q + q_j\}$ para restrição (5.11) e $M = \min\{Q, Q - q_j\}$ em (5.12).

#### *5.1.1.2 Formulação $F_2$*

A formulação $F_2$ também se baseia no problema dos múltiplos caixeiros viajantes. Entretanto, nesta formulação é introduzida uma variável adicional $f_{ij}$ que representa o fluxo

sobre um arco $(i, j)$. Esse fluxo indica a carga no veículo que percorre o arco $(i, j)$, caso o arco seja utilizado, para $(i, j) \in A$. Assim como na formulação $F_1$, é necessário adicionar restrições para considerar a capacidade dos veículos e o atendimento às demandas. Para isso, incluímos as seguintes restrições:

$$\sum_{i \in V} f_{ji} - \sum_{i \in V} f_{ij} = q_j, \quad j \in V \setminus \{0\} \tag{5.13}$$

$$\max\{0, q_i, -q_j\} X_{ij} \leq f_{ij} \leq \min\{Q, Q + q_i, Q - q_j\} X_{ij}, \quad (i, j) \in A \tag{5.14}$$

A restrição (5.13) garante o equilíbrio de fluxo nos arcos que entram e saem de um vértice. Isso significa que a quantidade de bicicletas que chegam a um ponto deve ser igual à quantidade que sai, ajustada pela demanda $q_j$ do vértice $j$.

A restrição (5.14) impõe limites superiores e inferiores aos fluxos em cada arco. Esses limites são ajustados conforme a necessidade de coletar ou entregar bicicletas. Para o limite inferior, por exemplo, se um arco $(i, j)$ é utilizado, então $f_{ij}$ deve ser maior que $q_i$, se $q_i > 0$ (visto que $q_i$ bicicletas acabaram de ser coletadas) ou maior que $-q_j$ se $q_j < 0$ (pois $q_j$ bicicletas precisam ser supridas para a próxima estação). Os limites superiores seguem a mesma lógica, com o fluxo $f_{ij}$ limitado pela capacidade do veículo $Q$ e ajustado conforme a demanda de cada estação.

#### *5.1.1.3 Restrição adicional para as formulações $F_1$ e $F_2$*

Podemos adicionar uma restrição mais forte às formulações $F_1$ e $F_2$ ao incluir uma restrição proposta por Hernández-Pérez e Salazar-González (2004), que garante que as soluções respeitem as demandas sem exceder a capacidade dos veículos. A restrição adicional é definida da seguinte maneira:

$$\sum_{i \in S} \sum_{j \in S} X_{ij} \leq |S| - \max\left\{1, \left\lceil \frac{|\sum_{i \in S} q_i|}{Q} \right\rceil \right\}, \quad S \subseteq V \setminus \{0\}, \quad S \neq \emptyset \tag{5.15}$$

Esta restrição é uma variação da clássica eliminação de subciclos (5.6), porém mais rigorosa. Para cada subconjunto $S$, o número de arcos que têm ambas as extremidades em $S$ não deve exceder a cardinalidade de $S$, menos o número mínimo de veículos necessários para atender às demandas de $S$. Esse número mínimo de veículos é obtido dividindo o valor absoluto da soma das demandas em $S$ pela capacidade $Q$ do veículo.

Note que, há a possibilidade de que a soma das demandas em $S$ seja zero. Nesse caso, o valor 1 é utilizado, pois ao menos um veículo será necessário para atender às demandas. Essa formulação adicional pode acelerar a convergência para a solução ótima. No entanto, assim como a restrição clássica de eliminação de subciclos, ela requer um procedimento de separação.

### *5.1.2 Síntese das Formulações e Restrições*

A tabela abaixo resume as principais variáveis e restrições envolvidas nas formulações $F_1$ e $F_2$. A formulação $F_1$ utiliza as variáveis de decisão $X_{ij}$ e $\theta_j$, com restrições relacionadas à visita das estações, consideração de carga dos veículos e atendimento às demandas, enquanto a formulação $F_2$ substitui $\theta_j$ por $f_{ij}$, incorporando ideias de fluxo. Ambas as formulações compartilham certas restrições, como a visitação de vértices, mas diferem na maneira como abordam as questões relacionadas à capacidade e atendimento de demandas.

Tabela 2 – Resumo das formulações, suas variáveis e restrições

| **Formulação** | **Variáveis** | **Restrições** |
|---|---|---|
| $F_1$ | $x_{ij}, \theta_j$ | (5.2) – (5.7), (5.10) – (5.11) e (5.15) |
| $F_2$ | $x_{ij}, f_{ij}$ | (5.2) – (5.7), (5.13), (5.14) e (5.15) |

Fonte: elaborado pelo autor.

## 5.2 Implementação dos modelos selecionados

Esta seção aborda a implementação dos modelos selecionados na seção anterior, detalhando o ambiente computacional e as ferramentas utilizadas. Abordaremos o algoritmo de B&C, além de discutir a desigualdade válida e o procedimento de separação.

### *5.2.1 Ambiente computacional*

Os modelos foram implementados em *Python* utilizando *Notebooks* no ambiente do *Visual Studio Code*[1]. Para a modelagem, foi utilizada a biblioteca *Python-MIP*[2], na versão 1.15.0. O solver utilizado para resolver os modelos através do algoritmo de B&C foi o *CBC Solver*[3]. Todos os códigos foram executados em um processador Intel® Core™ Ultra 7 155h, 1,40 GHz, e 32 GB de memória RAM.

[1] https://code.visualstudio.com/
[2] https://www.python-mip.com/
[3] https://www.coin-or.org/Cbc/cbcuserguide.html

### *5.2.2 Algoritmo de B&C*

Os modelos apresentados na seção anterior envolvem um número exponencial de restrições. Para resolvê-los, utilizamos o algoritmo B&C através do *CBC Solver*. A cada nó da árvore de busca, o solver resolve a relaxação linear do modelo. Quando uma solução fracionária é encontrada, o procedimento de separação desenvolvido pelo usuário é invocado para identificar possíveis cortes que serão adicionados ao modelo. Esse procedimento de separação é implementado por meio de *callbacks*, que são funções integradas ao algoritmo e responsáveis por realizar essas chamadas.

Para o nó raiz da árvore, utilizamos uma heurística gulosa para obter uma solução inicial factível. É comum em muitas aplicações do B&C incluir uma solução inicial, pois isso pode acelerar a convergência do algoritmo em encontrar uma solução ótima. A heurística escolhida foi a busca do vizinho mais próximo, porém com adaptações para considerar as características do problema, como a utilização de múltiplos veículos, as capacidades dos veículos e as demandas das estações. Para o desenvolvimento dessa heurística, é necessária a introdução de uma desigualdade válida, que será utilizada para identificar caminhos viáveis levando em conta a capacidade do veículo. A explicação detalhada tanto da desigualdade quanto da heurística será apresentada na próxima seção.

A estratégia de ramificação selecionada foi a estratégia padrão adotada pelo *CBC Solver*.

### *5.2.3 Desigualdades válidas*

Para introduzir a desigualdade válida, é necessário definir algumas notações. Seja $P$ um caminho que consiste em uma sequência de vértices, começando no depósito. Denotamos $|P|$ como o número de vértices em $P$, e por $P(i)$ o vértice na $i$-ésima posição do caminho $P$, para $i = 0, 1, \ldots, |P| - 1$. Por convenção, temos $P(0) = 0$, onde o vértice 0 representa o depósito.

Dado o caminho $P$, definimos dois operadores:

$$q_P^{\min} = \min_{i=1}^{|P|-1} \left\{ \sum_{j=1}^{i} q_{P(j)} \right\}$$
$$q_P^{\max} = \max_{i=1}^{|P|-1} \left\{ \sum_{j=1}^{i} q_{P(j)} \right\}$$

Esses operadores representam, respectivamente, a demanda acumulada mínima e máxima ao longo do caminho $P$. De acordo com Hernández-Pérez e Salazar-González (2004), um caminho $P$ é inviável em relação à capacidade dos veículos se

$$q_P^{\max} - q_P^{\min} > Q$$

onde $Q$ é a capacidade máxima do veículo. Se o veículo parte do depósito sem nenhuma carga, então ao longo do caminho o veículo atingirá a carga máxima igual a $q_P^{\max}$, que não pode exceder a capacidade $Q$. Note também que $q_P^{\min}$ pode assumir valores negativos, mesmo em uma solução viável. Nesse caso, o veículo deve partir do depósito contendo $-q_P^{\min}$ bicicletas. Assim, a carga máxima ao longo do caminho se torna $q_P^{\max} - q_P^{\min}$, que também não pode ultrapassar $Q$.

#### *5.2.3.1 Exemplo Computacional*

Para ilustrar o cálculo de $q_P^{\min}$ e $q_P^{\max}$, consideremos o caminho $P = \{0,4,5,7,6\}$, com demandas $q = \{0,1,-1,2,3\}$ e a capacidade do veículo $Q = 10$. Lembrando que $q_0 = 0$ (demanda no depósito). Vamos calcular os valores acumulados $\left\{\sum_{j=1}^{i} q_{P(j)}\right\}$ para $i = 1,2,3,4$.

– Para $i = 1$, temos:

$$\sum_{j=1}^{1} q_{P(j)} = q_4 = 1$$

– Para $i = 2$, temos:

$$\sum_{j=1}^{2} q_{P(j)} = q_4 + q_5 = 1 + (-1) = 0$$

– Para $i = 3$, temos:

$$\sum_{j=1}^{3} q_{P(j)} = q_4 + q_5 + q_7 = 1 + (-1) + 2 = 2$$

– Para $i = 4$, temos:

$$\sum_{j=1}^{4} q_{P(j)} = q_4 + q_5 + q_7 + q_6 = 1 + (-1) + 2 + 3 = 5$$

Portanto, o conjunto de somas acumuladas é $\{1,0,2,5\}$. Logo, podemos calcular:

$$q_P^{\min} = \min\{1,0,2,5\} = 0$$
$$q_P^{\max} = \max\{1,0,2,5\} = 5$$

Assim, concluímos que para o caminho $P$, temos $q_P^{\min} = 0$ e $q_P^{\max} = 5$. Fazendo $q_P^{\max} - q_P^{\min} > Q$ percebemos que o caminho $P$ é viável com respeito à capacidade do veículo.

### *5.2.4 Busca do vizinho mais próximo*

Nesta seção, descrevemos a heurística gulosa que utiliza o princípio do vizinho mais próximo para criar uma solução inicial no processo de busca. O algoritmo constrói rotas a partir do depósito, sempre estendendo o caminho para o vizinho mais próximo que ainda não foi atendido, respeitando as restrições de capacidade do veículo e as demandas das estações, conforme discutido na seção anterior.

Para garantir a viabilidade das rotas geradas, a heurística utiliza dois procedimentos auxiliares: um para calcular as demandas acumuladas ao longo de um caminho e outro para identificar o vizinho mais próximo ainda não atendido. Esses procedimentos são apresentados nas subseções a seguir e são fundamentais para o funcionamento da heurística completa, descrita ao final desta seção.

#### *5.2.4.1 Função auxiliar: cálculo da demanda cumulativa*

Este algoritmo é responsável por calcular as demandas cumulativas mínima e máxima ao longo de uma rota parcial. Essa verificação é essencial para assegurar que a extensão da rota continue viável em relação à capacidade do veículo.

**Algoritmo 1:** Algoritmo para calcular os operadores $q_P^{\max}$ e $q_P^{\min}$

**Input:** Caminho: $P$, Demandas: $q$

**Output:** Demanda Cumulativa Mínima $q_{\min}$, Máxima $q_{\max}$

**Function** `calculate_cumulative_demand(`*P, q*`)`**:**

- $cumulative \leftarrow 0$;
- $q_{\min}, q_{\max} \leftarrow \infty, -\infty$;
- **for** $i \leftarrow 1$ ***to*** $|P| - 1$ **do**
  - $vertex \leftarrow P[i]$;
  - $cumulative \leftarrow cumulative + q[vertex]$;
  - $q_{\min} \leftarrow \min(q_{\min}, cumulative)$;
  - $q_{\max} \leftarrow \max(q_{\max}, cumulative)$;
- **end**
- **return** $q_{\min}, q_{\max}$;

#### 5.2.4.2 *Função auxiliar: seleção do vizinho mais próximo*

Este procedimento identifica, com base na matriz de distâncias, o cliente mais próximo da posição atual do veículo que ainda não foi visitado. Ele representa o passo de decisão gulosa que guia a construção da rota.

**Algoritmo 2:** Algoritmo para encontrar o vizinho não atendido mais próximo

**Input:** Posição atual: $i$, Lista de vizinhos não atendidos: $V'$, Matriz de distâncias: $C$

**Output:** Vizinho mais próximo

**Function** `find_closest_unserved_customer(`$i$, $V'$, $C$`):`

```
min_distance ← ∞;
closest ← −1;
foreach v ∈ V′ do
    if C[i][v] < min_distance then
        min_distance ← C[i][v];
        closest ← v;
    end
end
return closest;
```

#### 5.2.4.3 *Algoritmo completo: Heurística do Vizinho Mais Próximo*

A seguir, descrevemos a heurística completa. O algoritmo começa sempre no depósito, visitando o vizinho mais próximo que ainda não foi visitado. A extensão do caminho para um novo vizinho é permitida somente quando a inclusão desse novo vizinho torna o caminho factível em relação às capacidades do veículo e às demandas das estações (conforme discutido na seção anterior). Se nenhum vizinho factível puder ser visitado, a rota retorna ao depósito, e uma nova rota é iniciada de forma iterativa, repetindo o processo até que todos os vértices tenham sido visitados.

**Algoritmo 3:** Heurística do Vizinho Mais Próximo

**Input:** Depósito: 0, Lista de Clientes: $V$, Demandas: $q$, Matriz de Distâncias: $C$, Capacidade: $Q$

**Output:** Rotas *routes*

**Function** `closest_neighbor(0, V, q, C, Q):`

```
routes ← [];
while V ≠ ∅ do
    route ← [0];
    current ← 0;
    while True do
        next_customer ← find_closest_unserved_customer(current, V, C);
        if next_customer == −1 then
            break;
        end
        extended_route ← route;
        Adicione next_customer em extended_route;
        q_min, q_max ← calculate_cumulative_demand(extended_route, q);
        if q_max − q_min > Q then
            break ;
        end
        Adicione next_customer em route;
        Remova next_customer em V;
        current ← next_customer;
    end
    Adcione 0 em route;
    Adcione route em routes;
end
return routes;
```

Ao analisar a complexidade da heurística do vizinho mais próximo, observa-se que, no pior caso, cada cliente é examinado diversas vezes ao longo da construção das rotas. A cada passo, é necessário identificar o cliente não atendido mais próximo e verificar a factibilidade da extensão da rota com base na demanda acumulada. Ambas as operações têm custo proporcional

ao número de clientes restantes. Dessa forma, a complexidade total da heurística está limitada a uma ordem quadrática em relação ao número de clientes, ou seja, $O(n^2)$.

### *5.2.5 Procedimento de separação*

Para finalizar, apresentamos o procedimento de separação utilizado para identificar as restrições violadas, conforme discutido anteriormente (5.1.1.3). Esse procedimento atua sobre a variável de decisão $\bar{X}_{ij}$ (utilizamos essa notação para indicar que a variável pode assumir valores fracionários), a qual pode resultar em soluções fracionárias devido à relaxação no algoritmo de B&C.

Para separar as restrições exponenciais de eliminação de subciclos (5.6), construímos um grafo $\bar{G} = (\bar{V}, \bar{A})$, onde $\bar{V} = V$ e $\bar{A} = (i, j) \in A : \bar{X}_{ij} > 0$. Para cada arco $(i, j) \in \bar{A}$, associamos o valor de $\bar{X}ij$ como sua capacidade. Em seguida, calculamos o fluxo máximo no grafo $\bar{G}$, utilizando o depósito como origem e qualquer vértice $i$ como destino. Se o valor do fluxo máximo for menor que um, isso indica que $i$ está desconectado do depósito. Nesse caso, avaliamos as restrições violadas em (5.6) por meio do conjunto $S$ induzido pelo corte mínimo. O mesmo processo é aplicado para as restrições mais fortes (5.15), e então as restrições violadas são adicionadas ao modelo.

Abaixo há a formalização do algoritmo de separação de restrições:

**Algoritmo 4:** Procedimento de Separação

**Input:** Conjunto de vértices: $V$, Arcos: $A$, Variáveis de decisão: $X$, Demandas: $q$, Capacidade do veículo: $Q$

**Output:** Cortes adicionados ao modelo

**Function** `SeparationProcedure(`$V$, $A$, $X$, $q$, $Q$`)`**:**

- $cut\_pool \leftarrow \{\}$;
- $depot \leftarrow 0$;
- $\bar{G} \leftarrow$ Novo grafo direcionado;
- Adicionar nós em $\bar{G}$ a partir de $V$;
- Adicionar arcos $(i, j)$ com capacidade $\bar{X}_{ij}$ em $\bar{G}$ se $\bar{X}_{ij} > 0$;
- **foreach** $i \in V \setminus \{depot\}$ **do**
  - $flow\_value \leftarrow$ fluxo máximo de $\bar{G}$ entre $depot$ e $i$;
  - **if** $flow\_value < 1$ **then**
    - Obter o conjunto de corte $S$ a partir do $min\_cut$ em $\bar{G}$ utilizando o $depot$ e $i$;
    - **if** $S \neq \emptyset$ **then**
      - $tour \leftarrow \sum_{i \in S} \sum_{j \in S} \bar{X}_{ij}$;
      - **if** $tour \geq |S| - 1$ **then**
        - Adicionar corte fraco $\sum_{i \in S} \sum_{j \in S} X_{ij} \leq |S| - 1$ ao pool de cortes;
      - **end**
      - $demand \leftarrow \sum_{i \in S} q[i]$;
      - $min\_vehicles \leftarrow \lceil demand/Q \rceil$;
      - **if** $tour \geq |S| - \max\{1, min\_vehicles\}$ **then**
        - Adicionar corte forte $\sum_{i \in S} \sum_{j \in S} X_{ij} \leq |S| - \max\{1, min\_vehicles\}$ ao pool de cortes;
      - **end**
    - **end**
  - **end**
- **end**
- Adicionar todos os cortes do $cut\_pool$ ao modelo;

## 5.3 Validação e testes com os dados da literatura

Para validar a implementação dos modelos, do algoritmo de B&C e do procedimento de separação apresentados nas seções anteriores, bem como para avaliar a qualidade das soluções encontradas, esta seção compara os resultados obtidos com os do trabalho de Dell'Amico *et al.* (2014), utilizando as mesmas instâncias de entrada. A validação foi realizada aplicando o algoritmo de B&C em conjunto com as formulações propostas em diversas instâncias de SCB ao redor do mundo.

As instâncias foram coletadas por meio dos sites de várias empresas que prestam o serviço de SCB em diferentes países, e estão divididas da seguinte forma:

- Itália: Bari, Brescia, Bergamo, La Spezia, Parma, Roma, Torino, Treviso, e Reggio Emilia.
- Irlanda: Dublim.
- Estados Unidos: Boston, Denver, Madison, Miami, Minneapolis, e San Antonio.
- Canadá: Ottawa e Toronto.
- México: Cidade do México e Guadalajara.
- Argentina: Buenos Aires.
- Brasil: Rio de Janeiro.

Para calcular a matriz de distâncias $C_{ij}$, os pesquisadores coletaram as coordenadas de cada estação e do depósito. Em seguida, utilizaram a api do *Google Maps* para calcular a menor distância (em metros) entre as estações e o depósito. Caso a localização do depósito não estivesse disponível, os pesquisadores assumiram que ele se encontrava no mesmo local de uma das estações.

Todos os dados podem ser acessados na página[4] do grupo de pesquisa Regor da *University of Modena and Reggio Emilia*. A implementação do trabalho de Dell'Amico *et al.* (2014) foi realizada em C/C++, utilizando o *solver CPLEX 12.2*. As configurações da máquina em que os modelos foram executados são: processador Intel Core i3-2100 de 3,10 GHz e 4,00 GB de RAM.

A tabela 3 apresenta as cidades utilizadas na comparação, bem como informações adicionais, como, por exemplo, o número de estações, a demanda média e a distância média entre as estações.

Os resultados obtidos para as instâncias selecionadas são apresentados na tabela 4.

[4] http://www.or.unimore.it/site/home/online-resources/bike-sharing-rebalancing-problems/articolo1090035457.html

Tabela 3 – Instâncias escolhidas para comparação entre resultados.

| Cidade | País | $\|V\|$ | $\min\{q_i\}$ | $\mu\{q_i\}$ | $\max\{q_i\}$ | $\sigma\{q_i\}$ | $\min\{c_{ij}\}$ | $\mu\{c_{ij}\}$ | $\max\{c_{ij}\}$ | $\sigma\{c_{ij}\}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Bari | Itália | 13 | $-5$ | $-1,54$ | 5 | 2,70 | 400 | 2283,97 | 5400 | 1067,62 |
| Parma | Itália | 15 | $-6$ | $-1,07$ | 4 | 2,76 | 200 | 3121,43 | 8800 | 1857,86 |
| San Antonio | EUA | 23 | $-4$ | 1,74 | 8 | 3,43 | 98 | 1950,98 | 4808 | 944,61 |
| Guadalajara | México | 41 | $-11$ | $-1.07$ | 1 | 1,94 | 60 | 3278,30 | 14728 | 2440,40 |
| Denver | EUA | 51 | $-8$ | $-0,69$ | 7 | 3,28 | 211 | 3873,94 | 12000 | 2643,54 |

Fonte: elaborado pelo autor.

Nessa tabela, "Cidade $(|V|, Q)$"indica a instância associada a cada cidade, onde $|V|$ representa a quantidade de estações ou vértices e $Q$ a capacidade dos veículos. O valor de $UB$ corresponde à melhor solução factível encontrada pelo algoritmo de B&C, utilizando as formulações $F_1$ e $F_2$. Por fim, o tempo indica o número de segundos que cada modelo levou para encontrar uma solução.

Tabela 4 – Solução para as instâncias escolhidas.

| Cidade $(\|V\|, Q)$ | $UB$ | Tempo $(F_1)$ | Tempo $(F_2)$ |
|---|---|---|---|
| Bari (13,30) | 14600 | 0,06 | 0,11 |
| Parma (15, 30) | 29000 | 0,05 | 0,03 |
| San Antonio (23,30) | 22982 | 0,19 | 2,74 |
| Guadalajara (41,30) | 57476 | 1,16 | 2,97 |
| Denver (51,30) | 51583 | 66,57 | 11,01 |

Fonte: elaborado pelo autor.

O algoritmo de B&C foi executado em um limite máximo de 3600 segundos para encontrar as soluções.

### *5.3.1 Comparação dos resultados obtidos*

Por fim, a tabela abaixo apresenta os resultados obtidos das instâncias de entrada da tabela 3 da aplicação deste trabalho.

O algoritmo de B&C foi executado com um limite máximo de tempo de 720 segundos. Os resultados obtidos pela formulação $F_1$ mostram que o algoritmo encontrou as mesmas soluções nas instâncias de Bari, Parma e San Antonio, variando apenas o tempo necessário para encontrar a solução. Nas instâncias de Guadalajara e Denver, embora o algoritmo tenha utilizado todo o tempo máximo, as soluções obtidas foram próximas às encontradas no artigo base.

Já a formulação $F_2$ apresentou um comportamento similar, encontrando os mesmos valores de $UB$ para as instâncias de Bari, Parma e San Antonio, mas com tempos de execução

Tabela 5 – Comparação dos resultados das formulações $F_1$ e $F_2$

| Cidade $(\|V\|, Q)$ | Formulação $F_1$ | | Formulação $F_2$ | |
|---|---|---|---|---|
| | $UB$ | Tempo (s) | $UB$ | Tempo (s) |
| Bari (13,30) | 14600 | 3,0 | 14600 | 0,10 |
| Parma (15,30) | 29000 | 7,8 | 29000 | 0,05 |
| San Antonio (23,30) | 22982 | 15,2 | 22982 | 17,0 |
| Guadalajara (41,30) | 58674 | 720 | 57398 | 252 |
| Denver (51,30) | 52448 | 720 | 56177 | 6,1 |

Fonte: elaborado pelo autor.

consideravelmente menores. Para a instância de Guadalajara, o algoritmo conseguiu uma solução superior à da formulação $F_1$, utilizando 252 segundos. Na instância de Denver, a solução foi pior, mas próxima da solução encontrada no artigo base, com um tempo menor.

Esses resultados indicam que a formulação $F_2$ tende a ser mais eficiente em termos de tempo de execução, especialmente em instâncias maiores, sem comprometer significativamente a qualidade das soluções.

## 5.4 Coleta e tratamento dos dados do sistema Bicicletar

Nesta seção, detalhamos o processo de coleta e tratamento dos dados do sistema Bicicletar. O Bicicletar é o serviço público de compartilhamento de bicicletas (SCB) da cidade de Fortaleza. O sistema é fruto de uma parceria público-privada entre a Prefeitura de Fortaleza e a empresa Unimed, sendo operado pela Serttel, uma companhia especializada em soluções de mobilidade urbana.

Atualmente, o SCB de Fortaleza possui mais de 200 estações distribuídas por toda a capital e opera diariamente entre 5h às 23h59. Para utilizar o serviço, é necessário realizar um cadastro na plataforma mobile. O sistema oferece planos anuais, mensais e diários. Além disso, estudantes e usuários do Bilhete Único podem utilizar o serviço gratuitamente. Os usuários podem alugar bicicletas quantas vezes quiserem, com um limite de 60 minutos por uso. No entanto, após a devolução, é necessário aguardar 15 minutos para realizar um novo aluguel.

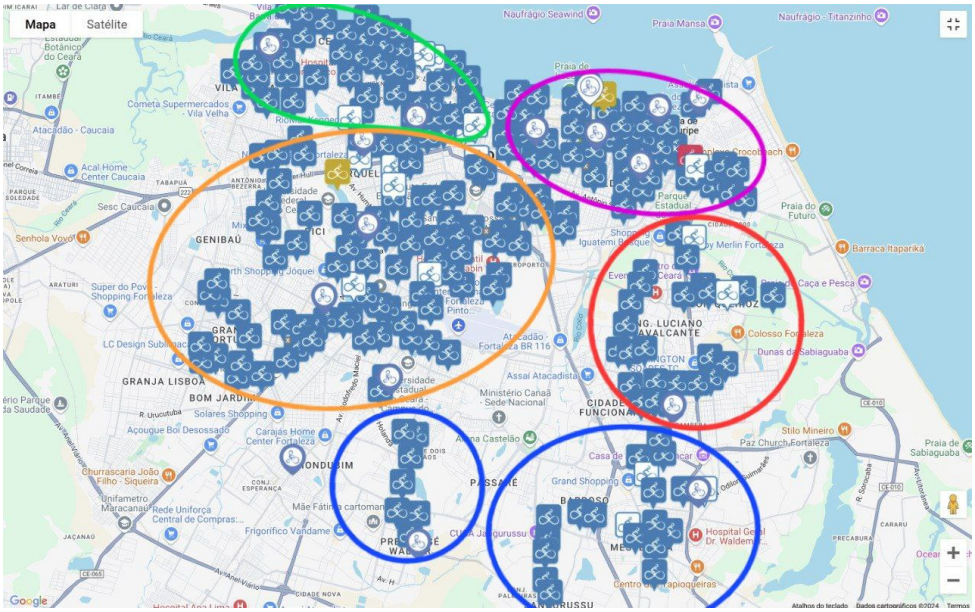
### *5.4.1 Coleta dos dados*

Para a coleta de dados foi retirada uma "foto"da situação, o SCB de Fortaleza disponibiliza informações como a quantidade de estações, suas capacidades, localização geográfica

(coordenadas) e suas disponibilidades na página oficial [5] do serviço. Devido ao grande número de estações, o primeiro passo foi agrupar todo o sistema em 5 *clusters*, facilitando a análise e organização dos dados. A divisão dos clusters pode ser visualizada na Figura 2.

Para cada *cluster*, foi gerado um arquivo *Comma-separated values* (CSV), organizando as informações em tuplas que contêm os seguintes dados: número da estação, endereço, latitude, longitude, quantidade de bicicletas disponíveis e espaços livres.

Figura 2 – Mapa das estações do Bicicletar e *clusters* criados




**Fonte:** Elaborado pelo autor.

As informações sobre a localização do depósito central, a quantidade de veículos responsáveis pelo reabastecimento e suas capacidades não estão disponíveis no site oficial e foram obtidas por meio de contato com a Autarquia Municipal de Trânsito e Cidadania (AMC), com a mediação do setor de mobilidade urbana da Secretaria Municipal da Conservação e Serviços Públicos (SCSP) de Fortaleza. Esses órgãos estão localizados, respectivamente, na Av. Desembargador Gonzaga, 1630, no bairro Cidade dos Funcionários, e no Edifício Magna, na Av. Pontes Vieira, no bairro Aldeota. Atualmente, o sistema conta com 12 veículos, cada um com capacidade para transportar até 16 bicicletas. O uso dos veículos é dividido por turnos, sendo 6 para o período diurno e 6 para o período noturno. O depósito central (estação 0) está situado no bairro Montese.

[5] https://www.bicicletar.com.br/mapaestacao.aspx

### *5.4.2 Criação das instâncias*

Para finalizar a criação das instâncias com as informações já coletadas, a próxima etapa envolveu a construção das matrizes de distância para todos os *clusters* e o cálculo das demandas de cada estação.

Para criar as matrizes, foi utilizada a *API* do *Google Maps* por meio de um *script* em *Python*. Nessa etapa, as coordenadas de latitude e longitude foram fundamentais. É importante ressaltar que o problema em questão é assimétrico, e esse aspecto foi considerado durante a construção da matriz de distância.

No cálculo das demandas, utilizamos a mesma estratégia proposta por Dell'Amico *et al.* (2014), com o objetivo de balancear as estações e atingir uma configuração final desejada. Os pesquisadores seguiram uma prática comum em diversos SCB, que consiste em estabelecer a configuração final como metade da capacidade total ocupada. A fórmula utilizada para determinar a demanda de cada estação é a seguinte:

$$q_i = \text{quantidade de bicicletas disponíveis} - \text{configuração final}$$

onde a configuração final é definida como:

$$\text{configuração final} = \frac{\text{bicicletas disponíveis} + \text{vagas disponíveis}}{2}$$

É importante destacar que a demanda do depósito central é sempre zero. Após o cálculo da matriz de distância e das demandas para cada *cluster*, o passo final consistiu em criar um arquivo *JavaScript Object Notation* (JSON) para cada *cluster*, facilitando a consulta e a organização dos dados. Cada arquivo JSON segue a estrutura: número de vértices, vetor de demandas, capacidade do veículo e matriz de distância. As instâncias podem ser visualizadas na tabela 6.

Tabela 6 – Instâncias geradas a partir dos dados do Bicicletar

| Instância | Cluster | $\|V\|$ | $\min\{q_i\}$ | $\mu\{q_i\}$ | $\max\{q_i\}$ | $\sigma\{q_i\}$ | $\min\{c_{ij}\}$ | $\mu\{c_{ij}\}$ | $\max\{c_{ij}\}$ | $\sigma\{c_{ij}\}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Inferior | Azul | 23 | $-12$ | $-1,43$ | 4 | 3,42 | 108 | 6263,92 | 17312 | 3778,36 |
| Inferior | Vermelho | 26 | $-8$ | $-2,88$ | 4 | 3,11 | 294 | 4266,40 | 1424 | 2656,30 |
| Superior | Roxo | 53 | $-9$ | $-0.94$ | 7 | 3,77 | 290 | 3813,93 | 15792 | 2248,84 |
| Superior | Verde | 49 | $-8$ | 2.30 | 2 | 2,50 | 4 | 3863,48 | 11503 | 2068,32 |
| Central | Laranja | 91 | 10 | 2,38 | 6 | 3,07 | 170 | 5675,91 | 18693 | 2744,95 |

Fonte: elaborado pelo autor.

## 5.5 Apresentação e análise das soluções obtidas

Após a criação das instâncias, foram obtidas as soluções geradas pelas formulações $F_1$ e $F_2$. Nesta etapa, o foco foi analisar as soluções obtidas para extrair *insights* sobre as rotas e realizar uma comparação entre as duas formulações. No capítulo 6, os resultados serão apresentados e descritos detalhadamente.

# 6 RESULTADOS

Neste capítulo, são apresentados os resultados da execução dos modelos $F_1$ e $F_2$ sobre as instâncias geradas a partir dos dados do sistema Bicicletar, sintetizadas na tabela 6. Inicialmente, são descritos os resultados obtidos e, em seguida, é realizada uma análise detalhada das rotas obtidas por cada formulação.

## 6.1 Resultados obtidos

Para avaliar o desempenho das formulações, todas as instâncias foram executadas em um limite de tempo de 1800 segundos. As tabelas 7 e 8 apresentam os resultados obtidos, contendo informações como capacidade do veículo $Q$, o melhor valor da solução viável encontrada $UB$, quantidade de cortes, o número de nós explorados na árvore de busca e o tempo total de execução.

Tabela 7 – Soluções encontradas para a formulação $F_1$

| Instância - Cluster | $\|V\|$ | $Q$ | $UB$ | Cortes | Nós | Tempo |
|---|---|---|---|---|---|---|
| Inferior - Azul | 23 | 30 | 61807 | 457 | 54576 | 1800 |
| Inferior - Vermelho | 26 | 30 | 89932 | 1045 | 32680 | 1800 |
| Superior - Roxo | 53 | 30 | 88600 | 174 | 5329 | 1800 |
| Superior - Verde | 49 | 30 | 104339 | 262 | 4707 | 1800 |
| Central - Laranja | 91 | 50 | 153881 | 680 | 997 | 1800 |

Fonte: elaborado pelo autor.

Tabela 8 – Soluções encontradas para a formulação $F_2$

| Instância - Cluster | $\|V\|$ | $Q$ | $UB$ | Cortes | Nós | Tempo |
|---|---|---|---|---|---|---|
| Inferior - Azul | 23 | 30 | 56852 | 1775 | 11242 | 62,79 |
| Inferior - Vermelho | 26 | 30 | 87895 | 346 | 7151 | 577,95 |
| Superior - Roxo | 53 | 30 | 74808 | 672 | 6136 | 1800 |
| Superior - Verde | 49 | 30 | 92536 | 305 | 5984 | 1800 |
| Central - Laranja | 91 | 50 | 143529 | 1993 | 476 | 1800 |

Fonte: elaborado pelo autor.

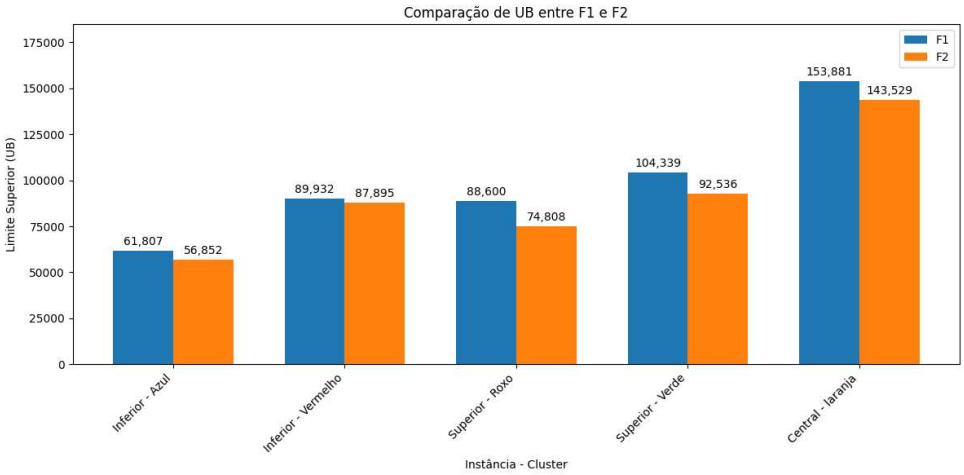
As instâncias Inferior Azul, Inferior Vermelho, Superior Roxo utilizaram 3 veículos como valor de $m$ na entrada do algoritmo, enquanto as instâncias Superior Verde e Central Laranja utilizaram 4 e 5 veículos, respectivamente. A seguir, apresentamos os principais resultados para cada formulação.

## 6.2 Análise Comparativa das Formulações

Para a formulação $F_1$, o algoritmo não encontrou nenhuma solução ótima em nenhuma das instâncias testadas, gerando apenas soluções viáveis. Em contraste, a formulação $F_2$ encontrou soluções ótimas para as instâncias Inferior Azul e Inferior Vermelho, em um tempo consideravelmente menor.

A comparação entre as soluções encontradas é ilustrada na figura 3. Podemos observar que a formulação $F_2$ encontrou soluções melhores em todas as instâncias testadas, especialmente com um número de vértices grande. Esses resultados indicam que, no geral, a formulação $F_2$ desempenha melhor que a primeira formulação.

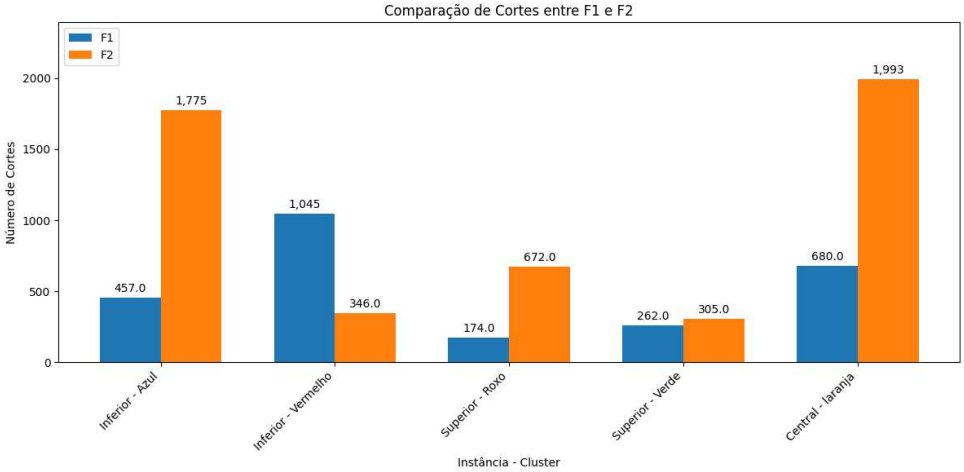
Figura 3 – Comparação entre $UB$ encontradas por ambas formulações




**Fonte:** Elaborado pelo autor.

A figura 4 mostra a quantidade de cortes adicionados por ambas as formulações. Em média, a formulação $F_2$ adicionou 1018,2 cortes, enquanto a formulação $F_1$ adicionou apenas 523,6 cortes, em média.

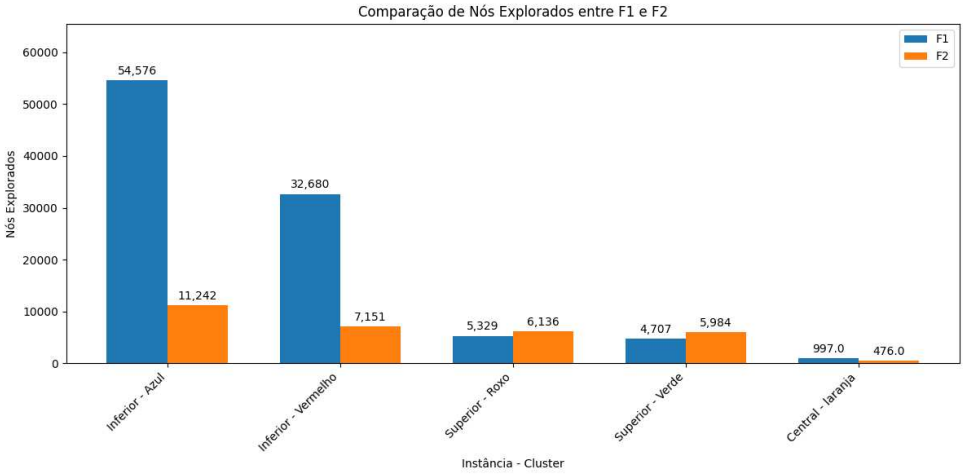
Figura 4 – Comparação entre cortes encontradas por ambas formulações




**Fonte:** Elaborado pelo autor.

Quanto ao número de nós explorados, a figura 5 mostra que a formulação $F_1$ explora mais nós em comparação à formulação $F_2$. Isso pode indicar que a formulação $F_2$ aproxima melhor as soluções encontradas da solução ótima.

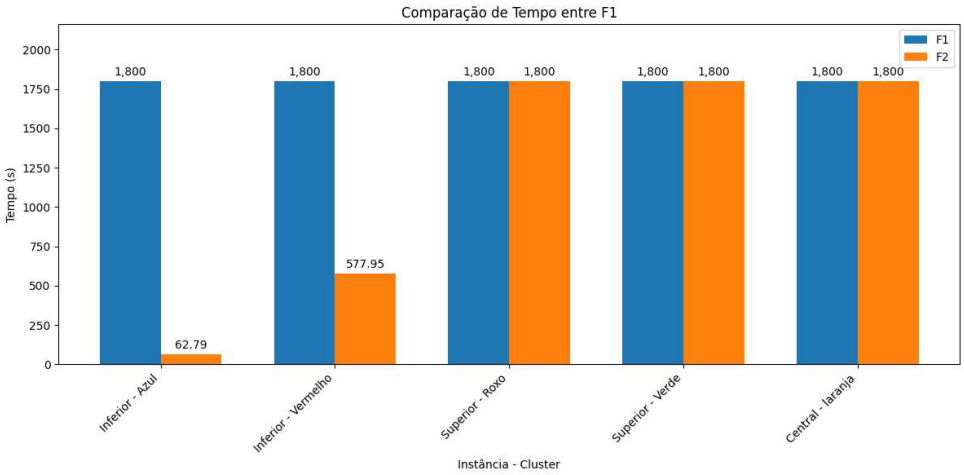
Figura 5 – Comparação entre nós explorados por ambas formulações




**Fonte:** Elaborado pelo autor.

Por fim, analisando o tempo levado para encontrar soluções em ambas as formulações, a figura 6 mostra que a formulação $F_2$ levou menos tempo para encontrar soluções ótimas para instâncias pequenas e, mesmo levando o mesmo tempo que a formulação $F_1$ para instâncias grandes, vimos que a segunda formulação encontrou soluções melhores.

Figura 6 – Tempo levado para encontrar soluções em ambas formulações



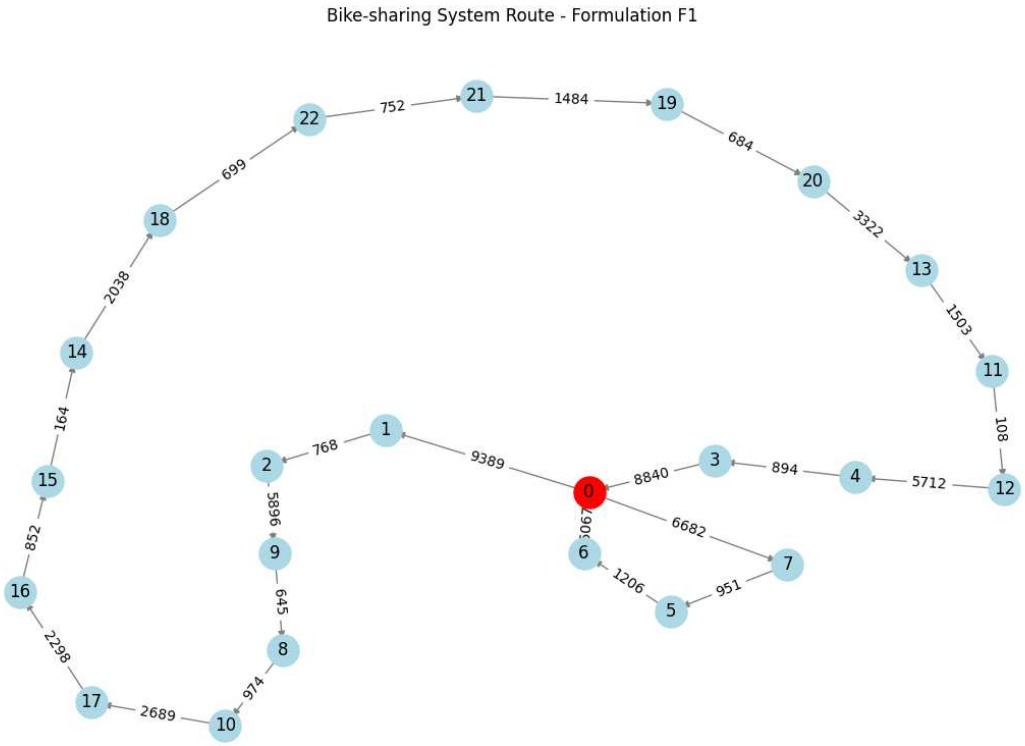

**Fonte:** Elaborado pelo autor.

Concluímos que a formulação $F_2$ é superior em termos de tempo de execução e qualidade de soluções encontradas, mesmo em instâncias com um número maior de vértices. Essa formulação não só encontra soluções melhores, como também as encontra de forma mais eficiente.

## 6.3 Exemplo de soluções encontradas

A figura 7 apresenta a solução encontrada para a instância Inferior Azul para a formulação $F_1$.

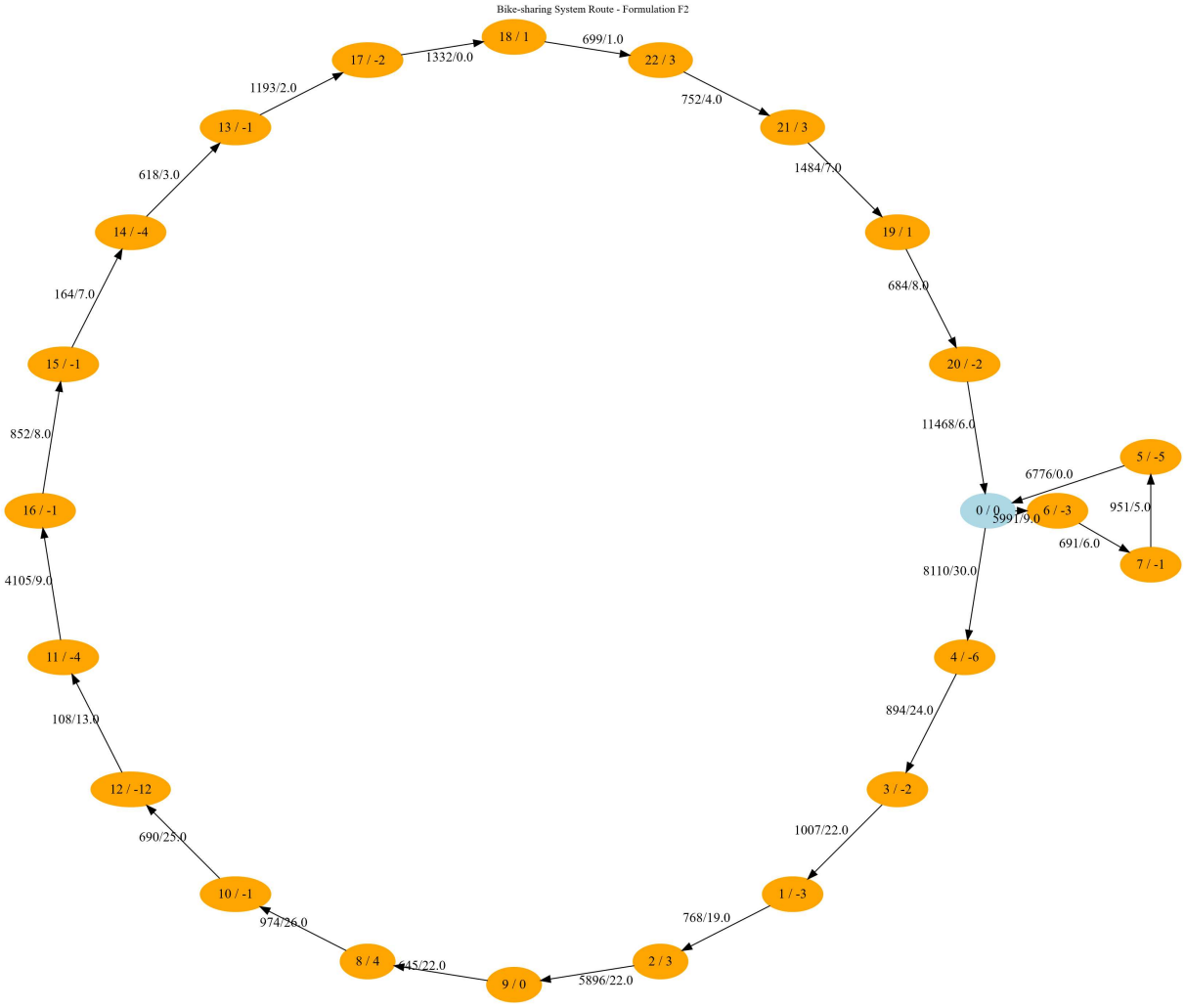
Figura 7 – Solução da instância inferior azul para a formulação $F_1$




**Fonte:** Elaborado pelo autor.

A figura 8, por sua vez, apresenta a solução para a segunda formulação utilizando a mesma instância. Na representação da solução, cada aresta contém informações sobre o custo do percurso e o valor da variável de fluxo $f_{ij}$ para todos os arcos escolhidos na solução. Além disso, os vértices mostram tanto qual é a estação e sua demanda associada.

Figura 8 – Solução da instância inferior azul para a formulação $F_2$




**Fonte:** Elaborado pelo autor.

## 6.4 Limitações das formulações

As soluções encontradas, sintetizadas nas tabelas 7 e 8, utilizaram veículos com capacidade total de 30 ou 40 bicicletas. Entretanto, como dito anteriormente em 5.4.1, o Bicicletar possui 12 veículos com 16 de carga total, divididos em 6 veículos para o período diurno e noturno.

As soluções apresentadas nas tabelas 7 e 8 consideraram, para a entrada do algoritmo, veículos com capacidade total de 30 ou 40 bicicletas. No entanto, conforme descrito anteriormente em 5.4.1, o sistema Bicicletar atende às estações com 12 veículos, cada um com capacidade de 16 bicicletas, divididos em 6 veículos para o período diurno e 6 para o período noturno. Nas tabelas 10 e 9 apresentamos as soluções utilizando veículos com 16 de capacidade total e discutimos suas limitações.

Para determinar a quantidade de veículos em cada instância, optamos por escolher o menor número de veículos disponíveis no período noturno, de maneira a encontrar uma solução viável para ambas formulações. A quantidade de veículos escolhida para cada instância foi: 3 veículos para a instância Inferior Azul, 5 para Inferior Vermelho e 4 para Superior Roxo. Para as instâncias Superior Verde e Central Laranja, o algoritmo não encontrou nenhuma solução, mesmo utilizando o máximo de veículos do período noturno.

Tabela 9 – Soluções encontradas para a formulação $F_2$ utilizando veículos com 16 de carga total

| Instância - Cluster | $\|V\|$ | $Q$ | $UB$ | Cortes | Nós | Tempo |
|---|---|---|---|---|---|---|
| Inferior - Azul | 23 | 16 | 76273 | 2846 | 199 | 22,16 |
| Inferior - Vermelho | 26 | 16 | 135786 | 447 | 27243 | 1800 |
| Superior - Roxo | 53 | 16 | 109090 | 713 | 2894 | 1800 |
| Superior - Verde | 49 | 16 | — | — | — | — |
| Central - Laranja | 91 | 16 | — | — | — | — |

Fonte: elaborado pelo autor.

Tabela 10 – Soluções encontradas para a formulação $F_1$ utilizando veículos com 16 de carga total

| Instância - Cluster | $\|V\|$ | $Q$ | $UB$ | Cortes | Nós | Tempo |
|---|---|---|---|---|---|---|
| Inferior - Azul | 23 | 16 | 80822 | 3060 | 29490 | 1800 |
| Inferior - Vermelho | 26 | 16 | 154185 | 1604 | 34786 | 1800 |
| Superior - Roxo | 53 | 16 | 149182 | 1153 | 2827 | 1800 |
| Superior - Verde | 49 | 16 | — | — | — | — |
| Central - Laranja | 91 | 16 | — | — | — | — |

Fonte: elaborado pelo autor.

De modo geral, a qualidade das soluções foi pior para ambas as formulações quando comparada às soluções obtidas com veículos de 30 ou 40 de capacidade total. Quando a capacidade de carga dos veículos é baixa, observa-se a necessidade de aumentar o número de veículos fornecidos como entrada ao algoritmo. Entretanto, mesmo com esse ajuste, a qualidade das soluções não apresenta melhoras tanto em relação ao valor de $UB$ quanto ao tempo de execução. Para instâncias grandes, encontrar soluções com baixa capacidade de carga é praticamente inviável. Dessa forma, a capacidade dos veículos é essencial para gerar soluções de melhor qualidade, sendo um fator mais determinante que a quantidade de veículos disponíveis.

# 7 CONCLUSÕES E TRABALHOS FUTUROS

Esta monografia apresentou dois modelos matemáticos para o Problema de Balanceamento Estático em Sistemas de Compartilhamento de Bicicletas (PBESCB). O estudo foi aplicado no sistema Bicicletar, o Sistema de Compartilhamento de Bicicletas (SCB) da cidade de Fortaleza, Ceará. Neste problema, busca-se criar rotas para diversos veículos capacitados, de modo a balancear as estações e melhorar a disponibilidade e usabilidade do sistema. Para a geração de soluções, foi aplicado o algoritmo de B&C sobre instâncias criadas a partir dos dados reais do Bicicletar.

As análises realizadas com base nos resultados obtidos indicam que a segunda formulação matemática proposta é superior à primeira, gerando soluções melhores em todas as instâncias criadas. A qualidade é superior em diferentes aspectos, como tamanho de percurso total e menor tempo de resolução para instâncias menores, entre outros critérios. A análise também apontou algumas limitações, especialmente em situações em que a capacidade total de carga dos veículos é pequena. Nesse contexto, a qualidade das soluções piora, dificultando encontrar soluções viáveis.

Para trabalhos futuros, pretende-se modificar a geração das demandas nas estações, considerando não apenas uma configuração final estática (metade da capacidade total), mas uma configuração específica para cada estação, permitindo atender melhor às particularidades do sistema e das estações. Além disso, pretende-se propor e implementar uma nova heurística para a solução inicial, com o objetivo de explorar possíveis ganhos de desempenho nas soluções geradas.

# REFERÊNCIAS


BEKTAS, T. The multiple traveling salesman problem: an overview of formulations and solution procedures. **Omega**, v. 34, n. 3, p. 209–219, 2006. ISSN 0305-0483. Disponível em: https://www.sciencedirect.com/science/article/pii/S0305048304001550.

BERBEGLIA, G.; CORDEAU, J.-F.; GRIBKOVSKAIA, I.; LAPORTE, G. Static pickup and delivery problems: a classification scheme and survey. **Top**, Springer, v. 15, p. 1–31, 2007.

CHASTRE, M. R.; ANDRADE, A. R. Rebalancing in bike sharing systems: Application to the lisbon case study. **Case Studies on Transport Policy**, v. 13, p. 101071, 2023. ISSN 2213-624X. Disponível em: https://www.sciencedirect.com/science/article/pii/S2213624X23001256.

CLARKE, G.; WRIGHT, J. W. Scheduling of vehicles from a central depot to a number of delivery points. **Operations research**, Informs, v. 12, n. 4, p. 568–581, 1964.

CORRAR, L. J.; GARCIA, E. A. d. R. Programação linear: uma aplicação à contabilidade de custos no processo de tomada de decisão. **Revista Cruzando Fronteras: Tendencias de Contabilidad Directiva para el Siglo XXI**, 2001.

DANTZIG, G. B. Notes on linear programming—part iii: Computational algorithm of the revised simplex method. Rand Corporation, 1953.

DANTZIG, G. B. Linear programming. **Operations Research**, v. 50, n. 1, p. 42 – 47, 2002. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0036234899&doi=10.1287%2fopre.50.1.42.17798&partnerID=40&md5=f0a66148e590fc4de96e3c4912344fdc.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management science**, Informs, v. 6, n. 1, p. 80–91, 1959.

DELL'AMICO, M.; HADJICOSTANTINOU, E.; IORI, M.; NOVELLANI, S. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. **Omega**, v. 45, p. 7–19, 2014. ISSN 0305-0483. Disponível em: https://www.sciencedirect.com/science/article/pii/S0305048313001187.

DEMAIO, P. Bike-sharing: History, impacts, models of provision, and future. **Journal of Public Transportation**, v. 12, n. 4, p. 41–56, 2009. ISSN 1077-291X. Disponível em: https://www.sciencedirect.com/science/article/pii/S1077291X22002600.

HERNáNDEZ-PéREZ, H.; SALAZAR-GONZáLEZ, J.-J. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. **Discrete Applied Mathematics**, v. 145, n. 1, p. 126–139, 2004. ISSN 0166-218X. Graph Optimization IV. Disponível em: https://www.sciencedirect.com/science/article/pii/S0166218X0400071X.

HILLIER, F. S.; LIEBERMAN, G. J. **Introdução à pesquisa operacional**. [*S. l.*]: McGraw Hill Brasil, 2013.

KADRI, A. A.; KACEM, I.; LABADI, K. A branch-and-bound algorithm for solving the static rebalancing problem in bicycle-sharing systems. **Computers  Industrial Engineering**, v. 95, p. 41–52, 2016. ISSN 0360-8352. Disponível em: https://www.sciencedirect.com/science/article/pii/S0360835216300183.

KARP, R. M. **Reducibility among combinatorial problems. Complexity of Computer Computations (The IBM Research Symposia Series)**. [*S. l.*]: Springer, New York, 1972.

LENSTRA, J. K.; KAN, A. R. Complexity of vehicle routing and scheduling problems. **Networks**, Wiley Online Library, v. 11, n. 2, p. 221–227, 1981.

MORA, R.; MORAN, P. Portraying perceptions of bike-sharing schemes (bss) in santiago, chile: What both regular users and pedestrians tell us. **Transportation Research Interdisciplinary Perspectives**, v. 13, p. 100534, 2022. ISSN 2590-1982. Disponível em: https://www.sciencedirect.com/science/article/pii/S2590198221002396.

NAGY, G.; SALHI, S. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. **European Journal of Operational Research**, v. 162, n. 1, p. 126–141, 2005. ISSN 0377-2217. Logistics: From Theory to Application. Disponível em: https://www.sciencedirect.com/science/article/pii/S0377221703008361.

PRESTES, e. **Introdução à Teoria dos Grafos**. [*S. l.*: *s. n.*], 2020.

RAVIV, T.; TZUR, M.; FORMA, I. A. Static repositioning in a bike-sharing system: models and solution approaches. **EURO Journal on Transportation and Logistics**, v. 2, n. 3, p. 187–229, 2013. ISSN 2192-4376. Disponível em: https://www.sciencedirect.com/science/article/pii/S2192437620301175.

SMITH, W. E. *et al.* Various optimizers for single-stage production. **Naval Research Logistics Quarterly**, v. 3, n. 1-2, p. 59–66, 1956.

TAHA, H. A. Chapter 1 - integer optimization and its applications. In: TAHA, H. A. (Ed.). **Integer Programming**. Academic Press, 1975. p. 1–33. ISBN 978-0-12-682150-5. Disponível em: https://www.sciencedirect.com/science/article/pii/B9780126821505500064.

WOLSEY, L. A. **Integer programming**. [*S. l.*]: John Wiley & Sons, 2020.