Decomposição de Benders para o problema da entrega compartilhada (crowdshipping)

Carlos Eduardo da Silva Trindade

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DA FACOM-UFMS

Data da Defesa:

Assinatura:_____

Decomposição de Benders para o problema da entrega compartilhada (crowdshipping)

Carlos Eduardo da Silva Trindade

Orientadora: *Edna Ayako Hoshino*

Monografia apresentada como requisito parcial para aprovação na Componente Curricular Não-Disciplinar Trabalho de Conclusão de Curso do curso de Graduação em Ciência da Computação, da Universidade Federal de Mato Grosso do Sul.

UFMS - Campo Grande
Dezembro/2025

Resumo

O crescimento do comércio eletrônico tem intensificado a demanda por soluções eficientes de entrega na última milha, impulsionando o interesse por modelos logísticos fundamentados na economia compartilhada. Entre essas abordagens, destaca-se o *employee-based crowdshipping*, no qual funcionários realizam entregas no trajeto para suas residências após o expediente. Este trabalho aborda o problema de associação entre demanda e oferta nesse contexto, considerando restrições operacionais como capacidade dos veículos, desvio máximo permitido e ganho mínimo esperado pelos funcionários. Para sua resolução, implementa-se um algoritmo exato proposto na literatura, fundamentado na lógica de Decomposição de Benders. Complementarmente, desenvolve-se uma meta-heurística GRASP (Greedy Randomized Adaptive Search Procedure) para a geração de soluções iniciais de alta qualidade, em substituição à Heurística Gulosa originalmente sugerida. Experimentos computacionais são realizados com instâncias disponibilizadas na literatura, a fim de comparar o desempenho entre as abordagens. Os resultados demonstram que a meta-heurística GRASP produz soluções iniciais superiores, configurando-se como uma alternativa mais robusta e eficaz para o problema de *employee-based crowdshipping*.

Sumário

CAPÍTULO

1

Introdução

1.1 Motivação

Na última década, o crescimento do comércio eletrônico tem impulsionado significantemente a demanda por serviços de transporte e logística. Em 2025, a receita deste tipo de comércio deverá atingir U$ 3,66 trilhões, com uma taxa média de crescimento anual de 6,29 % entre 2025 e 2030, alcançando um volume de mercado de U$ 4,6 trilhões até 2030 (Statista, 2025).

Esta tendência de crescimento vem alavancando a demanda por diversas modalidades de entrega, especialmente aquelas relacionadas à "entrega da última milha", isto é, todas as atividades de logística relacionadas à entrega de remessas a domicílios de clientes particulares em áreas urbanas. Esta etapa do processo logístico é notoriamente uma das mais onerosas, principalmente devido aos custos com pessoal (Boysen et al., 2021).

Um estudo de simulação com dados reais da Finlândia indica que as opções tradicionais de entrega baseadas em vans geram custos de 2 a 6 euros por remessa, variando conforme a densidade de clientes. Entre os principais fatores que cotribuem para esses custos elevados estão os engarrafamentos e a falta de vagas em ruas congestionadas, bem como o fato de que parte dos clientes não se encontra em casa para receber suas encomendas (Boysen et al., 2021).

Diante desse cenário, encontrar modelos de entrega alternativos que permitam reduzir os custos operacionais e otimizar o uso de recursos no transporte de mercadorias entre varejistas em consumidores tem se tornado um grande desafio ao setor logístico no mundo inteiro.

Um importante conceito, responsável por guiar na elaboração de alternativas eficientes ao problema de entrega nas últimas milhas, é o da Economia Compartilhada. De acordo com Botsman e Foley (2010), a economia compartilhada é um "sistema que ativa o valor inexplorado de ativos por meio de modelos e mercados que permitem maior eficiência e acesso".

Ao aplicar esse conceito ao contexto logístico, surge o modelo de entrega compartilhada, conhecido como *crowdshipping*. De acordo com Behrend e Maisel (2018), o *crowdshipping* pode ser definido como a aplicação de entregadores terceirizados na entrega de mercadorias de outras pessoas em rotas que eles fariam de qualquer forma, em suas viagens cotidianas.

Neste modelo, motoristas particulares que já possuem rotas planejadas executam entregas durante essas viagens. Para isso, eles estão dispostos a fazer pequenos desvios de sua rota original, com o objetivo de coletar e entregar encomendas no destino do consumidor. A cada entrega realizada, o motorista recebe uma compensação financeira. Ao todo, os ganhos recebidos pelas entregas realizadas podem cobrir, pelo menos parcialmente, os custos de sua viagem original.

Desta forma, quando bem planejada, a abordagem de *crowdshipping* pode proporcionar entregas mais rápidas e de baixo custo, quando comparadas a empresas de entregas comerciais, ao mesmo tempo em que viabiliza o aproveitamento de viagens particulares, contribuindo para uma logística urbana mais eficiente e sustentável.

De acordo com Boysen *et al.* (2022), o *crowdshipping* pode ser classificado em duas modalidades principais: baseada em motoristas e baseada em clientes. Na primeira, empresas como Amazon e Uber contratam motoristas independentes que são pagos por hora e se inscrevem antecipadamente para realizar as entregas. Na segunda, clientes realizando compras em uma loja física podem se voluntariar para realizar entregas de consumidores online de sua vizinhança.

Segundo os autores, a grande desvantagem dessas modalidades se encontra na dificuldade dos varejistas em garantir seus padrões de qualidade quando delegam o serviço de entrega a um indivíduo terceirizado, caracterizando a falta de confiabilidade como um grande obstáculo na aplicação do *crowdshipping*.

Esta problemática deu origem a uma terceira modalidade, baseada em funcionários, denominada *employee-based crowdshipping*. Nessa abordagem, em vez de contratar motoristas terceirizados ou clientes, os varejistas oferecem o serviço de entrega aos próprios funcionários, que podem realizá-lo após o expediente de trabalho, no trajeto que fariam para casa. De acordo com Boysen *et al.* (2022), *crowdshipping* baseado em funcionários apresenta diversas

vantagens, quando comparada às outras modalidades:

- Funcionários são conhecidos pelo empregador, o que facilita identificar entregadores confiáveis para os serviços.
- Os funcionários podem ser treinados e equipados com roupas que os identifiquem. Isso habilita uma melhor interação com o cliente no momento da entrega, transmitindo maior confiança.
- O esforço de coordenação para planejamento e controle das entregas é reduzido, uma vez que os funcionários já estão integrados à estrutura organizacional do varejista.

Essas questões evidenciam o potencial do *employee-based crowdshipping* como uma alternativa viável para o problema de entregas na última milha, possibilitando a redução de custos logísticos, trazendo eficiência e sustentabilidade ao processo de entrega urbana e, ao mesmo tempo, garantindo confiabilidade e oferecendo benefícios adicionais aos funcionários envolvidos.

1.2 Objetivos

O objetivo geral deste trabalho é desenvolver e avaliar um modelo de entrega urbana baseado no conceito de *employee-based crowdshipping*, utilizando uma abordagem exata com o método de Branch-and-Benders-Cut (BBC), proposta por Boysen *et al.* (2022). Além disso, propõe-se o desenvolvimento de uma meta-heurística do tipo GRASP (Greedy Randomized Adaptive Search Procedure) para encontrar soluções iniciais de alta qualidade com um custo computacional reduzido, de modo a aprimorar o desempenho do método exato, uma vez que uma abordadem de otimização adequada demonstra ser crucial para o sucesso do *crowdshipping*. Para o alcance desse objetivo, foram definidos os seguintes objetivos específicos:

- Revisar a literatura existente sobre o problema de entregas na última milha, com ênfase na aplicação do *crowdshipping* baseado em funcionários, bem como os métodos exatos e heurísticos utilizados para sua resolução.
- Formalizar matematicamente o problema de entrega na última milha, no contexto do *employee-based crowdshipping*, considerando as restrições operacionais como capacidade de transporte, desvios de rota e ganhos mínimos, conforme apresentado por Boysen *et al.* (2022).
- Implementar o modelo proposto utilizando o método exato de Branch-and-Benders-Cut (BBC) para resolver o problema de entrega na última milha, empregando o resolvedor SCIP como ferramenta de otimização.

- Desenvolver uma meta-heurística do tipo GRASP (Greedy Randomized Adaptive Search Procedure) com o objetivo de gerar soluções iniciais de alta qualidade e baixo custo computacional, contribuindo para o desempenho do método exato.

- Comparar os resultados obtidos com aqueles apresentados na literatura, analisando a qualidade das soluções iniciais, o custo computacional e o comportamento do método de Branch-and-Benders-Cut (BBC) ao utilizar as soluções geradas pela meta-heurística como ponto de partida.

- Discutir o potencial de aplicação prática do modelo desenvolvido no contexto de entregas urbanas sustentáveis, destacando suas vantagens e limitações.

1.3 Trabalhos Relacionados

Encontrar modelos de entrega que permitam otimizar o uso de recursos de transporte tem se tornado uma tendência crescente no setor logístico em todo o mundo. Consequentemente, observa-se um aumento significativo no número de publicações dedicadas a esse tema nos últimos anos. Para oferecer uma visão geral sobre esse assunto, são considerados trabalhos relacionados à entrega compartilhada, logística urbana e entrega na última milha, além de um artigo de revisão dedicado ao algoritmo de Decomposição de Benders.

O conceito de *employee-based crowdshipping* consiste em utilizar funcionários para realizar entregas em rotas que já percorreriam, aproveitando seus deslocamentos habituais. Boysen *et al.* (2022) propõem um algoritmo de otimização exato, baseado na Decomposição de Benders, para combinar a oferta de serviços de transporte, disponibilizada por esses funcionários, com a demanda por entregas de encomendas. Neste problema, os autores buscam maximizar o número de pacotes associados a cada funcionário voluntário de um centro de distribuição, de modo que as entregas sejam realizadas após o expediente, no caminho que fariam para casa. O problema considera ainda restrições operacionais impostas pelos próprios funcionários, como capacidade de carga, desvio máximo da rota e ganho mínimo aceitável. Os autores reportam um desempenho superior do algoritmo proposto em comparação com a abordagens tradicionais baseadas em Programação Inteira Mista (*Mixed Integer Programming* – MIP).

Rahmaniani *et al.* (2017) apresentam uma revisão abrangente do metódo de Decomposição de Benders (BD), abordando desde sua formulação clássica até suas variações mais recentes, melhorias e aplicações. Os autores reportam um número crescente de publicações que utilizam o método de BD para

resolver problemas complexos de otimização, destacando sua eficácia em lidar com problemas de grande escala. Os autores mencionam ainda a técnica de Branch-and-Benders-Cut (BBC) como uma variação moderna do método clássico, destacando suas vantagens no uso de ferramentas de reotimização dos resolvedores MIP.

De forma complementar, Alvarez *et al.* (2024) introduzem uma variação do problema clássico de roteamento de veículos, denominado *Consistent Vehicle Routing Problem with Stochastic Customers and Demands* (ConVRP-SCD). Esta variação incorpora incertezas na presença e na demanda dos clientes, além da restrições de consistência nas atribuições de motoristas. Os autores propõem um modelo estocástico em dois estágios, resolvido por meio de uma abordagem híbrida de Sample Average Approximation (SSA) combinada com Branch-and-Cut, equilibrando custo operacional com regularidade no atendimento.

Díaz-Ríoz e Salazar-González (2024) ampliam o estudo da consistência temporal em problemas de roteamento de veículos, propondo formulações matemáticas para o Consistent Travelling Salesman Problem (CSTP), uma variação do TSP em que os horários de atendimento aos clientes devem permanecer semelhante em diferentes dias. Os autores analisam duas versões do problema, uma sem tempo de espera (CTSP1) e outra permitindo esperas (CTSP2), e apresentam uma nova formulação baseada em fluxo multi-commodity, adequada para resolução via Decomposição de Benders. Esta contribuição destaca a aplicabilidade de BD na resolução de problemas complexo de roteamento de veículos com restrições temporais.

A literatura revisada evidencia a evolução das abordagens de otimização aplicadas à logística urbana e ao roteamento de veículos, destacando o papel crescente da Decomposição de Benders e suas variações modernas, como o Branch-and-Benders-Cut (BBC) e demonstrando sua eficácia na resolução de problemas de grande escala e elevada complexidade combinatória, tanto em contextos de entregas colaborativas quanto em modelos de roteamento consistentes e estocásticos. Assim, o uso de métodos baseados em Benders surgem como tendência promissora para o desenvolvimento de modelos exatos mais eficientes na área de entrega urbana e colaborativa, servindo como base conceitual para o presente trabalho.

1.4 Organização do Texto

O presente trabalho está estruturado da seguinte forma: no Capítulo 2, apresenta-se a descrição formal do problema de entrega na última milha, baseada no conceito de *employee-based crowdshipping*, incluindo a formulação matemática do modelo MIP, proposto na literatura, e suas restrições operaci-

onais. No Capítulo 3, discute-se a metodologia adotada, detalhando o método de Branch-and-Benders-Cut (BBC) e a meta-heurística GRASP desenvolvida para gerar soluções iniciais. O Capítulo 4 é dedicado à implementação do modelo, apresentando os aspectos técnicos e as ferramentas utilizadas. No Capítulo 5, são apresentados os resultados obtidos, acompanhados de uma análise comparativa com a literatura existente. Finalmente, o Capítulo 6 conclui o trabalho, destacando as contribuições, limitações e sugestões para pesquisas futuras.

CAPÍTULO

2

Descrição Formal do Problema

Este capítulo apresenta as configurações do problema abordado. Na Seção 2.1, é realizada a caracterização do problema, acompanhada de um exemplo ilustrativo. Em seguida, a Seção 2.2 traz uma descrição formal, e a Seção 2.3 apresenta o modelo de Programação Inteira Mista (MIP) proposto por Boysen *et al.* (2022).

2.1 Caracterização do problema

A tarefa consiste em associar a oferta e a demanda entre dois conjuntos: um de funcionários voluntários para a realização das entregas e outro de encomendas a serem entregues aos clientes. Do lado da oferta, cada funcionário voluntário especifica um volume de capacidade máxima que consegue carregar a bordo de seu carro, seu destino após o trabalho (por exemplo, o endereço de sua casa), o máximo de tempo de viagem adicional à sua rota original e um ganho mínimo esperado por unidade tempo. Observe que essas informações podem variar de um dia para o outro, mas na maioria das vezes as informações padrão podem ser utilizadas, de modo que, ao vincular o horário de trabalho do respectivo funcionário, todas as informações necessárias podem ser fornecidas com apenas alguns cliques, em algum aplicativo para smartphone que gerencie este procedimento, antes do turno de trabalho.

Do lado da demanda, temos um conjunto de pacotes para ser entregue aos clientes durante o dia. Cada um desses pacotes possui um volume e um endereço de entrega. Observe que informações adicionais sobre prazos de entrega previstos não são relavantes. Apenas as remessas que chegam dentro do prazo, quando entregues por um funcionário após o próximo turno de tra-

balho ou, caso não sejam atribuídas com sucesso, por uma transportadora comercial, são consideradas candidatas adequadas para o *crowdshipping*.

Ao associar a oferta e a demanda, devem ser satisfeitas as seguintes condições:

- Nenhum funcionário pode transportar um volume de remessas superior à capacidade de seu veículo;
- O desvio na rota causado pelas entregas não pode exceder o tempo máximo adicional aceitável pelo funcionário;
- O ganho obtido por cada funcionário não deve ser inferior a um valor mínimo, por unidade de tempo, especificado.

Realizar todas essas associações produz custos adicionais para os varejistas de modo que o *crowdshipping* baseado em funcionários só se torna atrativo quando a taxa fixa paga a um funcionário for menor que a taxa a pagar à uma transportadora terceirizada para realizar as entregas dos pacotes remanescentes que não puderam ser atribuídos. Parte-se do pressuposto de que as entregas realizadas através do *crowdshipping* são mais econômicas, uma vez que aproveitam trajetos que os funcionários já percorreriam em suas rotinas habituais. Dessa forma, maximizar o número de pacotes entregues pelos funcionários implica na minimização dos custos de entrega para o varejista, sendo, portanto, este o objetivo a ser otimizado.

Exemplo: considere um centro de distribuição com cinco funcionários, enumerados de 0 a 4, voluntários para o *crowdshipping* e 10 pacotes a serem entregues nas casas dos clientes. Os tempos de viagem entre as localizações são representados como os pesos das arestas da Figura 2.1. Embora exista uma aresta entre cada par de localizações, apenas um subconjunto delas é exibido na Figura 2.1, a fim de preservar a legibilidade. Cada funcionário possui as seguintes características:

- Capacidades de 18, 20, 22, 14 e 15 unidades para F0, F1, F2, F3 e F4, respectivamente;
- Desvios máximos tolerados de 11, 26, 8, 8 e 15 unidades de tempo, respectivamente;
- Remuneração mínima esperada de 0,2, 0,18, 0,23, 0,18 e 0,18 unidades monetárias por unidade de tempo, respectivamente.

Além disso, os pacotes P0, P1, P2, P3, P4, P5, P6 e P8 ocupam 1 unidade de volume, enquantos os pacotes P7 e P9 ocupam 5 unidades. O centro de

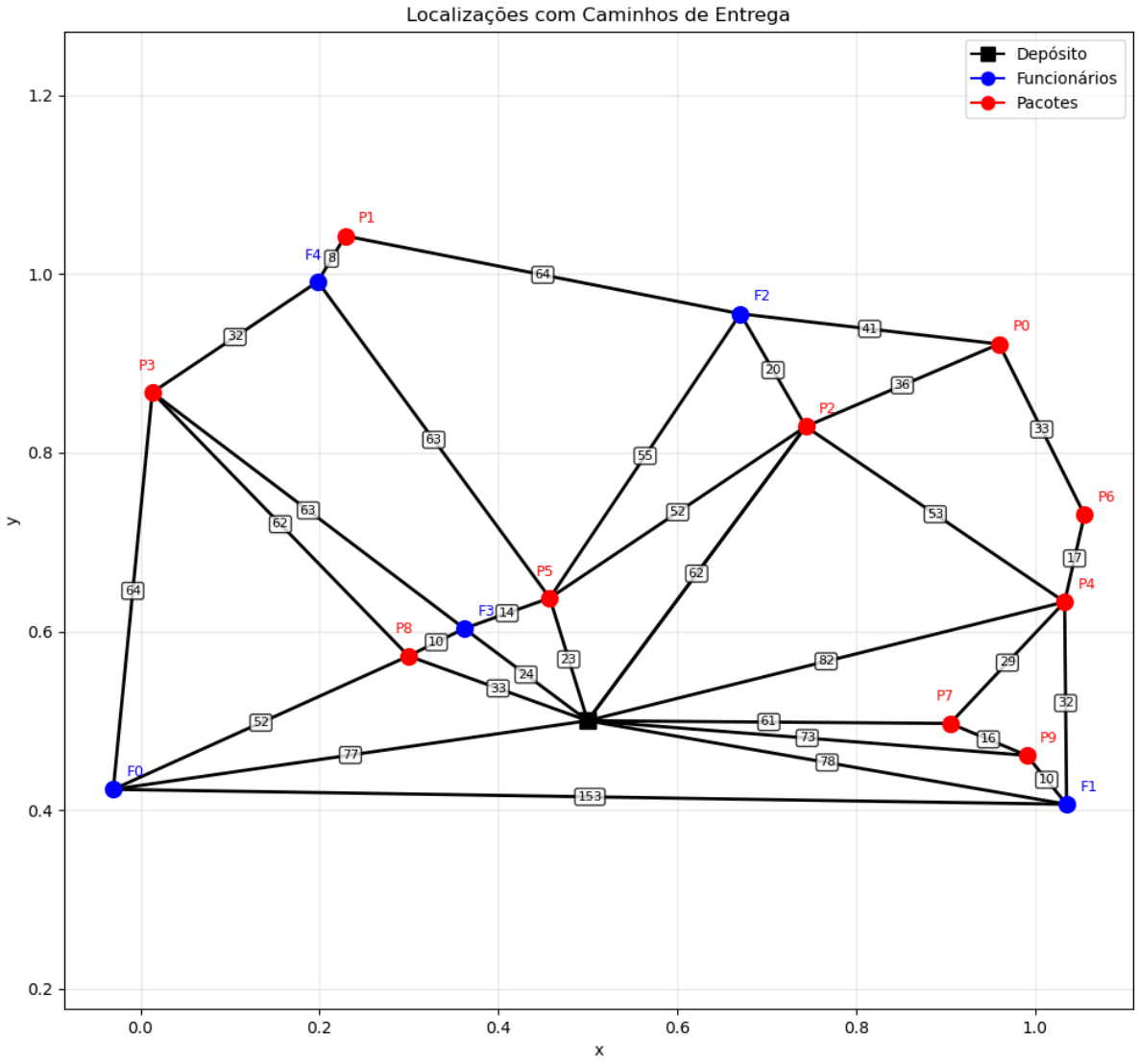

Figura 2.1: Exemplo ilustrativo da instância do problema.

distribuição oferece 2 unidades monetárias ao funcionário por cada pacote entregue ao cliente.

Uma solução ótima para esta instância do problema é apresentada na Figura 2.2, na qual quatro pacotes são atribuídos com sucesso a funcionários do centro de distribuição para realização de entregas após o expediente. Nessa solução, o funcionário F1 é responsável pelos pacotes P7 e P9, o que implica em um desvio de 9 unidades de tempo em sua rota original e resulta em um ganho aproximado de 0,44 unidades monetárias por unidade de tempo. O funcionário F0 realiza a entrega do pacote P8, acrescentando 8 unidades de tempo à sua rota e obtendo um ganho de 0,25 por unidade de tempo. Por fim, o funcionário F4 entrega o pacote P5, com um desvio adicional de 3 unidades de tempo e um ganho aproximado de 0,66 unidades monetárias por unidade de tempo. Neste exemplo, 6 pacotes não foram atribuídos a funcionários, portanto, deverão ser entregues por uma transportadora terceirizada.

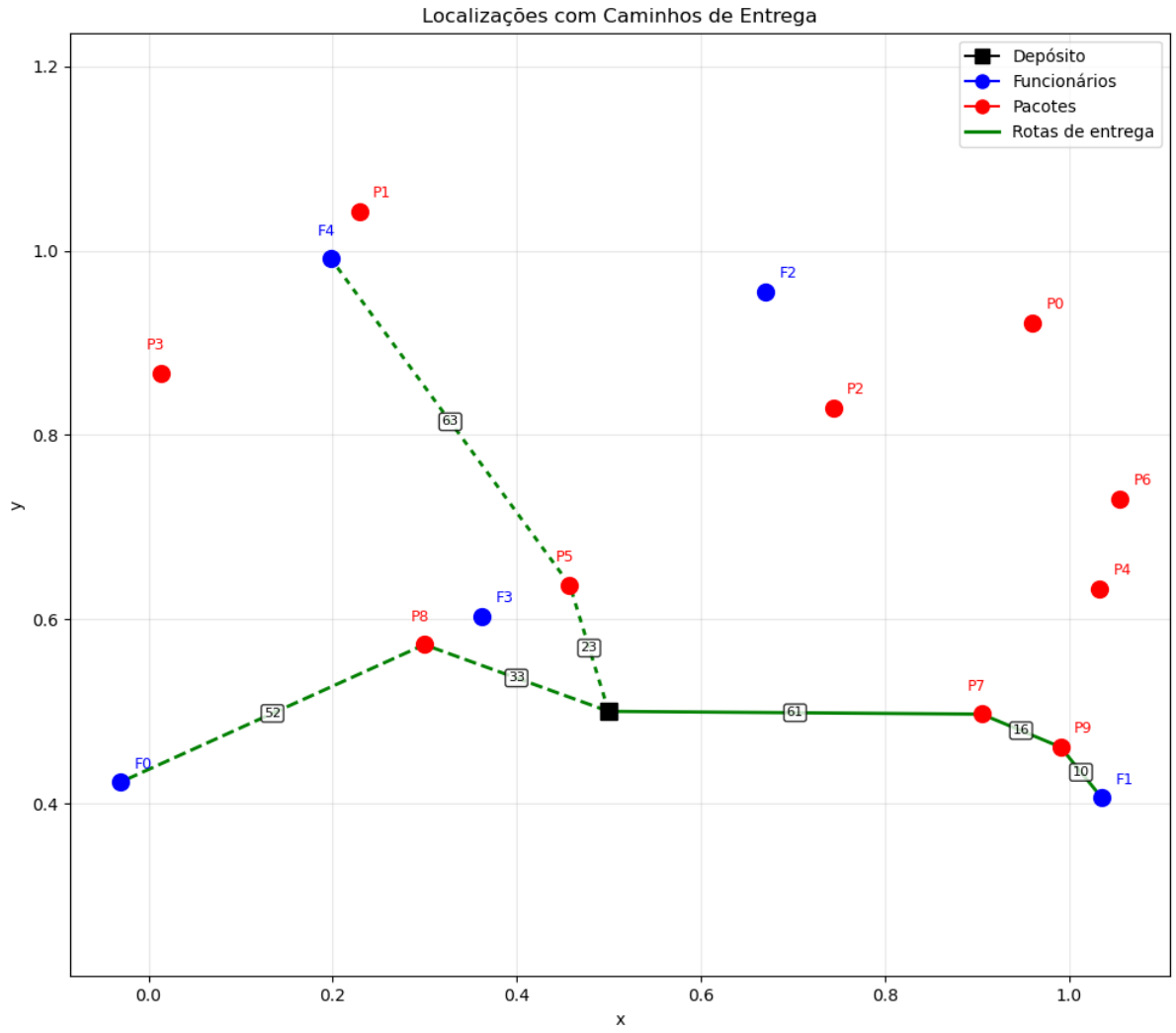

Figura 2.2: Solução para a instância de exemplo.

A partir desse cenário, a formulação formal do problema combinado de atribuição e roteamento para o *crowdshipping* baseado em funcionários é apre-

sentada na seção seguinte.

2.2 Definição do problema

Dado um conjunto $J = \{1, ..., n\}$ de pacotes a serem atribuídos a funcionários voluntários do conjunto $I = \{n+1, ..., n+m\}$. Cada pacote $j \in J$ possui um destino de entrega e cada funcionário $i \in I$ possui um endereço residencial, que é seu destino final. O tempo de de viagem entre quaisquer duas localizações é conhecido. Considere que $N = \{0\} \cup J \cup I$ representa o conjunto de todas as localizações, na qual 0 representa o depósito e c_{ij} o tempo de viagem entre quaisquer duas localizações $i, j \in N$. Desta forma, considere que:

- c_{0j}: a duração da viagem entre o depósito e o endereço do pacote $j \in J$;
- $c_{jj'}$: a duração da viagem entre os endereços dos pacotes j e j' (com $j, j' \in J$ e $j \neq j'$);
- c_{ji} a duração da viagem entre o endereço do pacote $j \in J$ e o endereço da casa do funcionário $i \in I$;
- c_{0i}: a duração da viagem entre o depósito e o endereço da casa do funcionário $i \in I$ se nenhuma entrega é realizada.

Observe que os tempos de viagem $c_{0j'}$ e $c_{jj'}$ podem conter um tempo adicional esperado do ato de entrega do funcionário ao cliente do pacote $j' \in J$. Assume-se que é possível viajar entre quaisquer duas localizações e que a desigualdade triangular é válida.

Funcionários utilizam seus carros particulares, que possuem uma capacidade limitada Q_i, da qual cada pacote ocupa q_j unidades de volume. Além disso, os funcionários não estão dispostos a fazer longos desvios, portanto, cada funcionário $i \in I$ possui um tempo de desvio máximo $c_i^{\max}$ que estão dispostos a aceitar acima de seu tempo normal do trajeto para casa c_{0i}. Finalmente, os funcionários recebem uma quantia fixa F por cada pacote entregue e não estão dispostos a aceitar menos do que uma quantia f_i por unidade de tempo. Observe que os ganhos por unidade de tempo dependem tanto do número de pacotes entregues quanto do tempo gasto para entregá-los. Quanto maior for o número de entregas e menor o desvio incorrido, maior será os ganhos por unidade de tempo.

Uma solução para o problema consiste em uma partição $\{R_0, ..., R_m\}$ do conjunto J e uma permutação π_i de cada conjunto R_i, representando o caminho que o funcionário i realiza as entregas dos pacotes em R_i, $\forall i \in I$. Portanto, $\pi_i(k) \in R_i \subseteq J$ é o k-ésimo pacote a ser entregue pelo funcionário i. Considere que os pacotes em R_0 não foram atribuídos a nenhum funcionário e, portanto,

serão entregues por uma transportadora terceirizada. Uma solução é viável se e somente se satisfaz as seguintes condições:

- $\sum_{j \in R_i} q_j \leq Q_i, \ \forall i \in I$: nenhum funcionário pode carregar mais pacotes do que cabe em seu carro;
- $g(\pi_i) \leq c_i^{\max} + c_{0i}, \ \forall i \in I$: nenhum funcionário pode exceder seu desvio máximo aceito (considere que $g(\pi_i) = c_{0,\pi_i(1)} + \sum_{k=1}^{|R_i|-1} c_{\pi_i(k),\pi_i(k+1)} + c_{\pi_i(|R_i|),i}$ seja o tempo da viagem π_i);
- $\frac{|R_i| \cdot F}{g(\pi_i) - c_{0i}} \geq f_i$: cada funcionário deve receber pelo menos o ganho mínimo por unidade de tempo esperado.

Uma solução é considerada ótima se for viável e maximizar o número de pacotes atribuídos aos funcionários ou, equivalentemente, minimizar o número de pacotes em R_0 não atribuídos. Essa tarefa de atribuição configura um problema de otimização NP-difícil, no qual as restrições de capacidade e a tarefa de roteamento de cada funcionário podem ser modeladas, respectivamente, como um problema da mochila binária e um problema do caixeiro viajante.

2.3 Modelo MIP

Com base na definição formal do problema apresentada na seção anterior, o modelo de Programação Inteira Mista (MIP) proposto por Boysen *et al.* (2022) é apresentado a seguir.

$$\text{[MIP] Maximize} \quad \sum_{i \in I} \sum_{j \in J} x_{ij} \tag{1}$$

$$\text{Sujeito a} \quad \sum_{i \in I} x_{ij} \leq 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{j \in J} q_j \cdot x_{ij} \leq Q_i \qquad \forall i \in I \tag{3}$$

$$y_{jj'} + y_{j'j} \geq x_{ij} + x_{ij'} - 1 \qquad \forall i \in I, \ j < j' \in J \tag{4}$$

$$t_j \geq c_{0j} \cdot x_{ij} \qquad \forall i \in I, \ j \in J \tag{5}$$

$$t_{j'} \geq t_j + c_{jj'} - (M - c_{jj'}) \cdot (1 - y_{jj'}) \qquad \forall i \in I, \ j \neq j' \in J \tag{6}$$

$$t_i \geq t_j + c_{ji} \cdot x_{ij} - (1 - x_{ij}) \cdot M \qquad \forall i \in I, \ j \in J \tag{7}$$

$$t_i \leq c_i^{\max} + c_{0i} \qquad \forall i \in I \tag{8}$$

$$\sum_{j \in J} x_{ij} \cdot F \geq (t_i - c_{0i}) \cdot f_i \qquad \forall i \in I \tag{9}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I,\ j \in J \qquad (10)$$

$$y_{jj'} \in \{0,1\} \qquad \forall j \neq j' \in J \qquad (11)$$

$$t_i \geq c_{0i} \qquad \forall i \in I \qquad (12)$$

Seja x_{ij} uma variável binária que assume o valor 1 se o pacote $j \in J$ for atribuído ao funcionário $i \in I$ e 0 caso contrário; $y_{jj'}$ uma variável binária que assume o valor 1 se o pacote j for entregue antes do pacote j' pelo mesmo funcionário e 0 caso contrário; e t_i uma variável contínua que representa o tempo total da viagem do funcionário $i \in I$. M é um valor suficientemente grande para garantir a validade das restrições.

A função objetivo (1) maximiza o número total de pacotes atribuidos aos funcionários. A restrição (2) garante que cada pacote seja atribuído a no máximo um funcionário. A restrição (3) assegura que a capacidade do veículo de cada funcionário não seja excedida. As restrições (4) a (7) modelam o roteamento de cada funcionário, garantindo a ordem correta das entregas e o cálculo do tempo total da viagem. A restrição (8) assegura que o tempo total da viagem de cada funcionário não exceda o desvio máximo permitido. A restrição (9) garante que o ganho por unidade de tempo de cada funcionário seja pelo menos o valor mínimo esperado. As restrições (10) e (11) definem as variáveis binárias, enquanto a restrição (12) assegura que o tempo total da viagem seja pelo menos o tempo direto do depósito até a casa do funcionário.

CAPÍTULO

3

Metodologia

Neste capítulo, serão descritas as técnicas utilizadas para a realização deste trabalho. Na Seção 3.1, será apresentada a Programação Linear Inteira (PLI), que é a base para a formulação dos problemas abordados, especialmente os Programas Lineares Inteiros Mistos (MIP). Na Seção 3.2, será detalhada a técnica de Decomposição de Benders e sua abordagem moderna, o Branch-and-Benders-Cut, utilizada para resolver problemas de grande escala. Por fim, na Seção 3.3, será descrito o método Greedy Randomized Adaptive Search Procedure (GRASP), uma meta-heurística empregada para encontrar soluções aproximadas de forma eficiente.

3.1 Programação Linear Inteira

Para explicarmos a Programação Linear Inteira (PLI), devemos começar com a definição de Programação Linear (PL), uma vez que a PLI é uma versão mais restrita da PL, onde as variáveis de decisão são restringidas a valores inteiros. A Programação Linear é uma técnica matemática utilizada para otimizar (minimizar ou maximizar) uma função linear sujeita a um conjunto de restrições também lineares. Em outras palavras, a PL busca encontrar o melhor valor possível para uma função objetivo, respeitando as limitações operacionais impostas pelo cenário do problema em que está sendo aplicada. Um exemplo clássico de problema que pode ser modelado utilizando PL é o problema da mochila fracionária, no qual o objetivo é maximizar o valor total dos itens colocados em uma mochila, respeitando a capacidade máxima de peso que a mochila pode suportar. Observe que na mochila fracionária, é permitido dividir os itens, ou seja, pegar uma fração do item se necessário. Wolsey (2020)

fornece uma formulação genérica para problemas de Programação Linear da seguinte forma:

$$\max\{cx : Ax \leq b, x \geq 0\} \tag{PL}$$

No qual A é o objeto matemático que representa as informações das restrições do problema, isto é, as limitações operacionais que devem ser respeitadas. Este objeto é representado por uma matriz de coeficientes que relacionam as variáveis de decisão com os limites ou recursos disponíveis, representados pelo vetor b. O vetor c representa os coeficientes da função objetivo, indicando a contribuição de cada variável de decisão para o valor total que se deseja otimizar. As variáveis de decisão são representadas pelo vetor x, que contém as quantidades que se deseja determinar para maximizar a função objetivo, sujeitas às restrições impostas por $Ax \leq b$ e à condição de não negatividade $x \geq 0$.

O algoritmo proposto por Dantzig (1951), conhecido como o Método Simplex, é amplamente utilizado para resolver problemas de Programação Linear, estando presente em diversos resolvedores comerciais e de código aberto, como CPLEX, Gurobi, GLPK, SCIP, entre outros. Este método é um algoritmo iterativo que navega pelos vértices do politopo definido pelas restrições lineares do problema, isto é, o conjunto de soluções viáveis formado pelas interseções das restrições, buscando o vértice que maximiza (ou minimiza) a função objetivo do problema. O Método Simplex é eficiente para a maioria dos problemas de PL encontrados na prática, embora existam casos em que seu desempenho pode ser pior do que o esperado.

Wolsey (2020) define a Programação Linear Inteira (PLI) como uma extensão da Programação Linear, no qual as variáveis de decisão são restritas a valores inteiros. A PLI é particularmente útil em situações onde as decisões envolvem valores discretos, como a alocação de recursos, planejamento de produção, roteamento de veículos, entre outros. Como no exemplo da mochila fracionária, se considerarmos que os itens não podem ser divididos, ou seja, devemos pegar o item inteiro ou não pegá-lo, estamos diante do problema da mochila inteira ou binária, um clássico exemplo de PLI. Há diversas variações de problemas de PLI, como a Programação Linear Inteira Mista (MIP), Programação Linear Binária (BIP), entre outros.

A formulação genérica para problemas de Programação Linear Inteira segue a mesma estrutura da PL, mas com a adição da restrição de integralidade nas variáveis de decisão, conforme apresentando por Wolsey (2020):

$$\max\{cx : Ax \leq b, x \geq 0, x \in \mathbb{Z}^n\} \tag{PLI}$$

Já para a Programação Linear Inteira Mista (MIP), no qual apenas algumas

variáveis de decisão são restritas a valores inteiros, a formulação genérica forneceda por Wolsey (2020) é a seguinte:

$$\begin{aligned} \text{(MIP)} \quad & \max\ cx + hy \\ & Ax + Gy \leq b \\ & x \geq 0,\ x \in \mathbb{Z}^n,\ y \geq 0 \end{aligned}$$

No qual as variáveis de decisão são divididas em dois conjuntos: o vetor x, que contém as variáveis inteiras, e o vetor y, que contém as variáveis contínuas. O vetor h representa os coeficientes da função objetivo associados às variáveis contínuas, enquanto a matriz G relaciona essas variáveis com as restrições do problema.

A Programação Linear Binária, utilizada no contexto do problema deste trabalho, é um caso especial onde as variáveis de decisão são restritas aos valores do conjunto $\{0,1\}$, indicando decisões binárias, como "sim"ou "não". No contexto da mochila binária, por exemplo, uma variável de decisão igual a 1 indica que o item foi incluído na mochila, caso contrário, o item não foi incluído. Já no contexto do crowshipping baseado em funcionário que será abordado neste trabalho, uma variável de decisão igual a 1 indica que determinado pacote foi atribuído a um funcionário para entrega, enquanto uma variável igual a 0 indica que o pacote não foi atribuído a esse funcionário. A formulação genérica para problemas de Programação Linear Binária é dada por Wolsey (2020) da seguinte forma:

$$\begin{aligned} \text{(BIP)} \quad & \max\ cx \\ & Ax \leq b \\ & x \in \{0,1\}^n \end{aligned}$$

Para a resolução de problemas de PLI, MIP e BIP, diversos algoritmos exatos foram desenvolvidos, sendo o algoritmo Branch-and-Bound, proposto por Land e Doig (1960), amplamente utilizado. Este algoritmo utiliza uma técnica de enumeração que divide o problema original em subproblemas menores, resolvendo-os de forma recursiva e descartando aqueles que não podem levar a uma solução ótima. Na maior parte dos casos, a resolução dos subproblemas é feita utilizando o Método Simplex para resolver a relaxação linear do problema, ou seja, o problema obtido ao remover a restrição de integralidade das variáveis de decisão. A resolução desta relaxação linear fornece um limitante supeior (ou inferior, dependendo se o problema é de maximização ou minimização) para o valor ótimo do problema original, auxiliando na exclusão de subproblemas que não podem melhorar a solução atual.

Além da técnica de Branch-and-Bound, uma outra abordagem amplamente utilizada para resolver problemas de PLI é a técnica de Branch-and-Cut, que combina o método de Branch-and-Bound com a geração de cortes (restrições adicionais) para melhorar a relaxação linear do problema. Esta técnica é especialmente eficaz para problemas de grande escala, onde a geração de cortes pode reduzir significativamente o espaço de busca e acelerar a convergência para a solução ótima.

Todos os métodos mencionados acima são implementados no resolvedor de código aberto SCIP, que será utilizado neste trabalho para resolver os modelos matemáticos propostos.

3.2 Decomposição de Benders

A Decomposição de Benders, proposta por Benders (1962), é uma técnica utilizada para resolver problemas de otimização de grande escala, especialmente aqueles que podem ser divididos em dois conjuntos de variáveis: variáveis mestres e variáveis subordinadas. Conforme Wolsey (2020), a decomposição de Benders é uma abordagem para resolver problemas de programação linear inteira mista (MIP) que possuem variáveis de decisão inteiras e contínuas, permitindo a decomposição do problema original em subproblemas que envolvam apenas um tipo de variável.

A ideia central do algoritmo de Decomposição de Benders é iterativamente resolver um problema mestre que envolve apenas as variáveis inteiras, enquanto as variáveis contínuas são tratadas em subproblemas separados. A cada iteração, o problema mestre é resolvido para obter valores provisórios para as variáveis inteiras, que são então utilizados para resolver os subproblemas contínuos. Com base nas soluções dos subproblemas, cortes de Benders são gerados e adicionados ao problema mestre para refinar a solução. Os cortes podem ser de dois tipos: cortes de viabilidade, que garantem que as soluções do problema mestre sejam viáveis para os subproblemas, e cortes de otimalidade, que melhoram a função objetivo do problema mestre com base nas soluções dos subproblemas. O processo é repetido até que uma solução ótima seja encontrada ou até que um critério de parada seja atendido.

Uma variação moderna da Decomposição de Benders é abordada por Rahmaninani *et al.* (2017), denominada Branch-and-Benders-Cut (BBC). Ao contrário do método clássico de Decomposição de Benders, onde um problema de PLI é resolvido até a otimalidade a cada iteração e, a cada corte gerado, uma nova árvore de Branch-and-Bound é construída, o método BBC integra a geração de cortes de Benders diretamente no processo de Branch-and-Bound. Isso significa que, durante a exploração da árvore de Branch-and-Bound, cor-

tes de Benders são gerados e adicionados dinamicamente aos nós da árvore. Essa abordagem permite um maior aproveitamento das ferramentas de reotimização disponíveis em resolvedores modernos, além de uma superioridade numérica significativa em relação ao método clássico.

3.3 Greedy Randomized Adaptive Search Procedure (GRASP)

O GRASP (Greedy Randomized Adaptive Search Procedure) é uma metaheurística desenvolvida para resolver problemas de otimização combinatória, introduzida por Feo e Resende (1995). Uma heurística é uma técnica que busca encontrar boas soluções para problemas complexos de forma eficiente, isto é, em um tempo computacional razoável, sem garantir a otimalidade da solução encontrada.

O GRASP é um procedimento iterativo, no qual cada iteração consiste em duas fases, a fase construtiva e a fase de busca local. Na fase construtiva, uma solução viável é produzida, e na fase de busca local, uma busca na vizinhança da solução construída é realizada, na tentativa de encontrar uma solução ainda melhor. A melhor solução encontrada ao longo de todas as iterações é retornada como a solução final do algoritmo.

Na fase de construção, uma solução viável é construída elemento por elemento, combinando uma abordagem gulosa com uma seleção aleatória. Inicialmente, um conjunto de candidatos é criado, contendo todos os elementos que podem ser adicionados à solução. A cada passo, uma lista restrita de candidatos (RCL) é formada, contendo os melhores candidatos, com base em um critério guloso, que mensura o benefício de adicionar cada canditado à solução atual. A heurística é adaptativa, pois o benefício de cada candidato é recalculado a cada iteração, levando em consideração a solução parcial construída até o momento. O componente probabilístico é introduzido ao selecionar aleatoriamente um elemento da RCL para ser adicionado à solução, o que ajuda a diversificar as soluções geradas a cada iteração. O parâmetro $\alpha \in [0,1]$ é utilizado para controlar o tamanho da RCL, onde $\alpha = 0$ resulta em uma abordagem totalmente gulosa, enquanto $\alpha = 1$ permite uma seleção completamente aleatória, conforme descrito por Resende (1998). A RCL é definida da seguinte forma:

$$\text{RCL} = \{s \in C \mid g(s) \leq \underline{s} + \alpha(\overline{s} - \underline{s})\}$$

no qual C é o conjunto de candidatos, $g(s)$ é a função que avalia o benefício de adicionar o candidato s à solução atual, $\underline{s}$ é o valor mínimo de $g(s)$ entre to-

dos os candidatos em C, e $\bar{s}$ é o valor máximo de $g(s)$ entre todos os candidatos em C. Desta forma, a RCL contém os candidatos cujo benefício está dentro de um intervalo definido pelo parâmetro α.

Após a construção da solução inicial, a fase de busca local é iniciada. Nesta fase, a vizinhança da solução atual é explorada, buscando melhorias incrementais. Diversas estratégias podem ser empregadas para definir a vizinhança, como troca de elementos, inserção ou remoção de elementos, entre outras. A busca local continua até que nenhuma melhoria adicional possa ser encontrada na vizinhança da solução atual.

Todo o procedimento é repetido por um número $maxitr$ pré-definido de iterações. Ao final, a melhor solução encontrada ao longo de todas as iterações é retornada como solução final do algoritmo GRASP.

CAPÍTULO

4

Trabalho Desenvolvido

Neste capítulo, são apresentados os métodos desenvolvidos para a resolução do problema de *crowdshipping* baseado em funcionários, previamente descrito no Capítulo 2. Inicialmente, é introduzida a lógica de decomposição, fundamentada no esquema geral de Decomposição de Benders, proposta por Boysen *et al.* (2022), juntamente com os respectivos detalhes de implementação. Em seguida, descreve-se a heurística gulosa utilizada para a obtenção de soluções iniciais, também proposta pelos autores, e a meta-heurística GRASP, proposta neste trabalho. Por fim, discute-se o algoritmo de pré-processamento necessário para a aplicação da lógica de decomposição.

4.1 Lógica de decomposição

Conforme Boysen *et al.* (2022), o problema de associação entre demanda e oferta no contexto do *crowdshipping* baseado em funcionários pode ser formulado por meio de uma abordagem de decomposição fundamentada na Decomposição de Benders. Nessa formulação, o problema é dividido em dois níveis: o problema mestre e o subproblema (ou problema escravo). No problema mestre, as encomendas são atribuídas individualmente aos funcionários. Já no subproblema, verifica-se, para cada funcionário, se a atribuição realizada resulta em uma rota viável, considerando as restrições de desvio máximo permitido e de ganho mínimo esperado. Caso a atribuição não seja viável, é gerado um corte de viabilidade. Os dois problemas são resolvidos iterativamente e a melhor solução encontrada é ótima.

4.1.1 Problema mestre

O objetivo do problema mestre consiste em atribuir os pacotes aos funcionários, sem, contudo, determinar exatamente as rotas de entrega correspondentes. A seguir, apresenta-se formalmente a formulação desse problema:

$$\text{[Master] Maximize} \quad (1) \tag{13}$$

$$\text{Sujeito a} \quad (2), (3), (10), \text{ e}$$

$$\sum_{j \in J'} x_{ij} \leq k - 1 \qquad \forall i \in I,\ 1 \leq k \leq K,\ J' \in \overline{J}_i^k \tag{14}$$

$$\sum_{j \in J'} (1 - x_{ij}) + \sum_{j \in J \setminus J'} x_{ij} \geq 1 \qquad \forall i \in I,\ 1 \leq k \leq K,\ J' \in \underline{J}_i^k \tag{15}$$

A função objetivo (13) e as restrições (2), (3) e (10) continuam inalteradas do modelo MIP. Já as restrições (14) e (15) são inequações válidas geradas.

Considere que $\tilde{d}_i(\tilde{J})$ seja o caminho mínimo entre o depósito e a casa do funcionário i, visitando todos os endereços associados aos pacotes no conjunto $\tilde{J}$. Desta forma, podemos computar os conjuntos de inequações válidas da seguinte forma:

- $\overline{J}_i^k = \{J' \subseteq J : |J'| = k \wedge [\sum_{j \in J'} q_j > Q_i \vee \tilde{d}_i(J') > c_i^{\max} + c_{0i}]\}$: conjunto de clientes que não podem ser atendidos pelo funcionário i sem que este exceda a capacidade do veículo ou o desvio máximo aceito;
- $\underline{J}_i^k = \{J' \subseteq J : |J'| = k \wedge J' \notin \overline{J}_i^k \wedge k \cdot F < (\tilde{d}_i(J') - c_{0i}) \cdot f_i\}$: conjunto de clientes que não podem estar na mesma rota do funcionário i porque os ganhos seriam inferiores ao mínimo esperado.

As restrições (14) e (15) são necessárias para que o problema mestre não realize uma atribuição inviável ao funcionário i. Observe que uma violação por capacidade ou desvio máximo só pode ser resolvida removendo pacotes da rota de entrega (restrições (14)); similarmente, uma violação de ganhos abaixo do esperado talvez possa ser resolvida adicionando pacotes à rota de entrega (restrições (15)).

No entanto, embora essas restrições orientem o problema mestre a não realizar atribuições inviáveis, gerar todas a inequações possíveis acarretaria em um aumento expressivo no número de restrições, tornando o problema computacionalmente inviável para instâncias de grande porte. No pior cenário, todas as combinações possíveis de pacotes devem ser consideradas para cada funcionário i, correspondendo ao conjunto das partes (conjunto potência) de J.

Portanto, os conjuntos de restrições são limitados ao parâmetro K: quanto maior for K, mais pacotes serão considerados em $\overline{J}_i^k$ e $\underline{J}_i^k$. Quando $K = 0$, nenhuma inequação válida é gerada, fazendo com que o problema mestre realize as atribuições sem qualquer orientação inicial. De forma oposta, quando $K = n$, $O(m \cdot n \cdot 2^n)$ inequações válidas são geradas, no qual m e n representam o número de funcionários e pacotes, respectivamente. Experimentos realizados por Boysen *et al.* (2022) sugerem $K = 4$ um bom valor, equilibrando esforço de pré-processamento e velocidade de convergência do método.

Observe que, para o cálculo de $\tilde{d}_i(J')$, é necessário, para cada funcionário $i \in I$ e para cada $\tilde{J} \subseteq J$, resolver um problema de caminho mínimo envolvendo $O(K)$ nós. A Subseção 4.1.3 apresenta o método empregado para a resolução desse problema.

O resolvedor SCIP é empregado na resolução do problema mestre. Sempre que o processo de Branch-and-Cut encontra uma solução inteira, esta é passada para o subproblema. No subproblema, um ou mais cortes são gerados e adicionados ao problema mestre. Diferentemente, da abordagem clássica de Decomposição de Benders (Benders, 1962), o problema mestre não é resolvido do zero sempre que um corte é adicionado. Ao invés disso, os cortes são adicionados à árvore de enumeração do problema mestre como *lazy constraints*. Rahmaniani *et al.* (2017) referem-se a esta abordagem como Branch-and-Benders-Cut.

4.1.2 Subproblema

Quando uma solução inteira e viável $\bar{x}$ é encontrada no problema mestre, ela é encaminhada ao subproblema para verificar a viabilidade das rotas atribuídas a cada funcionário i. Caso $K \neq n$, a solução proposta pelo problema mestre pode se mostrar inviável para determinado funcionário, uma vez que o problema mestre garante viabilidade apenas para viagens com até K pacotes.

Considere que $R_i = \{j \in J \mid \bar{x}_{ij} = 1\}$ seja o conjunto de pacotes atribuídos ao funcionário i em determinada solução do problema mestre. Para checar se R_i é viável, precisamos encontrar uma permutação π_i de R_i de tal forma que $g(\pi_i)$ seja mínimo, isto é, precisamos encontrar a menor rota entre o depósito e a casa do funcionário i onde todas os pacotes em R_i sejam entregues. Este problema é NP-difícil, uma vez que se o endereço da casa do funcionário for o mesmo do depósito e $R_i = J$, encontrar o caminho mínimo é equivalente a resolver o Problema do Cacheiro Viajante (TSP) (Boysen et al., 2022).

Para a resolução do subproblema, constrói-se um grafo orientado $G(V, E, f)$, em que o conjunto de vértices é dado por $V = R_i \cup \{0, |R_i| + 1\}$, o conjunto de arestas por $E = V \times V \setminus \{(i,i) : i \in V\}$, e função de pesos $f : E \to \mathbb{R}^+$, representa o tempo de deslocamento entre pares de vértices (ver Figura 2.1). A solução

do Problema do Cacheiro Viajante (TSP) nesse grafo fornece a solução para o subproblema.

É utilizado o resolvedor SCIP na implementação do TSP, cuja formulação é expressa como um Problema Linear Inteiro (PLI). As restrições de eliminação de subrotas são incorporadas de forma iterativa, à medida que o método de Branch-and-Cut identifica soluções inteiras viáveis. Mais detalhes da implementação deste problema são fornecidos na Subseção 4.1.3.

Considere que f^* seja a duração do caminho mínimo fornecido pelo problema do TSP no grafo G, ou seja, o tempo total da rota do funcionário i. Se $f^* > c_i^{\max} + c_{0i}$ ou $f_i > \frac{|R_i| \cdot F}{f^* - c_{0i}}$, então a atribuição feita pelo problema mestre ao funcionário i é inviável porque ou desvio é muito longo ou os ganhos estão abaixo do esperado. Desta forma, um dos seguintes cortes de inviabilidade é adicionado à árvore de enumeração do problema mestre:

$$\sum_{j \in R_i} (1 - x_{ij}) \geq 1, \quad \text{se } f^* > c_i^{\max} + c_{0i} \tag{16}$$

$$\sum_{j \in R_i} (1 - x_{ij}) + \sum_{j \in \setminus R_i} x_{ij} \geq 1, \quad \text{se } \frac{|R_i| \cdot F}{f^* - c_{0i}} < f_i \wedge f^* \leq c_i^{\max} + c_{0i} \tag{17}$$

forçando o problema mestre a reatribuir pelo menos um dos pacotes atribuídos ao funcionário i.

Este procedimento é aplicado para todo funcionário $i \in I$. Caso nenhum corte de inviabilidade seja gerado, a atribuição feita pelo problema mestre é viável e pode ser ótima. Portanto, ela é armazenada e o seguinte corte de otimalidade é adicionado ao problema mestre:

$$\sum_{i \in I} \sum_{j \in J} x_{ij} \geq \sum_{i \in I} \sum_{j \in J} \bar{x}_{ij} + 1 \tag{18}$$

Esse processo é repetido até que não existam mais soluções viáveis a serem exploradas ou verificadas. Ao término da execução, a melhor solução identificada pelo método é considerada ótima.

4.1.3 Problema do caminho mínimo

Para que o subproblema seja resolvido, é necessário encontrar o caminho mínimo que visita todos os endereços dos pacotes associados a cada funcionário i em determinada solução do problema mestre. Conforme Boysen *et al.* (2022), esse problema é equivalente ao Problema do Cacheiro Viajante (TSP) em um grafo completo e orientado $G(V, E, f)$, onde o conjunto de vértices é dado por $V = R_i \cup \{0, |R_i| + 1\}$, o conjunto de arestas por $E = V \times V$, e a função de pesos

$f : E \to \mathbb{R}^+$ representa o tempo de deslocamento entre pares de vértices. Para a construção deste grafo, a seguinte função de pesos é utilizada:

$$f((j,j')) = \begin{cases} c_{0j'} & \text{se } j = 0 \wedge j' \in R_i \\ c_{jj'} & \text{se } j \in R_i \wedge j' \in R_i \\ c_{ji} & \text{se } j \in R_i \wedge j' = |R_i| + 1 \\ 0 & \text{se } j = |R_i| + 1 \wedge j' = 0 \\ \infty & \text{caso contrário} \end{cases}$$

Observe que a aresta $(|R_i| + 1, 0)$ está sempre presente na solução, o que assegura que o caminho mínimo até a residência do funcionário i, após a realização de todas as entregas, corresponda ao valor da função objetivo do problema. Adicionalmente, a atribuição de custo infinito às arestas que não são coerentes no contexto do problema impede que essas sejam alternativas consideradas na solução (por exemplo, trajetos de entregas que iniciem pela casa do funcionário).

O problema do TSP é modelado como um Problema Linear Inteiro (PLI) e solucionado por meio do resolvedor SCIP. A modelagem empregada segue a formulação apresentada por Wolsey (2020), incorporando adaptações específicas para o contexto do problema. A seguir, apresenta-se a formulação completa do modelo.

$$\text{[TSP] Minimize} \quad \sum_{(j,j') \in E} f((j,j')) \cdot y_{jj'} \tag{19}$$

$$\text{Sujeito a} \quad \sum_{j' \in V, j' \neq j} y_{jj'} = 1 \qquad \forall j \in V \tag{20}$$

$$\sum_{j \in V, j \neq j'} y_{jj'} = 1 \qquad \forall j' \in V \tag{21}$$

$$\sum_{j \in S} \sum_{j' \in S, j' \neq j} y_{jj'} \leq |S| - 1 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 1 \tag{22}$$

$$y_{jj'} \in \{0,1\} \qquad \forall (j,j') \in E \tag{23}$$

A função objetivo (19) minimiza o custo total do percurso, na qual $y_{jj'}$ é uma variável binária que indica se a aresta (j,j') faz parte do caminho mínimo. As restrições (20) e (21) garantem que cada vértice seja visitado exatamente uma vez. As restrições (22) são as restrições de eliminação de subrotas, que evitam a formação de ciclos menores que incluam apenas um subconjunto dos vértices. Essas restrições são adicionadas de forma dinâmica durante o processo de resolução, sempre que uma solução inteira que contenha subrotas é encontrada.

4.2 Heurísticas

Para a obtenção de soluções iniciais para o problema, são utilizadas duas heurísticas distintas: uma heurística gulosa, proposta por Boysen *et al.* (2022), e uma meta-heurística GRASP, desenvolvida neste trabalho. Ambas as heurísticas buscam construir soluções viáveis para o problema de *crowdshipping* baseado em funcionários, estabelecendo um limitante inferior que servirá como ponto de partida para o método exato de resolução.

Esta seção detalha os procedimentos e estratégias adotadas em cada uma dessas heurísticas, destacando suas características principais, critérios de seleção e mecanismos de construção das soluções. A Subseção 4.2.1 aborda a heurística gulosa, enquanto a Subseção 4.2.2 explora a meta-heurística GRASP.

4.2.1 Heurística gulosa

A heurística gulosa proposta por Boysen *et al.* (2022) obtém uma solução inicial para o problema por meio de um processo iterativo que busca atribuir pacotes aos funcionários sempre que possível, respeitando as restrições operacionais do problema. Para isso, o trajeto de cada funcionário $i \in I$ é inicializado como $\pi_i := \emptyset$. Começando do funcionário $i = 1$ até $i = m$, o procedimento busca checar se cada pacote $j \in J$ pode ser inserido na rota de entrega do funcionário i. Uma vez que todos os pacotes são avaliados para o funcionário i, o processo é repetido para o próximo funcionário, até que todos os funcionários tenham sido considerados. Para que um pacote $j \in J$ seja inserido na rota de entrega do funcionário i ($\bar{\pi} := \pi_i \cup \{j\}$), as seguintes condições devem ser satisfeitas:

- A capacidade do veículo do funcionário i não deve ser excedida com a adição do pacote j, ou seja, $\sum_{j \in \bar{\pi}} q_j \leq Q_i$;
- O desvio total da rota do funcionário i, considerando a adição do pacote j, não deve ultrapassar o desvio máximo permitido, ou seja, $g(\bar{\pi}) \leq c_i^{\max} + c_{0i}$;
- O ganho total esperado com a entrega dos pacotes na rota do funcionário i, incluindo o pacote j, deve ser maior ou igual ao ganho mínimo esperado, ou seja, $\frac{|\bar{\pi}| \cdot F}{g(\bar{\pi}) - c_{0i}} \geq f_i$.

Caso todas as condições sejam satisfeitas, o pacote j é adicionado ao fim da rota de entrega do funcionário i ($\pi_i := \bar{\pi}$) e removido do conjunto de pacotes a serem entregues ($J := J \setminus \{j\}$).

Após o procedimento ser aplicado para todos os funcionários, uma solução viável inicial é obtida, o que fornece um limitante inferior (LB) para o problema. Portanto, a seguinte restrição é adicionada ao problema mestre:

$$\sum_{i \in I} \sum_{j \in J} x_{ij} \geq \text{LB} + 1 \tag{24}$$

4.2.2 Meta-heurística GRASP

A meta-heurística GRASP (Greedy Randomized Adaptive Search Procedure) desenvolvida neste trabalho busca construir soluções iniciais para o problema de *crowdshipping* baseado em funcionários por meio de um processo iterativo que combina elementos de construção gulosa e aleatoriedade.

O procedimento inicia-se em uma fase de construção, na qual uma lista de pacotes candidatos a cada funcionário $i \in I$ é gerada. Essa lista é composta pelos pacotes que podem ser atribuídos ao funcionário i sem violar as restrições operacionais do problema, ou seja, pacotes que satisfazem as condições de capacidade do veículo, desvio máximo permitido e ganho mínimo esperado, conforme descrito na Subseção 4.2.1. Para cada candidato pertencente a essa lista, calcula-se um *score* por meio de uma função gulosa, a qual expressa o benefício da atribuição do candidato $j \in J$ ao funcionário $i \in I$. A função de *score* adotada é definida da seguinte forma:

$$\text{score}(i, j) = \frac{f_i}{F_i} + (1 - \frac{L_i + w(j)}{C_i}) + (1 - \frac{\Delta t_i(j)}{\Delta_i}) \tag{25}$$

O score é composto por três termos que avaliam diferentes aspectos da atribuição do pacote j ao funcionário i, sendo eles $\frac{f_i}{F_i}$, que representa a razão entre o ganho mínimo esperado e os ganhos já obtidos pelo funcionário i; $(1 - \frac{L_i + w(j)}{C_i})$, que avalia a utilização da capacidade do veículo do funcionário i com a adição do pacote j; e $(1 - \frac{\Delta t_i(j)}{\Delta_i})$, que mede o impacto do tempo adicional necessário para entregar o pacote j em relação ao desvio máximo permitido para o funcionário i.

O valor de $\Delta t_i(j)$ representa o aumento do tempo de viagem do funcionário i ao incluir o pacote j em sua rota de entrega. Para calcular esse valor, considera-se a rota atual do funcionário i e determina-se o incremento no tempo de viagem resultante da inserção do pacote j na melhor posição possível dentro dessa rota.

Cada termo é normalizado para garantir que todos contribuam de maneira equilibrada para o score final. Observe que F_i, L_i e Δ_i são atualizados a cada atribuição, refletindo a caracteristica adaptativa do método GRASP.

Cada termo é normalizado para garantir que todos contribuam de maneira equilibrada para o score final. Observe que F_i, L_i e Δ_i são atualizados a cada atribuição, refletindo a caracteristica adaptativa do método GRASP.

Após o cálculo do *score* para cada pacote $j \in J$, é construído uma lista restrita de candidatos contendo aqueles com melhores valores. O parâme-

tro $\alpha \in [0,1]$ é utilizado para controlar o equilíbrio entre aleatoriedade e caráter guloso no processo de seleção dos pacotes. Nos experimentos realizados, utilizou-se $\alpha = 0.1$, cuja escolha é discutida em detalhe no Capítulo 5. A partir dessa lista restrita, os pacotes são selecionados aleatoriamente e atribuídos ao funcionário i. Esse procedimento é repetido até que não restem pacotes candidatos a serem atribuídos a qualquer funcionário.

Com a solução obtida na etapa de construção, inicia-se uma busca local em sua vizinhança. Essa busca consiste verificar a possibilidade de atribuir pacotes ainda não alocados, avaliando se sua inclusão mantém a solução viável e produz uma melhoria no valor da função objetivo. É importante observar que determinados candidatos podem ter sido descartados na fase de construção por não apresentarem ganhos mínimos individuais, mas podem tornar-se viáveis quando considerados em conjunto com outros pacotes já atribuídos. A busca local prossegue até que não seja possível identificar novas melhorias.

O tempo de execução do GRASP é determinado pelas fases de construção e busca local. Assim, sua complexidade pode ser expressa como

$$O(T \cdot (C(n) + L(n)))$$

em que $T = \text{maxitr}$ corresponde ao número máximo de iterações. Na fase de construção, no pior caso, considera-se uma lista de candidatos C de tamanho n. A cada seleção de um candidato, essa lista é reduzida para $|C| - 1$. Além disso, para cada candidato selecionado, é necessário avaliar o melhor ponto de inserção na rota do funcionário i, o que demanda $O(n)$ operações. Sendo realizadas n seleções e, para cada uma, um custo adicional proporcional ao tamanho da lista remanescente, obtém-se uma complexidade assintótica de $C(n) = O(n^3)$ para esta etapa. A etapa de construção, no pior caso, equivale à etapa de construção, uma vez que esta etapa consiste na tentativa de inserção de pacotes ainda não alocados. Portanto, obtém-se uma complexidade assintótica $O(T \cdot (n^3 + n^3) = O(n^3)$ para todo o procedimento do GRASP.

Os procedimentos de construção e de busca local são executados 100 vezes, e a melhor solução obtida ao final dessas iterações é utilizada para gerar a restrição de limitante inferior (24) idêntica à descrita na Subseção 4.2.1.

4.3 Pré-processamento

Antes da aplicação do método de decomposição, é necessário realizar um pré-processamento a fim de calcular os conjuntos de inequações válidas $\overline{J}_i^k$ e $\underline{J}_i^k$ para cada funcionário $i \in I$ e para cada $1 \leq k \leq K$. Conforme discutido na Subseção 4.1.3, o cálculo desses conjuntos requer a resolução do problema do caminho mínimo para todas as combinações possíveis de pacotes de tamanho

até K. Para isso, uma instância do problema do TSP, conforme descrito na Subseção 4.1.3, é construída e resolvida para cada combinação de pacotes $J' \subseteq J$ e para cada funcionário $i \in I$.

O algoritmo inicia com a geração de todas as combinações possíveis de pacotes de tamanho k, para k começando em 1 e variando até K. Para cada combinação gerada, as restrições de capacidade, desvio máximo e ganho mínimo são avaliadas. Caso a combinação viole a restrição de capacidade ou desvio máximo, ela é adicionada ao conjunto $\overline{J}_i^k$. Caso contrário, se a combinação não violar essas restrições, mas não atender à restrição de ganho mínimo, ela é adicionada ao conjunto $\underline{J}_i^k$.

É importante notar que, caso uma combinação possua um subconjunto que já tenha sido identificado como inviável por violar as restrições de capacidade ou desvio máximo, essa combinação pode ser automaticamente classificada como inviável sem a necessidade de resolver o problema do TSP. Isso ocorre porque a adição de mais pacotes a uma rota que já é inviável não pode resultar em uma rota viável. Essa otimização reduz significativamente o número de instâncias do problema do TSP que precisam ser resolvidas, melhorando a eficiência do pré-processamento. Para agilizar a verificação desses subconjuntos inviáveis, empregam-se técnicas de bitmasking e estruturas de tabela hash.

Além disso, o algoritmo utiliza técnicas de paralelização para distribuir a carga de trabalho entre múltiplas hyper-threads, aproveitando os recursos computacionais disponíveis. O trabalho é dividido de forma que cada thread seja responsável por processar os subconjuntos dos funcionários atribuídos a ela. A distribuição de funcionários entre as threads é feita de maneira balanceada, garantindo que cada thread tenha uma carga de trabalho semelhante.

Por fim, ao término do pré-processamento, os conjuntos de inequações válidas $\overline{J}_i^k$ e $\underline{J}_i^k$ estão prontos para serem utilizados na formulação do problema mestre, conforme descrito na Subseção 4.1.1.

CAPÍTULO

5

Resultados Computacionais

Este capítulo apresenta os resultados computacionais obtidos a partir da implementação, execução e análise dos algoritmos propostos neste trabalho. Os experimentos foram conduzidos em um ambiente controlado, utilizando uma máquina com um processador x86_64 AMD Ryzen 7 5700G de 16 hyper-threads e 3,8 GHz de clock, 16 GB de memória RAM e sistema operacional Ubuntu 24.04 LTS. Todos os procedimentos foram implementados utilizando a linguagem C, com o compilador GCC na versão 13.3.0. O SCIP (versão 8.0.1) foi utilizado como resolvedor padrão. Foi adotado um limite de tempo de 900 segundos para a resolução de cada instância.

A organização deste capítulo se dá da seguinte forma: na Seção 5.1 são descritas as instâncias utilizadas nos experimentos. A Seção 5.2 detalha o processo de ajuste dos parâmetros do algoritmo GRASP. A Seção 5.3 apresenta e discute os resultados obtidos a partir dos experimentos computacionais realizados.

5.1 Instâncias

As instâncias empregadas nos experimentos foram disponibilizadas por Boysen *et al.* (2022). O conjunto é composto por 12 categorias de instâncias, cujas dimensões variam de 10 a 100 pacotes e de 5 a 90 funcionários. Para cada categoria, são fornecidas 10 instâncias distintas, totalizando 120 instâncias utilizadas nas análises. Cada instância possui informações acerca das localizações geográficas dos pacotes e dos funcionários, bem como o volume que cada pacote ocupa. Para cada funcionário, são especificados a capacidade máxima do veículo, o desvio máximo aceito e os ganhos mínimos esperados.

Conforme descrito pelos autores, as instâncias foram geradas seguindo as configurações descritas a seguir:

- Posições geográficas: as localizações dos pacotes e dos funcionários são distribuídas aleatoriamente ao redor de clusters centrais, em um plano cartesiano bi-dimensional com um sistema de coordenadas $(x,y) \in (0,1) \times (0,1)$. O parâmetro de densidade λ controla a dispersão dos pontos em relação aos centros dos clusters. Para os experimentos, foi utilizado $\lambda = 0,7$. O número cl de clusters centrais é definido, de forma aleatória, de um intervalo $\mathcal{U}\{5,\ldots,10\}$. Cada cluster central é posicionado em uma localização $(\tilde{x}_k, \tilde{y}_k)$, $k = 1,...,cl$, onde $\tilde{x}_k$ e $\tilde{y}_k$ são valores gerados aleatoriamente no intervalo $[0,1]$. Além disso, a localização do centro de distribuição é fixado ao centro do plano cartesiano, ou seja, na posição $(0,5;0,5)$. Para obter qualquer distância entre duas localizações, é utilizada a métrica Euclidiana, na qual $c_{jj'} = T \cdot \sqrt{(\bar{x}_j - \bar{x}_{j'})^2 + (\bar{y}_j - \bar{y}_{j'})^2} + s$, $c_{0j} = T \cdot \sqrt{(x^d - \bar{x}_j)^2 + (y^d - \bar{y}_j)^2} + s$, $c_{ji} = T \cdot \sqrt{(\bar{x}_j - x_i)^2 + (\bar{y}_j - y_i)^2}$ e $c_{0i} = T \cdot \sqrt{(x^d - x_i)^2 + (y^d - y_i)^2}$, com $T = 180 \cdot (1,5 - \lambda)$, tempo de entrega da encomenda ao funcionário $s = 3$ minutos e 180 minutos como o tempo máximo de deslocamento entre o centro de distribuição e a localização mais distante.

- Pacotes: os pacotes possuem três volumes distíntos, classificados como pequeno ($q_j = 1$), médio ($q_j = 5$) e grande ($q_j = 10$). A distribuição dos volumes segue a proporção de 50%, 30% e 20%, respectivamente. Cada pacote rende um ganho fixo de $F = 2$ unidades monetárias.

- Funcionários: a capacidade Q_i de cada funcionário $i \in I$ é um número inteiro obtido de uma distribuição normal com média $\mu = 15$ e desvio padrão $\sigma = 5$, com limites inferior e superior de 5 e 25, respectivamente. O desvio máximo $c_i^{\max}$ aceito por cada funcionário é definido como $c_i^{\max} = \min\{0,33, \max\{\alpha_i, 0,05\}\} \cdot c_{0i}$, no qual α_i é um número aleatório gerado a partir de uma distribuição normal com média $\mu = 0,2$ e desvio padrão $\sigma = 0,15$, limitado ao intervalo $[0,05, 0,33]$. Por fim, o ganho mínimo esperado por cada funcionário é obtido através de uma distribuição normal com média $\mu = 12$ e desvio padrão $\sigma = 2$, limitado inferiormente a 5 unidades monetárias.

5.2 Ajuste de Parâmetros do GRASP

A meta-heurística GRASP, proposta neste trabalho (Subseção 4.2.2), possui dois parâmetros principais que influenciam diretamente o seu desempenho,

sendo eles o número máximo de interações *maxitr* e o fator de aleatoriedade α. Para determinar os valores mais adequados para esses parâmetros, foi realizado um estudo preliminar utilizando um subconjunto das instâncias descritas na Seção 5.1. Para compor este subconjunto, foram selecionadas uma instância de cada tamanho disponível, totalizando 12 instâncias distintas.

O estudo foi conzido em duas etapas. Na primeira, os parâmetros *maxitr* e α foram variavelmente combinados a partir de conjuntos previamente definidos. O parâmetro α assumiu os valores de $\{0.0, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$, enquanto *maxitr* foi configurado como $\{10, 50, 200, 1000\}$. Cada combinação de valores foi avaliada considerando dois aspectos principais: a qualidade da solução e o tempo computacional de execução. A qualidade da solução foi mensurada pelo valor da função objetivo obtido, ao passo que o tempo de execução foi registrado em segundos. Cada experimento foi repetido 10 vezes para cada instância, e os resultados finais foram obtidos através da média dos valores coletados.

Após a execução de todos os experimentos, os resultados foram analisados com o objetivo de identificar quais valores de α apresentaram o melhor desempenho em termos de qualidade da solução e eficiência temporal. A síntese dessas análises é apresentada na Figura 5.1.

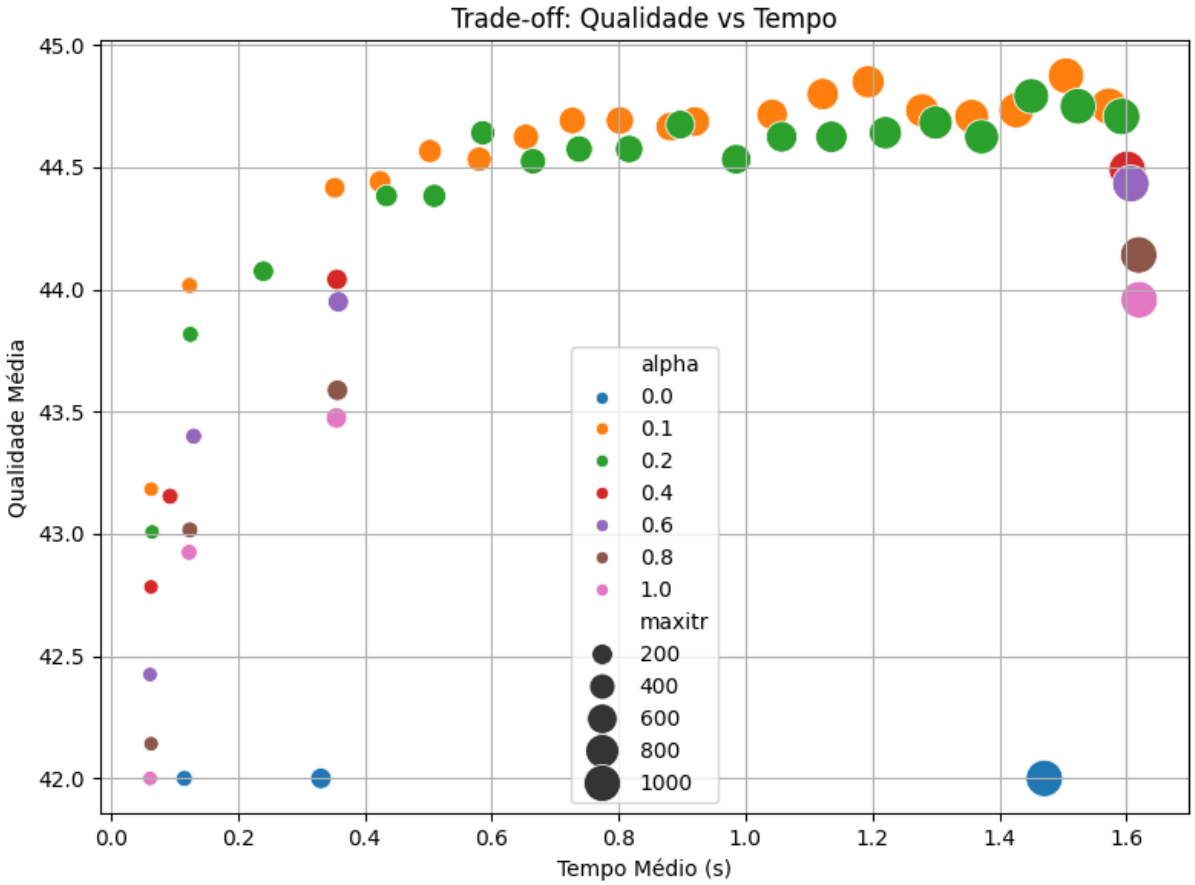

Figura 5.1: Resultados dos experimentos de parametrização do GRASP

Observa-se que alguns valores de *maxitr* e α podem coincidir graficamente, o que resulta na omissão de determinados pontos. Ademais, a legenda não inclui todos os valores distintos de maxitr, de modo a preservar a legibilidade da figura.

Com base nos resultados obtidos na primeira etapa, foi possível identificar que os valores de α entre $0,1$ e $0,2$ apresentaram soluções de melhor qualidade. A segunda etapa do estudo concentrou-se em refinar a escolha dos parâmetros, fixando α em $0,1$ e $0,2$, enquanto *maxitr* foi ajustado para valores mais específicos, variando em incrementos de 50 unidades, o que resultou no conjunto $\{200, 250, ..., 950, 1000\}$.

Como podemos observar na Figura 5.1, os melhores resultados foram alcançados com $\alpha = 0,1$ e $maxitr = 750$, proporcionando soluções de alta qualidade em um tempo de execução razoável. Portanto, esses valores foram adotados para os experimentos subsequentes apresentados na Seção 5.3.

5.3 Resultados Computacionais

Nesta seção, são apresentados os resultados computacionais obtidos a partir da aplicação dos métodos propostos. Especificamente, são comparados os desempenhos entre as duas heurísticas implementadas, a Heurística Gulosa (Subseção 4.2.1), proposta por Boysen *et al.* (2022) e o GRASP (Subseção 4.2.2), proposta neste trabalho, ambas seguidas pela aplicação do procedimento de Branch-and-Benders-Cut descrito na Seção 4.1.

n	**m**	**pre**	**Heurística Gulosa**				**GRASP**			
			gap_b (%)	**opt**	**obj**	**sec**	**gap_b (%)**	**opt**	**obj**	**sec**
10	5	< 0,1	3,70	9	2,5	< 0,1	0,00	10	2,7	< 0,1
20	10	0,1	21,30	3	5,7	< 0,1	2,34	8	7,3	< 0,1
30	15	0,28	22,49	0	10,6	< 0,1	1,18	9	13,7	< 0,1
40	15	0,74	22,74	1	12,8	< 0,1	2,95	6	16,3	< 0,1
50	20	1,22	28,29	0	17,3	< 0,1	3,24	4	23,4	0,2
80	30	9,35	30,11	0	32,6	< 0,1	8,97	0	42,7	0,5
100	40	89,07	31,40	0	46,2	< 0,1	13,51	0	58,9	1,1
100	50	91,11	28,71	0	53,0	< 0,1	12,45	0	65,3	1,5
100	60	139,91	24,35	0	59,7	< 0,1	8,98	0	71,9	1,8
100	70	178,42	23,35	0	66,0	< 0,1	12,02	0	75,8	2,3
100	80	190,63	19,42	0	66,9	< 0,1	7,11	0	77,4	2,7
100	90	204,22	15,46	0	73,4	< 0,1	6,67	0	81,6	3,3
	média	**75,43**	**22,61**	**1,1**	**37,2**	**< 0,1**	**6,62**	**3,1**	**44,8**	**1,1**

Tabela 5.1: Resultados computacionais das heurísticas

Os resultados apresentados na Tabela 5.1 sumarizam o desempenho das

heurísticas aplicadas ao nó raiz do procedimento de Branch-and-Benders-Cut. As colunas n e m identificam o número de pacotes e funcionários voluntários ao *crowdshipping*, respectivamente. A coluna de pré-processamento (pre) informa o tempo médio de execução, em segundos, do pré-processamento de cada instância. As colunas referentes à heurística gulosa e ao GRASP representam, respectivamente, o gap médio em porcentagem em relação ao melhor limitante superior encontrado (gap_b), o número de soluções ótimas encontradas (opt), o valor médio da função objetivo obtido (obj) e o tempo médio de execução em segundos (sec). O cálculo do gap é feito da seguinte forma:

$$\text{gap}_b = \frac{(\text{UB} - \text{Heur})}{\text{UB}}$$

No qual UB representa o melhor limitante superior encontrado e Heur representa o valor da solução encontrada pela heurística.

n	**m**	**Heurística Gulosa + BBC**				**GRASP + BBC**			
		gap (%)	**opt**	**obj**	**sec**	**gap (%)**	**opt**	**obj**	**sec**
10	5	0,00	10	2,7	< 0,1	0,00	10	2,7	< 0,1
20	10	0,00	10	7,5	< 0,1	0,00	10	7,5	< 0,1
30	15	0,00	10	13,9	< 0,1	0,00	10	13,9	0,1
40	15	0,00	10	16,8	0,5	0,00	10	16,8	0,4
50	20	0,00	10	24,2	2,2	0,00	10	24,2	2,0
80	30	1,73	7	45,9	346,0	2,20	7	45,9	332,4
100	40	4,88	5	64,2	607,3	5,93	3	64,0	719,4
100	50	4,39	3	71,2	801,0	4,42	3	74,4	707,1
100	60	1,58	6	77,7	409,1	1,62	5	77,7	500,7
100	70	3,90	3	82,7	703,0	4,28	3	82,5	708,0
100	80	1,62	5	81,7	516,7	2,09	5	81,6	569,0
100	90	0,56	6	86,4	583,6	1,30	5	86,3	612,9
	média	**1,56**	**7,1**	**47,9**	**330,8**	**1,82**	**6,8**	**48,1**	**346,0**

Tabela 5.2: Resultados computacionais do BBC no nó raiz após aplicação das heurísticas

A Tabela 5.2 apresenta os resultados obtidos após a aplicação das heurísticas no nó raiz do procedimento de Branch-and-Benders-Cut (em procedimentos separados). As colunas seguem a mesma estrutura da Tabela 5.1, com a modificação do *gap* médio em porcentagem que, neste caso, é em relação à solução ótima (gap). Os resultados reportados nos levam às seguintes considerações:

- A heurística GRASP claramente supera a Heurística Gulosa em termos de qualidade das soluções iniciais obtidas. Isso é evidenciado pelo menor *gap* médio obtido com o GRASP ($6,62\%$) em relação à Heurística Gulosa ($22,61\%$) conforme mostrado na Tabela 5.1. Além disso, o GRASP

demonstrou-se muito eficaz ao resolver quase que na otimalidade as instâncias menores (até 50 pacotes e 20 funcionários), alcançando um *gap* médio de $1{,}94\%$, nessas instâncias, contra $19{,}70\%$ da Heurística Gulosa. Em especial para as instâncias com 10 pacotes e 5 funcionários, no qual o GRASP conseguiu encontrar soluções ótimas em todas as 10 instâncias testadas. Em contra partida, seu tempo médio de execução é consideravelmente maior quando comparado à Heurística Gulosa, embora ainda seja baixo. No entanto, como as heurísticas são aplicadas apenas no nó raiz do procedimento de Branch-and-Benders-Cut, esse aumento no tempo de execução não impacta significativamente o tempo total de resolução das instâncias. Esses resultados consolidam a meta-heurística GRASP proposta como um método poderoso para a geração de soluções iniciais de alta qualidade com um baixo custo computacional.

- Por outro lado, quando analisamos os resultados de todo o procedimento de Branch-and-Benders-Cut com a aplicação de ambas as heurísticas no nó raiz (Tabela 5.2), observamos que os desempenhos são bastante similares, com uma leve vantagem para a Heurística Gulosa. Analisando a estrutura das soluções geradas por ambas as heurísticas, é possível notar que elas tendem a explorar diferentes regiões do espaço de soluções. Ao analisar as soluções geradas pelas duas heurísticas para a primeira instância do conjunto de instâncias com 80 pacotes e 30 funcionários, é possível observar que há uma interseção de apenas 3 variáveis iguais em ambas as soluções, com a Heurística Gulosa atribuindo 29 pacotes e o GRASP atribuindo 35. Considerando a distância de hamming entre as soluções, obtemos uma divergência aproximada de 90,6%, indicando que as soluções são substancialmente diferentes. Essa divergência entre as soluções reforça a ideia de que ambas as heurísticas exploram diferentes regiões do espaço de soluções, o que pode explicar a dificuldade em obter resultados melhores ao aplicar a meta-heurística GRASP no procedimento de Branch-and-Benders-Cut, uma vez que, com o GRASP gerando soluções iniciais mais próximas do ótimo, a diversidade das soluções exploradas diminui, limitando as informações que podem ser extraídas para aprimorar o processo de cortes no Branch-and-Benders-Cut. Portanto, uma possível direção para trabalhos futuros seria investigar estretégias para aumentar a convergência do método de Branch-and-Benders-Cut quando soluções iniciais de alta qualidade são fornecidas, potencialmente através da incorporação de técnicas que promovam a diversidade das soluções exploradas.

Desta forma, conclui-se que a meta-heurística GRASP proposta neste trabalho é eficaz na geração de soluções iniciais de alta qualidade para o pro-

blema de roteamento de pacotes via *crowdshipping* baseado em funcionários, superando significativamente a Heurística Gulosa em termos de qualidade das soluções obtidas. No entanto, quando integrada ao procedimento de Branch-and-Benders-Cut, ambas as heurísticas apresentam desempenhos similares, sugerindo que a diversidade das soluções geradas pode ser um fator crucial para o sucesso do método de Branch-and-Benders-Cut.

CAPÍTULO

6

Contribuições e Conclusões

Este trabalho teve como objetivo desenvolver, implementar e avaliar métodos exatos e heurísticos para o problema da entrega compartilhada baseado em funcionários (*employee-based crowdshipping*), no qual funcionários de um centro de distribuição entregam encomendas para clientes em suas vizinhanças, no caminho de volta para casa, conforme proposto por Boysen *et al.* (2022). As principais contribuições alcançadas incluem a implementação completa do método de Branch-and-Benders-Cut (BBC), a proposição de uma meta-heurística GRASP para a geração de soluções iniciais e o desenvolvimento de um módulo de pré-processamento eficiente para o cálculo das inequações válidas, utilizadas no problema mestre.

A meta-heurística GRASP desenvolvida mostrou-se eficaz na obtenção de soluções iniciais de alta qualidade, superando consistentemente a Heurística Gulosa. No entanto, observou-se que, quando utilizada como ponto de partida para o BBC, a melhoria na qualidade inicial não se traduz em um ganho significativo de desempenho, possivelmente devido à limitação na diversidade das soluções exploradas durante o processo de cortes. Ainda assim, o GRASP demonstrou ser uma ferramenta robusta para a obtenção de boas soluções em um tempo computacional reduzido, configurando-se como uma alternativa promissora para instâncias de grande escala, nas quais métodos estritamente exatos se tornam inviáveis ou excessivamente custosos.

O módulo de pré-processamento implementado demonstrou ser eficiente no cálculo dos conjuntos de inequações válidas, permitindo identificar combinações inviáveis rapidamente por meio de técnicas de programação paralela e estruturas de dados otimizadas.

Como direções para trabalhos futuros, destaca-se a necessidade de in-

vestigar de forma mais aprofundada o impacto da qualidade da solução inicial na eficácia dos cortes gerados pelo método BBC, especialmente quando o processo é iniciado a partir de soluções próximas do ótimo. Além disso, recomenda-se a exploração de estratégias alternativas de busca local, para o GRASP, que promovam maior diversidade nas soluções avaliadas, de modo potencializar o desempenho global da abordagem.

Referências Bibliográficas

Alvarez, A., Cordeau, J.-F., e Jans, R. (2024). The consistent vehicle routing problem with stochastic customers and demands. *Transportation Research Part B: Methodological*, 186:102968. Citado na página 7.

Behrend, M. e Meisel, F. (2018). The integration of item-sharing and crowdshipping: Can collaborative consumption be pushed by delivering through the crowd? *Transportation Research Part B: Methodological*, 111:227–243. Citado na página 4.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252. Citado na página 20.

Botsman, R. e Foley, K. (2010). *What's mine is yours*. Tantor Media, Incorporated. Citado na página 4.

Boysen, N., Emde, S., e Schwerdfeger, S. (2022). Crowdshipping by employees of distribution centers: Optimization approaches for matching supply and demand. *European Journal of Operational Research*, 296(2):539–556. Citado nas páginas 4, 5, 6, 9, 14, 23, 25, 26, 28, 33, 36, e 41.

Boysen, N., Fedtke, S., e Schwerdfeger, S. (2021). Last-mile delivery concepts: a survey from an operational research perspective. *Or Spectrum*, 43(1):1–58. Citado na página 3.

Dantzig, G. B. (1951). Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347. Citado na página 18.

Díaz-Ríos, D. e Salazar-González, J.-J. (2024). Mathematical formulations for consistent travelling salesman problems. *European Journal of Operational Research*, 313(2):465–477. Citado na página 7.

Feo, T. A. e Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133. Citado na página 21.

Land, A. H. e Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520. Citado na página 19.

Rahmaniani, R., Crainic, T. G., Gendreau, M., e Rei, W. (2017). The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817. Citado nas páginas 6 e 20.

Resende, M. G. e Ribeiro, C. (1998). Greedy randomized adaptive search procedures (grasp). *AT&T Labs Research Technical Report*, 98(1):1–11. Citado na página 21.

Statista (2025). E-commerce outlook worldwide. https://www.statista.com/outlook/emo/ecommerce/worldwide. Accessed: 2025-11-21. Citado na página 3.

Wolsey, L. A. (2020). *Integer programming*. John Wiley & Sons. Citado nas páginas 17, 18, 19, 20, e 27.