Modelagem e Desenvolvimento de Algoritmo para o Problema de Roteamento Dinâmico de Veículos

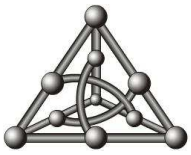
Wilton Gustavo Gomes da Costa

Modelagem e Desenvolvimento de Algoritmo para o Problema de Roteamento Dinâmico de Veículos[1]

Wilton Gustavo Gomes da Costa

Orientador: *Prof. Dr. Ricardo Ribeiro dos Santos*
Coorientador: *Prof. Dr. Willy Alves de Oliveira Soler*

Dissertação apresentada à Faculdade de Computação (FACOM) da Universidade Federal de Mato Grosso do Sul (UFMS) como parte dos requisitos necessários à obtenção do título de Mestre em Ciências de Computação.



FACOM
UFMS - Universidade Federal de Mato Grosso do Sul
Agosto/2022

[1]Este trabalho recebeu auxílio financeiro do CNPq sob processo: 133954/2020-0.

Aos meus pais,
Nilton e Wilsiene,

À minha irmã,
Nilene,

À Rhayssa e

Aos meus avós.

Agradecimentos

Entre grandes tropeços da vida, Deus a iluminou com muita compaixão, e diante de grandes desafios me fez questionar o grande propósito dela, com muito esforço, dedicação e a coragem de errar, lapidei meu conhecimento, e hoje sei o quanto tenho a aprender em aprender.

Gostaria de agradecer ao meu pai Doutor Nilton César Antunes da Costa e minha mãe Wilsiene Ramos Gomes da Costa, que mantiveram a luta constante em me dar todo suporte, saúde e educação com qualquer coisa que precisei antes mesmo de eu nascer.

À minha irmã Nilene Ramos Gomes da Costa que é uma luz em meu caminho, orientando minha mente, mesmo estando distante.

À minha amada Mestre Rhayssa Sonohata, que dedicou seu tempo em me apoiar, encorajar, e fazer com que eu enfrentasse as decisões mais difíceis, com amor, sabedoria, respeito e carinho.

Ao professor Doutor Ricardo Ribeiro dos Santos, que desde o começo do mestrado apoiou minhas decisões e me trouxe todo conhecimento necessário para progredir diante de situações inexploradas, além de comemorar as vitórias e me fazer questionar o porquê das derrotas e aprimorá-las para uma melhor decisão.

Ao professor Doutor Willy Alves Soler, que me apresentou um mundo da matemática que até então era desconhecida aos meus olhos, e sempre manteve a seriedade em acompanhar a evolução dos meus estudos.

E aos meus avós Iris e Wilseu, que me apoiam inteiramente para qualquer decisão tomada, e me cuidam com muito zelo. E aos meus avós Nilda e Miguel, que hoje brilham no céu, mas que estarão sempre em meu coração, lembro como se fosse ontem dele me dizendo: “Vô, nessa vida o bem mais precioso é o conhecimento, dele ninguém te tira”.

A professora Doutora Bianca, que se apoiou em correções pertinentes deste trabalho e se demonstrou interessada em meu trabalho, dando uma motivação maior a ele. E a professora Doutora Edna, que me permitiu estar presente

em suas aulas online e que serviram de grande auxílio para aprimorar técnicas já desenvolvidas, assim como aconselhar os caminhos que não seriam interessantes seguir.

E cordialmente agradeço a CNPq que realizou um investimento (sob o processo 133954/2020-0) neste trabalho, essencial para o desenvolvimento do mesmo. Cada passo nessa jornada, foi ímpar e fico lisonjeado por todo conhecimento adquirido nessa oportunidade.

Resumo

Encontrar rotas eficientes para uma frota de forma a minimizar a distância percorrida e o tempo de viagem e maximizar o lucro do serviço são alguns objetivos almejados na resolução do Problema de Roteamento de Veículos (PRV). O PRV e suas variantes são amplamente estudados na literatura técnica especializada, com diversas propostas de modelos, algoritmos e técnicas (métodos) de resolução. Neste trabalho de mestrado, o objetivo é resolver a variante do PRV, denominada Problema de Roteamento Dinâmico de Veículos (PRDV). O PRDV considera que os itens a serem entregues não são conhecidos *a priori* e podem aparecer para o roteamento de maneira dinâmica. Este é um problema atual e de interesse das empresas de logística, especialmente aquelas com enfoque em *marketplace*, que precisam lidar com milhares de itens de produtos para entregas ao longo do dia e possuem limitações de frotas de veículos e de horários para entrega. Neste trabalho foram desenvolvidos um algoritmo dinâmico, denominado *Dynamic Search per Neighbors Routes* (DSNR), e um algoritmo estático, denominado *Kmeans, Relax-and-Fix and Optimizations* (K-RFO). O cenário para o problema consiste em explorar o roteamento dinâmico a partir de lotes de pacotes para serem entregues em uma jornada de trabalho do mesmo dia. A técnica implementada no algoritmo DSNR é baseada em busca local associada a uma implementação de uma heurística denominada 2-Opt**, visando re-otimizar rotas vizinhas. Quando comparada com os algoritmos dinâmicos *QRP-Sweep* (QRPS) e K*means-Greedy* (KG), disponibilizados no repositório Loggibud, observaram-se economias de 17% nos custos de transporte e operacionais, ao utilizar a técnica DSNR.

Abstract

Finding efficient routes for a vehicle fleet to minimize distance and time and maximizing service profit are some of the goals pursued in solving the *Vehicle Routing Problem* (VRP). The VRP and its variants are widely studied in the area, with several proposals for models, algorithms, and techniques (methods). The goal of this work is to present an approach named *Dynamic Vehicle Routing Problem* (DVRP). In the DVRP the set of items to be delivered are not known in advance. This is a current problem targeted to logistics companies, especially those whose focus is on *marketplace*, that should manage thousands of products to be delivered along a working day subject to constraints on vehicle fleets and service hours. In this work, dynamic and static routing algorithms, named *Dynamic Search per Neighbors Routes* (DSNR) and *Kmeans, Relax-and-Fix and Optimizations* (K-RFO), respectively, are proposed. The scenery for the dynamic algorithm is the routing of batches of packages (items) to be delivered at the same working day. The DVRP algorithm is based on a local search together with a 2-Opt** heuristic aiming to re-optimize neighboring routes to accommodate dynamic packages. The DSNR algorithm has been evaluated and compared to dynamic algorithms *QRP-Sweep* (QRPS) and K*means-Greedy* (KG), achieving up to 17% of operational costs savings.

Sumário

Lista de Figuras

Lista de Tabelas

Siglas

ACO *Ant Colony Optimization*. 13

API *Application Program Interface*. 38

DSNR *Dynamic Search per Neighbors Routes*. ix, xi, 23, 42–56, 63, 64

DVRP *Dynamic Vehicle Routing Problem*. xi

K-RF *Kmeans and Relax-and-Fix*. 30

K-RFO *Kmeans, Relax-and-Fix and Optimizations*. ix, xi, xv, 23, 29, 51–55, 63, 64

KA K*means-Aggregation*. 40, 51–54, 63, 64

KG K*means-Greedy*. ix, xi, 41, 46–50, 55

KP K*means-Partition*. 40, 51–54, 63, 64

LKH-3 Lin-Kernighan Helsgaun-3. 39, 51–54, 63, 64

OSRM *Open Source Routing Machine*. 38

PRAV Problema de Roteamento Aberto de Veículos. 1

PRDV Problema de Roteamento Dinâmico de Veículos. ix, xiii, xv, 1–3, 6, 8, 10, 11, 13, 16, 21–23, 25, 37

PRV Problema de Roteamento de Veículos. ix, xv, 1–3, 5, 6, 8–10, 14, 16, 17, 19, 37, 40, 55

PRVC Problema de Roteamento de Veículos Capacitados. 1, 6, 8, 16, 18, 29

PRVJT Problema de Roteamento de Veículos com Janela de Tempo. 1, 6, 9, 10, 18

QRPS *QRP-Sweep*. ix, xi, 41, 46–50, 55

R&F *Relax-and-Fix*. xiii–xv, 17, 18, 29, 33, 35

TSP *Travelling Salesman Problem*. 1, 2, 16, 19, 39, 41

VND *Variable Neighbourhood Descent*. 21

VNS *Variable Neighborhood Search*. 21, 22

VRP *Vehicle Routing Problem*. xi

Lista de Algoritmos

CAPÍTULO

1

Introdução

A utilização do comércio eletrônico tem se tornado cada vez mais presente no dia a dia das pessoas. Até poucos anos atrás, a aquisição de boa parte de serviços e produtos demandava deslocamento físico até um local. Atualmente, é possível utilizar o celular para realizar uma compra e esperar que chegue em sua casa. Os pedidos online, *e-commerces* e *deliveries* cresceram de maneira considerável, principalmente no ano de 2020, onde ficaram registrados diversos *lockdowns* em inúmeros países. Com o aumento da demanda por serviços de entrega, devido às compras online, a escala dos problemas envolvidos no setor logístico aumentaram de forma proporcional. Nesse contexto está inserido o Problema de Roteamento de Veículos (PRV) e suas variantes.

O PRV é uma generalização do problema do caixeiro viajante ou *Travelling Salesman Problem* (TSP), introduzido por Dantzig and Ramser (1959). Trata-se de um problema com enfoque em minimizar a distância percorrida por determinada frota de veículos iniciada em um ponto inicial, também conhecido por depósito. A partir do estudo do PRV diversas variações surgiram, como o Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) estudada por Bräysy and Gendreau (2005a)-Bräysy and Gendreau (2005b), Problema de Roteamento de Veículos Capacitados (PRVC) explorada por Ralphs et al. (2003), Problema de Roteamento Dinâmico de Veículos (PRDV) disseminado por Psaraftis (1988)-Psaraftis (1995) e Larsen et al. (2002), Problema de Roteamento Aberto de Veículos (PRAV), entre outros. A versão tradicional inteiramente artificial (com distâncias Euclidianas e valores inteiros, respeitando desigualdade triangular, simétrico, etc.), alcança resultados ótimos, em problema com 25 demandas como o trabalho de Azi et al. (2007). Em situações reais, em que empresas lidam com milhares de demandas, utilizar esses

métodos exatos é impraticável diante a limitação do uso de recursos computacionais e tempo de processamento, para alcançar o resultado ótimo (rota(s) que minimiza(m) todos os custos e maximiza(m) o lucro). Em uma situação real, onde deseja-se realizar a entrega das demandas no mesmo dia da requisição, solucionar através do PRV é inviável, visto que durante a jornada de trabalho dos veículos, novas demandas podem surgir no depósito a qualquer momento, tendo de refazer todos os cálculos das rotas já construídas. Para este caso, a variante dinâmica PRDV é mais indicada, uma vez que a variante procura rotear as demandas dinâmicas. Apesar da literatura mostrar que realizar o planejamento de todas as demandas, primeiro, entrega mais economia em distância percorrida pelos veículos, o desafio maior continua sendo realizar a entrega das demandas no mesmo dia em que os clientes solicitam, algo inesperado pelo PRV.

O PRV é um problema NP-difícil (Lenstra and Kan, 1981), por se tratar de uma generalização do problema TSP que é um problema NP-completo. A literatura da área dispõe de métodos exatos que utilizam mecanismos de otimizações com modelagem matemática. Entretanto, resolver um problema desse com centenas ou milhares de demandas torna-o computacionalmente inviável, visto que o número de variáveis geradas no problema cresce exponencialmente ao aumentar o número de demandas, afetando também a memória disponível da máquina. Devido a isso, existem trabalhos na literatura que propõem métodos heurísticos, visando obter soluções de boa qualidade em tempo computacional compatível com as necessidades operacionais verificadas em diversas aplicações reais. O primeiro trabalho que se destacou com uma heurística efetiva foi desenvolvido por Clarke and Wright (1964), que conseguiu reduzir drasticamente a demanda por recursos computacionais e encontrar uma solução próxima da ótima ou ótima em alguns casos. Outras técnicas foram aplicadas ao problema, tais como: meta-heurísticas investigadas por Alvarenga (2005), Taillard et al. (1997), Ralphs et al. (2003), Dorigo and Di Caro (1999), Marinakis and Marinaki (2010), aprendizagem por reforço visto por Nazari et al. (2018), Fonseca-Galindo et al. (2022), aprendizagem de máquina estudado por Comert et al. (2018), busca local aprofundado por autores como Branchini et al. (2009), Xu et al. (2013), Toth and Vigo (2003), e heurísticas combinando mais de uma técnica como, por exemplo Marinakis et al. (2009).

O cenário atual de compras online possui enfoque destacado para a realização de serviços de entregas rápidas, muitas delas em prazos de um dia útil. Nesse cenário, o "problema" das entregas pode ser interpretado como um PRDV uma vez que a medida que novos pedidos são efetivados, os serviços de entregas devem ser agilizados. O desafio computacional, nesse caso, diante de

situações onde os pedidos de entrega são recebidos sob demanda, consiste em otimizar os recursos disponíveis para que as restrições (tempo da entrega dos pacotes, por exemplo) sejam respeitadas e atendendo aos objetivos (minimizar o custo da entrega, por exemplo) do problema.

Esta proposta, em nível de mestrado, tem como principal objetivo investigar métodos de soluções para o PRDV, com enfoque destacado na abordagem dinâmica. O contexto das soluções propostas envolvem cenários reais de entregas de uma empresa brasileira de logística de grande porte. Especificamente, o problema geral de roteamento abordado neste trabalho possui fases estática e dinâmica. Na fase estática, todos os pedidos de entrega são conhecidos *"a priori"*, permitindo assim realizar o roteamento e alocação de recursos (veículos). Os veículos possuem diferentes capacidades e não há necessidade de retorno ao depósito de onde os produtos a serem entregues foram retirados. Na fase dinâmica, novos pedidos de entregas chegam em ordem aleatória. Dessa forma, esses novos pedidos podem alterar rotas que foram previamente criadas, mas que ainda não foram iniciadas.

Este texto está organizado conforme segue: o Capítulo 2 apresenta a base de conhecimento teórica-metodológica sobre o PRV, suas variantes assim como técnicas comumente utilizadas na resolução do PRV; o Capítulo 3 apresenta os modelos de otimização considerados, premissas, métodos e algoritmos para resolução dos problemas de roteamento dinâmico e estático; os resultados são explorados no Capítulo 4; no Capítulo 5 são apresentadas as conclusões diante dos principais resultados, as limitações técnicas das soluções, assim como possibilidades para melhoria e evolução das técnicas em trabalhos futuros.

CAPÍTULO

2

Revisão da Literatura

Este capítulo apresenta o Problema de Roteamento de Veículos (PRV), suas variantes, principais modelos matemáticos e métodos heurísticos promissores propostos na literatura. Além disso, apresentam-se trabalhos da literatura que utilizaram algoritmos baseados em aprendizagem de máquina, técnicas com relaxação baseado em modelo matemático e busca local com operadores de troca para lidar com o PRV.

2.1 Problema de Roteamento de Veículos

O setor logístico enfrenta muitos desafios, um dos principais desafios é o Problema de Roteamento de Veículos (PRV). Para ilustrá-lo, apresenta-se na Figura 2.1, o desafio visa realizar a entrega de uma série de demandas que estão inicialmente em um depósito para todos os clientes e retornar os veículos para o depósito, de modo a economizar o máximo de distância percorrida pelos veículos utilizados, ou minimizar custos operacionais.

O PRV foi introduzido por Dantzig and Ramser (1959), e pode ser definido através de um grafo $G(V,A)$, que possuí um conjunto V de vértices representando os clientes e o depósito, um conjunto A de arcos que identificam os caminhos existentes de um vértice a outro do grafo. O vértice 0 considerado depósito, possuí características únicas, dessa forma, para separá-lo ele dos clientes é utilizado o conjunto $V' = V \backslash \{0\}$ para apresentar somente os clientes. Baseado nesse grafo é possível gerar um modelo matemático, onde são definidos as **variáveis de decisão**, os **parâmetros de entrada**, as **restrições** do problema e a **função objetivo**, que será uma métrica para avaliar os custos operacionais. Vale ressaltar, que o PRV na prática geralmente não acontece,

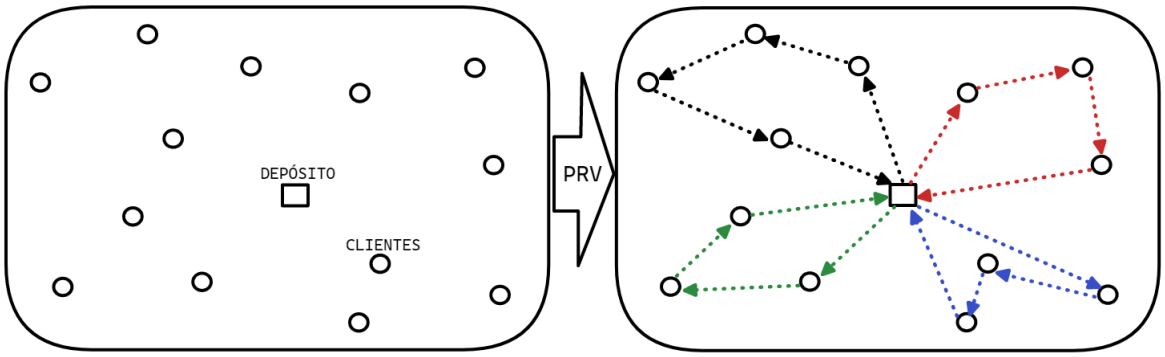


Figura 2.1: Representação do Problema de Roteamento de Veículos

porém, existem variantes do problema que se adéquam, na prática, como, por exemplo, o Problema de Roteamento de Veículos Capacitados (PRVC) que diferencia a capacidade máxima dos veículos disponíveis e o Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) que se preocupa em realizar a entrega de determinado cliente dentro de uma janela de tempo. Tais variantes são exploradas na Seção 2.2. A variante que será tratada neste trabalho é o Problema de Roteamento Dinâmico de Veículos que será explorada na Seção 2.3.

Parâmetros de Entrada

- V — Conjunto dos vértices, indexado por i, j ou h.
- K — Conjunto dos veículos, indexado por k.
- d_{ij} — Distância percorrida no arco (i, j).
- c_{ij}^k — Custo incorrido quando o veículo k percorre o arco (i, j).
- q_i — Quantidade de itens requisitadas pelo cliente i.
- Q_k — Capacidade máxima do veículo k.

Variáveis de Decisão

- $x_{ij}^k \in \{0,1\}$ — Variável binária. Assume o valor 1 caso o arco (i, j) seja percorrido pelo veículo k e 0, caso contrário.
- w_i^k — Variável binária. Indica se o veículo k atende o cliente i.
- u_{ij}^k — Variável inteira que indica a quantidade de itens transportadas pelo veículo k, ao percorrer o arco (i, j).

Conforme as variáveis e parâmetros é possível construir uma função objetivo para o PRV, a qual é responsável por calcular a soma dos custos gerados por cada rota. Neste cenário, o veículo deve sair e retornar ao depósito $\{0\}$.

O modelo apresentado foi proposto por Hannan et al. (2018), consiste em minimizar a função objetivo (2.1), sujeito às restrições (2.2) a (2.10).

$$\min \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ij}^k x_{ij}^k \tag{2.1}$$

$$\textbf{sujeito a} \quad \sum_{k \in K} \sum_{i \in V} x_{ij}^k = 1 \qquad \forall j \in V' \tag{2.2}$$

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k = 1 \qquad \forall i \in V' \tag{2.3}$$

$$\sum_{i \in V} x_{ih}^k = \sum_{j \in V} x_{hj}^k = w_i^k \qquad \forall h \in V', \forall k \in K \tag{2.4}$$

$$\sum_{j \in V'} x_{0j}^k = 1, \qquad \forall k \in K \tag{2.5}$$

$$\sum_{i \in V'} x_{i(0)}^k = 1, \qquad \forall k \in K \tag{2.6}$$

$$\sum_{i \in V} u_{ji}^k - \sum_{h \in V} u_{jh}^k = q_j \qquad \forall j \in V' \tag{2.7}$$

$$\sum_{i \in V} \sum_{k \in K} q_i x_{ij}^k \leq Q_k, \qquad \forall j, \forall k \tag{2.8}$$

$$u_{ij}^k \geq 0, \qquad \forall i \neq j \tag{2.9}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall i \neq j, \forall k \in K \tag{2.10}$$

A função objetivo 2.1 representa a soma dos custos das rotas dos veículos, tais custos geralmente envolvem, a distância percorrida e o número de veículos utilizados. As restrições 2.2 e 2.3 exigem que cada pacote pertença a apenas um veículo, a 2.4 são restrições de conservação de fluxo. Tais restrições, acompanhadas das restrições apresentadas pela Equação 2.2 obrigam a formação de um ciclo para cada rota. Em outras palavras, se uma rota chegar em h necessariamente deve sair de h, apenas uma vez. As restrições presentes na Equação 2.5 garantem que exista apenas uma partida do depósito por veículo. As restrições presentes na Equação 2.6 garantem que exista apenas um retorno ao depósito por veículo. Nas restrições 2.7 é garantido o atendimento das demandas dos clientes. Na Figura 2.2, o veículo que chega em j está transportando uma quantidade u_{ij}^k de itens e ao sair de j estará transportando uma quantidade de $u_{ij}^k - q_j$. Vale notar que $j \in V'$, pois, não há necessidade de entregar algum objeto ao depósito.

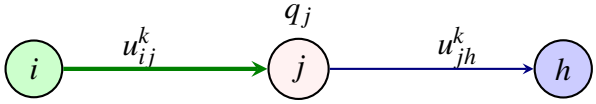


Figura 2.2: Detalhamento sobre as quantidades de itens que devem ser transportados entre os vértices i, j e h no problema de roteamento com veículos capacitados.

Com um modelo matemático construído é possível utilizar otimizadores,

como o CPLEX, para encontrar soluções factíveis. As características da frota e dos pacotes podem ser mais detalhadas e novas restrições, objetivos surgirão no problema, dando origem às variantes do PRV.

2.2 Variantes do Problema de Roteamento de Veículos

Variantes do PRV surgiram a partir da definição do problema original e, em várias situações, adotam a heterogeneidade entre os veículos da frota. Os modelos apresentados são para as variantes PRVC com heterogeneidade e para o PRDV que possuem como referência algumas restrições do modelo apresentado na Seção 2.1, mas também utilizarão novos dados de entrada, novas variáveis de decisão e restrições adicionais.

2.2.1 Problema de Roteamento de Veículos Capacitados

O Problema de Roteamento de Veículos Capacitados possui restrições que impedem os veículos carregarem cargas além de sua capacidade. Vale notar que uma frota pode assumir duas características: homogênea ou heterogênea. No caso da homogênea, todos os veículos possuem capacidades iguais, e na heterogênea, os veículos possuem capacidades distintas. O PRVC pode utilizar o modelo apresentado na Seção 2.1. Mantendo os parâmetros de entrada, a função objetivo e as variáveis de decisões por tratarem-se de questões sobre a capacidade conforme a restrição 2.8. Para problemas reais, que possuem um número elevado de demandas, o modelo é construído, mas a solução ótima, na maioria das vezes, não é encontrada. Isso decorre do fato do problema ser NP-difícil (Lenstra and Kan (1981)). Para contornar essa questão de tempo restrito, são geralmente utilizadas heurísticas visando aproximar de soluções ótimas com menos recursos computacionais. A heurística pode ser usada como uma combinação de heurísticas (construtivas, refinamento e meta-heurísticas) com métodos de otimizações.

A primeira e mais famosa heurística para o PRV foi proposta por Clarke and Wright (1964) ela é dividida em 4 etapas:

- **Etapa 1:** Considere n, sendo o número de demandas, então, monte n rotas. Cada uma, com uma demanda distinta da outra.

- **Etapa 2:** Monte uma lista de economias. Uma economia é dada, por $e_{ij} = d_{i0} + d_{0j} - d_{ij}$, onde i e j são demandas a serem percorridas.

- **Etapa 3:** Ordene a lista de economias, em ordem decrescente.

- **Etapa 4:** Percorra a lista de economias e, para cada e_{ij} selecionada, a demanda j é atribuída a rota de i, caso a rota de i possua capacidade suficiente para atender a demanda j.

Outra heurística conhecida introduzida por Helsgaun (1998) é apresentada na Seção 4.1.1.

2.2.2 Variante com Janela de Tempo

O Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) possui uma regra onde os clientes precisam ser atendidos em um intervalo de tempo. Um exemplo para este caso é: suponha que uma pessoa é envenenada por uma cobra e precisa de um antídoto em 1 hora, caso contrário ela morre. Neste caso, deve haver um intervalo de tempo de 0 até 1 hora para que uma ambulância chegue ao local e aplique o antídoto no paciente. Problemas deste tipo podem ser definidos em um modelo matemático. Para construção do modelo, adotam-se as mesmas definições de função objetivo, variáveis de decisões, parâmetros de entrada e restrições descritas no PRV e acrescenta-se um intervalo de tempo $[a_i, b_i]$ onde a_i é o tempo mínimo que o cliente i aceita o produto e b_i o tempo máximo que o cliente i suporta aguardar. Azi et al. (2007) propuseram um modelo matemático que utiliza os seguintes parâmetros e variáveis.

Parâmetros de Entrada

- t_{ij} — Tempo necessário para percorrer a aresta (i, j)).
- ts_i — Tempo de serviço a ser realizado no vértice i.
- $[a_i, b_i]$ — Janela de tempo que o serviço do cliente i deve ser realizado.
- β — Constante de ajuste para os clientes serem atendidos na janela de tempo, quanto mais próximo de 0, menos os tempos de serviços ts_i influenciam.

Variáveis de Decisões Adicionais

- σ^k — Dado pela Equação 2.17, tempo de serviço total realizado pelo veículo k.
- t_i^k — Tempo inicial do serviço do cliente i presente na rota do veículo k.
- $t_{inicial}^k$ — Tempo de início da rota do veículo k.
- t_{final}^k — Tempo final da rota do veículo k.

Além disso, para este problema foi adicionada a constante β, para permitir que os pedidos com mais urgência sejam atendidos primeiro. Azi et al. (2007) e Zirour (2008) adotaram $\beta = 0.2$. Para simplificar, o tempo de serviço total atribuído ao veículo k multiplicado por β será representado por σ^k, como mostra a Equação 2.17.

Considerando a função objetivo 2.1, um modelo para o Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) pode ser obtido, ao adicionar as restrições de 2.11 a 2.16 no modelo estabelecido da Seção 2.1.

$$\sum_{j \in V} x_{ij}^k = w_i^k, \qquad i \in V', k \in K, \quad (2.11)$$

$$t_i^k + ts_j + t_{ij} - M(1 - x_{ij}^k) \leq t_j^k, \qquad (i,j) \in G, k \in K \quad (2.12)$$

$$a_i w_i^k \leq t_i^k \leq b_i w_i^k, \qquad i \in V, k \in K \quad (2.13)$$

$$t_0^1 \geq \sigma^1 \quad (2.14)$$

$$t_{final}^k + \sigma^{k+1} \leq t_{inicial}^{k+1}, \quad k = \{1, \ldots, |K|-1\} \quad (2.15)$$

$$t_i^k \leq t_{inicial}^k + t_{max} \qquad i \in V, k \in K \quad (2.16)$$

$$\sigma^k = \beta \sum_{i \in N} ts_i w_i^k, \qquad k \in K, \beta = 0.2 \quad (2.17)$$

As restrições 2.11, indicam que se o pacote $i \in V'$ for carregado pelo veículo k, então o veículo k percorre o caminho (i, j). As restrições 2.12, consideram que todos os tempos não ultrapassem alguma janela de tempo, caso exista o caminho (i, j) pelo veículo k, considerando M como uma grande constante arbitrária. As restrições 2.13, garantem que todos os clientes serão atendidos na janela de tempo estabelecida pelo cliente i. Dado o σ^k pela Equação 2.17, são criadas as restrições 2.14 e 2.15, nas quais, indicam a influência, que o tempo de serviço total do veículo k tem em relação ao tempo total transcorrido. As últimas restrições em 2.16, indicam que o tempo máximo, não pode ser ultrapassado pelo tempo de trabalho que a frota gasta.

2.3 Problema de Roteamento Dinâmico de Veículos

Um problema é dito como estático caso todas as informações contidas nele não sejam alteradas conforme o andamento da resolução. Portanto, o problema dinâmico é quando não se tem o conhecimento total de todas as informações que podem ocorrer. Por exemplo, o PRV é considerado um problema estático por possuir todas as informações de todos os pacotes que serão entregues. Entretanto, quando a informação dos pacotes são entregues conforme a chegada de novos pacotes em um depósito, o problema estende-se para a variante Problema de Roteamento Dinâmico de Veículos (PRDV). Psaraftis

(1988), foi um dos pioneiros a estudar a variante, posteriormente, foi explorada por Larsen et al. (2002), onde propôs diferentes formas de avaliar o grau de dinamismo do problema.

Os estudos da literatura apresentam mais soluções para a abordagem estática, quando comparado com a abordagem dinâmica. Esse fato se justifica, pois, as conclusões com as abordagens estáticas são mais assertivas por terem o total conhecimento da localização dos clientes. Na abordagem dinâmica essa assertividade torna-se uma previsão em decidir qual a melhor rota a colocar um próximo pacote recebido. Assim, quando comparada ambas as abordagens, a estática se sobressai diante das métricas adotadas como distância percorrida pela frota e/ou o número de veículos utilizados. Contudo, a exploração por soluções dinâmicas, que sejam equiparadas a uma solução estática, é um grande desafio a ser resolvido.

Para realizar o roteamento dinâmico de veículos deve haver uma estratégia capaz de responder, o que fazer quando uma nova demanda é inserida no problema em que existem rotas pré-construídas. Além disso, um parâmetro destacado neste contexto é o tempo transcorrido a partir do início das entregas, dado que a cada instante a localização dos veículos é modificada. Do ponto de vista do problema, as novas demandas que surgem são desconhecidas.

A Figura 2.3 ilustra um exemplo do que é o PRDV:

1. Na primeira fase do processo representada por A, tem-se 7 clientes para serem atendidos totalmente conhecidos no tempo inicial, representado pelos vértices numerados de 1 a 7.

2. Na fase seguinte, representada em B, os veículos já haviam iniciado suas rotas e no momento em que havia um veículo no cliente 3, e o outro no cliente 5, surgiram as novas demandas 8 e 9.

3. Agora, as rotas pré-definidas precisam ser recalculadas de modo a reotimizar o problema.

4. O objetivo na última fase do processo, representada em C, é que todos os clientes sejam atendidos e os veículos retornem ao depósito.

Um conceito considerado em PRDVs, é que um sistema dinâmico possui um grau de dinamismo (Larsen et al. (2002)). Para calcular esse grau de dinamismo utiliza-se a Equação 2.18. Como exemplo de aplicação, na Figura 2.3 tem-se 2 demandas dinâmicas e 9 demandas no total, assim $\circ_{od} = \frac{2}{9} = 22,22\%$. Entretanto, esta medida de dinamismo não é efetiva quando são comparados dois sistemas dinâmicos. Por exemplo, suponha que em um primeiro sistema há mais demandas dinâmicas próximo ao tempo inicial. Enquanto

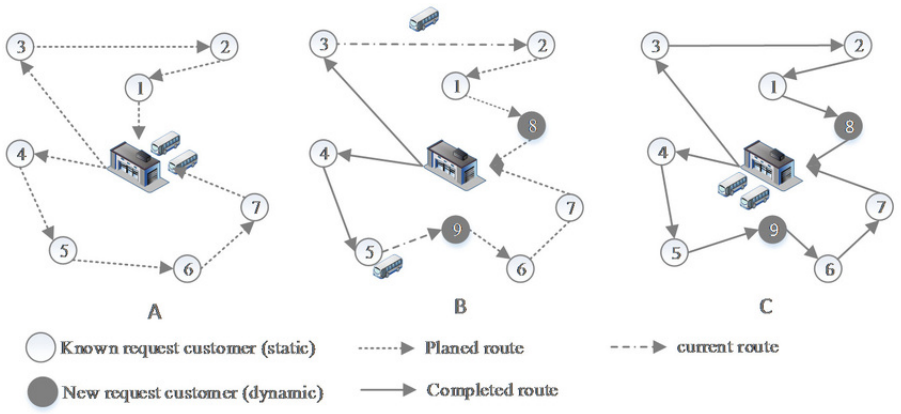


Figura 2.3: Representação do problema de roteamento dinâmico de veículos Chen et al. (2017).

um segundo sistema possua mais demandas dinâmicas no tempo final do expediente da frota. Nesse exemplo, o primeiro sistema possui mais tempo (e oportunidades) disponível para gerar rotas, do que o segundo sistema.

$$\circ_{od} = \frac{\textit{Número de demandas dinâmicas}}{\textit{Número total de demandas}} \tag{2.18}$$

Para o grau de dinamismo ser efetivo é preciso considerar o tempo t_i em que cada demanda dinâmica é recebida para ser incluída em alguma rota. Considere que o tempo inicial é 0 e o tempo final do expediente é dado por T, assim $0 \leq t_i \leq T$. O número de demandas dinâmicas é denotado por η_{din}, o número de demandas estáticas por η_{est} e o número total de demandas $\eta_{tot} = \eta_{din} + \eta_{est}$. Com isso, pode-se calcular um grau de dinamismo efetivo no tempo dado por $\zeta\circ_{od}$ de acordo com 2.19:

$$\zeta\circ_{od} = \frac{\sum_{i}^{\eta_{din}} \frac{t_i}{T}}{\eta_{tot}} \tag{2.19}$$

A partir de $\circ_{od}$ é possível verificar, se o sistema é puramente dinâmico ($\circ_{od} = 1$) e puramente estático ($\circ_{od} = 0$). De acordo com Larsen et al. (2002), um sistema dinâmico pode ser categorizado de três maneiras:

- **Fracamente dinâmico** — $0 < \circ_{od} \leq 20\%$.
- **Moderadamente dinâmico** — $20\% < \circ_{od} < 80\%$.
- **Fortemente dinâmico** — $80\% \leq \circ_{od} < 100\%$

O foco principal é a resolução das categorias fraca e moderadamente dinâmico, onde existe uma parcela considerável de demandas estáticas. Um modo para tratar a parte estática é utilizando o modelo matemático estabelecido na Seção 2.1 com as considerações de parâmetros e variáveis de decisões atribuídas na Seção 2.2.2. Após a construção das rotas, o problema torna-se

dinâmico e após novas demandas, é necessário haver uma re-otimização das rotas já construídas. A re-otimização considera que cada cliente possui uma demanda bem definida no depósito, proibindo a possibilidade de um veículo **A** em rota entregar diretamente a demanda para um cliente. Entretanto, um novo veículo **B** pode retirar o produto do depósito e entregá-lo ao veículo **A** que estava com uma rota definida. Para isto acontecer, uma re-otimização das rotas é realizada, pois o veículo **A** teve seus pacotes alterados e a rota do veículo **B** foi adicionada ao problema.

Um fator crucial no PRDV é o tempo limite para entrega, visto que a cada instante de tempo o número de demandas é variável (com cancelamentos e novos pedidos). Com isso, foi construída uma estratégia para inserção dos pacotes em lotes. Ou seja, é esperado um número pré-estabelecido de pacotes para realizar o roteamento. O primeiro lote inserido no problema pode ser categorizado como estático. Após resolvido, os próximos pacotes terão características dinâmicas. Contudo, a estratégia dinâmica também pode ser usada desde o primeiro pacote.

Um modo de prever a localidade das próximas demandas é por mineração de dados Fonseca-Galindo et al. (2022), utilizando dados históricos de uma região desejada. Permitindo o reconhecimento de padrões gerados pelos dados. Entretanto, a maioria dos estudos realizados são baseados em dados fictícios ou dados gerados pseudo-aleatórios, como o repositório de Solomon (1987), com exceção no Fonseca-Galindo et al. (2022) que realizou estudos baseados nos estados brasileiros do PA, RJ e DF, capaz de realizar inferências com árvore de padrões frequentes.

Outros trabalhos explorados que resolveram a variante do dinâmico utilizaram meta-heurísticas, sendo *Tabu Search* no artigo de Taillard et al. (1997), *Ant Colony Optimization* (ACO) nos trabalhos de Dorigo and Di Caro (1999), Montemanni et al. (2005) e Necula et al. (2017), Algoritmo Genético na tese de Alvarenga (2005), Enxame de Partículas no trabalho de Marinakis and Marinaki (2010). Dentre os artigos sobre meta-heurísticas, aqueles que se destacaram por explorarem a meta heurística ACO foram Montemanni et al. (2005) e Necula et al. (2017), os quais sendo um método de otimização inspirado na natureza das formigas. Elas trabalham através de feromônios e conforme o número de formigas aumenta em determinado caminho, ele fica mais atrativo para cada uma delas explorá-lo. O cálculo do feromônio considera τ_0 como parâmetro inicial, calculado por:

$$\tau_0 = (n_{av} L_{NN})^{-1} \tag{2.20}$$

$$L_{NN} = \sum_{i \in V} \sum_{j \in V} \sum_{k \in M} c_{ij}^k x_{ij}^k \tag{2.21}$$

- n_{av} — Quantidade de clientes disponíveis.
- L_{NN} — Distância percorrida por todas as rotas calculadas em 2.21.

2.4 Trabalhos relacionados

Nesta Seção será apresentado os trabalhos que trouxeram o conhecimento necessário para o surgimento das estratégias implementadas e desenvolvidas no Capítulo 3.

2.4.1 K-means

O *K-means* é um método não-supervisionado da aprendizagem de máquina, que pode ser utilizado para particionar elementos próximos entre si, de um problema inicial ou para classificação de novos elementos ao problema. Inicialmente, o *K-means* necessita de um parâmetro R, do qual indica o número de regiões a serem geradas. O algoritmo funciona com 4 passos:

- **1º Passo:** Dado o número R, R centroides distintos são gerados.
- **2º Passo:** Para cada elemento é feito uma busca do centroide mais próximo, encontrado, é dito que este elemento pertencerá à região deste centroide.
- **3º Passo:** Em cada uma das R regiões é calculado um novo centroide, gerado a partir da localização média de todos os elementos da região.
- **4º Passo:** Os passos 2º e 3º são executados novamente, até os elementos não trocarem mais de região, indicando uma estabilidade das regiões.

Resolver grandes instâncias de problemas não determinísticos em tempo polinomial ainda é uma questão em aberto na área da computação. Portanto, fragmentar um grande problema em partições que fazem sentido, também é um desafio que pode ser feito de inúmeras formas no PRV. O mais investigado neste trabalho foi o algoritmo *K-means* que pode ser utilizado para particionar o PRV e suas variantes, assim como a técnica *cluster-first* e *route-second* desenvolvida por Fisher and Jaikumar (1981). Essa técnica utiliza o algoritmo de *K-means* como um modo de particionar, e posteriormente realizar o roteamento de cada uma das partições geradas.

A utilização do *K-means* é geralmente para planejar o particionamento do PRV em larga escala e para problemas com características regionais, artigos como He et al. (2009), Reed et al. (2014), Singanamala et al. (2018), Xu et al. (2018) e Le et al. (2022) buscaram particionar o problema em uma primeira etapa, e depois trabalhar de forma mais eficiente no problema, pois faz com

que os veículos possuam rotas com pacotes próximos. Entretanto, buscar pelo melhor particionamento para o problema têm seus desafios, tais como:

- Definir o número de regiões.
- Definir o formato das regiões construídas.
- Definir a métrica utilizada para avaliar as regiões.
- Definir como os veículos irão atuar nas regiões.

Para decisão do número de regiões deve ser adotado um método para definir o melhor R que particiona o problema original, geralmente o número de regiões explorados variam de 2 até A_U, número definido pelo usuário. Dois métodos bem conhecidos, são conhecidos por:

- ***Elbow* (Método do cotovelo):**

  Os primeiros estudos desse método foram realizados por Thorndike (1953), primeiro são selecionados os valores possíveis de R, após isso basta realizar os cálculos do *K-means* obtendo R centroides. A métrica para obter o melhor R é dada pela quantidade de agrupamento, em que a soma dos quadrados intra-clusters seja o menor possível. Essa quantidade também é conhecida como inércia e assume que seus clusters são convexos e isotrópicos, evitando cluster com formatos irregulares ou alongados.

- **Coeficiente de Silhueta:**

  Esse método foi proposto por Rousseeuw (1987). Dados os valores possíveis de R para calcular qual deles é o melhor, cada indivíduo é avaliado através da Equação 2.22, onde cada indivíduo é representado pelo índice i. Para cada um é calculado o valor de $\mathcal{S}(i)$, $a(i)$ é a dissimilaridade média do objeto i em relação a todos os objetos do mesmo cluster e $b(i)$ é a dissimilaridade média entre o objeto i em relação a todos os objetos do grupo vizinho mais próximo a ele.

$$\mathcal{S}(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{2.22}$$

Por fim, existem algumas formas de inserir veículos nas regiões geradas pelo *K-means*, tais como:

- Considerar que cada região é responsável por um veículo.
- Considerar que os veículos só podem passar por uma região.
- Tratar as demandas próximas das fronteiras das regiões de forma diferente das outras.

- Permitir um veículo entregar demandas de regiões diferentes com algumas condições adicionais.

Vale notar que existem outras formas de realizar o particionamento seguindo uma ideia próxima do *K-means*, como a utilização do *K-medoid*, *K-means* restrito e clusterização aleatória. A diferença do *K-medoid* para o *K-means* está no 3º passo, onde se esperava calcular o novo centroide, a partir da média dos elementos da região. Neste caso, o novo centroide (considerado medoide) é um elemento existente e considerado o elemento central da região, e não um centroide que se difere dos elementos da região. O *K-means* restrito, é uma forma de restringir o número mínimo e o número máximo de elementos em determinada região. No trabalho de Comert et al. (2018) foi realizado uma exploração em uma gama de artigos que utilizaram o *K-means* para resolução do PRV. Geralmente, o *K-means* é utilizado como uma primeira fase de planejamento, em seguida, é realizado o roteamento dos pacotes para em cada região pela solução de um TSP, ou seja, cada região com apenas um veículo. Além do *K-means*, técnicas como o *K-medoid* e clusterização aleatória foram avaliadas, baseados no repositório do Solomon (1987).

Os estudos de He et al. (2009) agregaram a possibilidade de balancear as regiões geradas pelo *K-means*, demonstrando efetividade para problemas de larga escala. Essa ideia de balancear as regiões geradas foi estendida nos trabalhos de Reed et al. (2014) , Le et al. (2022), Singanamala et al. (2018). Além disso, o trabalho de Xu et al. (2018) abordou a variante PRDV.

O trabalho de He et al. (2009) explorou o balanceamento do *K-means* para resolver o PRV e demonstrou efetividade para larga escala. A solução proposta possui duas fases: a primeira fase é responsável por utilizar o *K-means* puramente inserindo o valor R a critério do autor. Na segunda fase, as demandas encontradas nas bordas das regiões são balanceadas umas com as outras, resultando em áreas com uma quantidade de cargas aproximadamente iguais. A proposta foi avaliada considerando um particionamento do problema de 1882 demandas. Ao aplicar o *K-means* inicialmente obteve uma variação de 1006 demandas entre os clusters. Após algumas iterações de balanceamento, a variação entre os clusters diminuiu significativamente para 18 demandas. Esse artigo agregou em como utilizar o *K-means* e para resolver o problema PRV.

Outro trabalho bem conhecido que buscou alternativas de agrupamento com o *K-means* foi o de Reed et al. (2014). Para resolver o PRVC primeiro é calculado o número máximo de clusters como o número mínimo de veículos possíveis K. O valor de R é justamente o número de veículos que percorrerão os trajetos. Para cada cluster haverá um único veículo. Para balancear as cargas totais dos clusters foi calculada a média de demanda que deveria ter em cada um, e identificados aqueles com carga em excesso. Em seguida, os nós

periféricos dos clusters em excesso são movidos para outras regiões até que fiquem balanceadas. A informação de cada cluster individual é repassada para um algoritmo baseado em otimização por colônia de formigas. Os resultados demonstraram um ganho expressivo no processamento computacional.

O terceiro trabalho de Le et al. (2022) também observou o ganho de tempo computacional para encontrar soluções ótimas para o problema. Neste caso o algoritmo *K-means* passou pelos 4 passos descritos na seção 2.4.1.

O quarto trabalho de Singanamala et al. (2018) que também aplicou o *K-means* para classificar a proximidade entre as demandas agrupando-as em determinada região. Para cada região foi aplicado o algoritmo de Clarke and Wright (1964) para realizar o roteamento de veículos. Para comparação de construção de roteamento foi utilizada a meta-heurística baseada na otimização de colônia de formigas para cada região. Os resultados mostraram uma menor distância quando cada região era resolvida pelo algoritmo Clarke and Wright (1964).

2.4.2 Heurística Relax-and-Fix

Os métodos exatos aplicados para resolução do PRV geralmente utilizam modelos matemáticos e são custosos computacionalmente. Devido a isso foi investigada a heurística *Relax-and-Fix* desenvolvida por Dillenberger et al. (1994) para realização do roteamento dos veículos, alguns autores como: de Menezes (2021) e Santos (2019) utilizaram essa heurística. Porém, para outras finalidades mostrada após a explicação do funcionamento da heurística.

A heurística é geralmente baseada em um modelo matemático de interesse, que possui variáveis de decisões inteiras ou binárias. Antes de implementar a heurística, é necessário particionar o problema em um horizonte de planejamento, onde as partições devem estar relacionadas diretamente com as variáveis de decisões escolhidas. Suponha um cenário que tenha um modelo matemático com uma variável de decisão x_t, onde t é um índice que varia de 0 até o número total de variáveis do problema. Feito isso, é preciso que a variável x_t seja particionada da forma que o usuário demonstrar mais interesse. Separadas as variáveis em r partes, como representado na Figura 2.4. Baseado nisso, as seguintes etapas são seguidas:

1. Relaxe todas as variáveis de interesse, transformando-as em números reais. No caso da variável ser binária, considere uma variação entre 0 e 1.

2. Selecione uma parte das r partes existentes, e a transforme em inteira/binária novamente.

3. Resolva o modelo matemático de interesse e fixe as variáveis inteiras encontradas.

4. Repita o processo 2 e 3 até que as r partes sejam resolvidas.

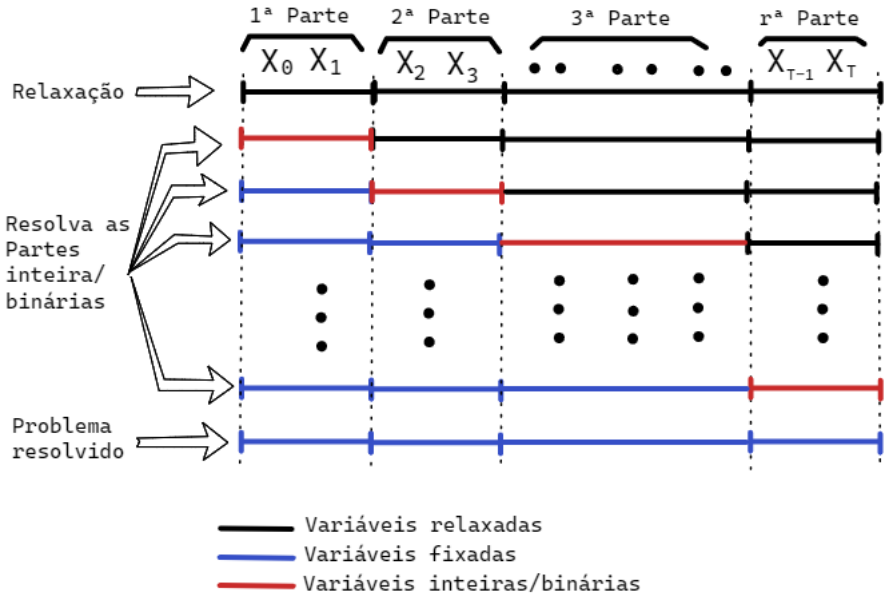


Figura 2.4: Descrição da *Relax-and-Fix*

O trabalho de Menezes (2021) explorou a heurística para carregar objetos de três dimensões em cada veículo. Primeiramente, ele resolve o PRVC com modelagem matemática e, após isso, utiliza a heurística para inserir os pacotes tridimensionais nos veículos. Já no trabalho de dissertação de Santos (2019) foi explorado o método para resolver o PRVJT com coleta e descarga de produtos químicos. O objetivo principal era reduzir o processo de planejamento que obteve um resultado considerável reduzindo o processo de 60 dias para 2 horas. A heurística foi implementada para resolver todas as variáveis de decisões do modelo. A partição foi definida separadamente para cada uma das variáveis existentes no problema, onde a primeira variável é a de roteamento e as restantes foram variadas para obter uma avaliação do melhor resultado.

É notória a escassez da utilização da *Relax-and-Fix* para o problema de roteamento de veículos. Dados os resultados de melhora de tempo computacional, neste trabalho a heurística foi investigada para resolver o roteamento

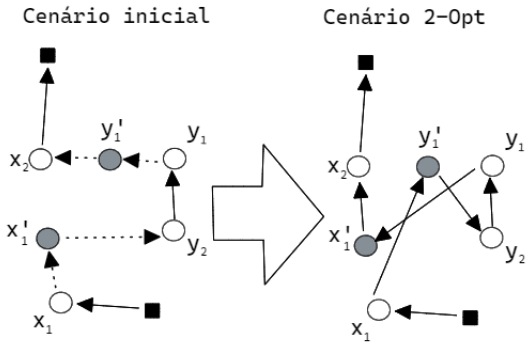


Figura 2.5: Cenário analisado pela 2-Opt.

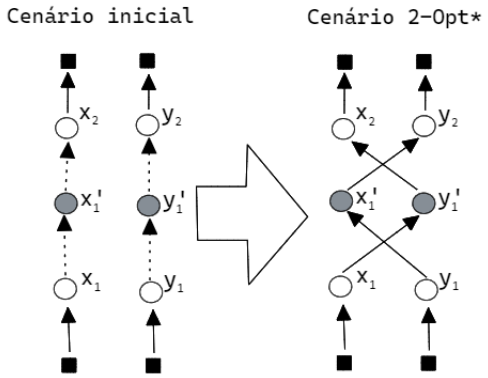


Figura 2.6: Cenário analisado pela 2-Opt*.

dos veículos para instâncias variando a quantidade de pacotes de 300 até 4000.

2.4.3 Busca local (2-Opt*)

Heurísticas de busca local são geralmente utilizadas após a obtenção de uma solução inicial do PRV como melhoria de rotas. A primeira busca local bem conhecida proposta por Croes (1958) é a 2-Opt normalmente aplicada para o TSP. A ideia da heurística é selecionar duas demandas de uma rota como ilustrado na Figura 2.5. Selecionadas as demandas x'_1 e y'_1 um cenário de troca entre elas é realizado caso haja ganho de distância. Caso contrário, não ocorrerá nenhuma troca. Contudo, a 2-Opt pode ser observada como uma heurística intra-rota que observa somente ela mesma. Em uma análise mais detalhada Potvin and Rousseau (1995) utilizou essa abordagem como inter-rota para um PRV denominando-a de 2-Opt*. Nesse caso, duas demandas são selecionadas e estão dispostas em rotas distintas, como apresentado na Figura 2.6, em que x'_1 pertence a uma rota e y'_1 em outra. Para a troca ser realizada deve-se avaliar o cenário que possui a menor distância percorrida entre as rotas. Posteriormente, o nome da heurística foi generalizada para $or-$Opt, onde or é o número de demandas selecionadas para realizar uma troca. Outros operadores de melhorias para Busca Local são conhecidos:

- ***Shifts/Exchanges*(ρ, τ):**

  Estudado por Potvin and Rousseau (1995) o mecanismo é usado como um operador inter-rotas e pode ser dado por *Shift*(ρ, τ), onde ρ é o número de pacotes selecionados na rota 1, e τ da rota 2. Com essa informação, a quantidade de movimentos realizados é de no máximo $\rho + \tau$ vezes. Em poucos casos esse tipo de movimentação pode anular totalmente uma das rotas caso a rota 1 tenha $\rho - \tau$ pacotes ou a rota 2 tenha $\tau - \rho$ pacotes.

- ***Chain-Exchange*:**

  Chain-Exchange é outro operador para busca local, desenvolvido por Fahrion and Wrede (1990). Seu funcionamento é dada por uma solução inicial, onde são selecionadas duas sub-cadeias $I = (i_1, ..., i_M)$ e $J = (j_1, ..., j_P)$, onde M é o tamanho da sub-cadeia de I e P o tamanho da sub-cadeia de J. Essas duas sub-cadeias não podem pertencer a mesma rota, tornando esta melhoria inter-rota. Após isso, uma análise de trocas entre o conjunto I e J é realizada, verificando as melhores possibilidades das cadeias serem inseridas entre as trocas.

- ***Cross-Exchange*:**

  Nos estudos de Taillard et al. (1997) foi apresentado o operador de troca *Cross-Exchange*, um segmento da rota X ($x'_1, ..., x'2$) e um da rota Y ($y'_1, ..., y'_2$) são selecionados, sendo efetuada uma troca direta caso haja melhora na distância percorrida, como ilustrado na Figura 2.7

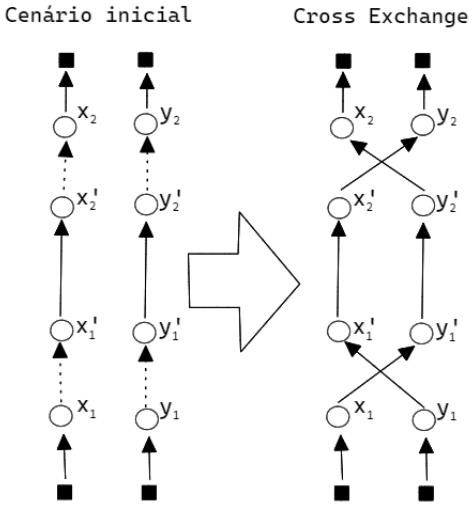


Figura 2.7: Representação do *Cross-Exchange*.

Geralmente dois pontos importantes precisam ser definidos pelo usuário para a utilização dos mecanismos de troca:

- **Estrutura de vizinhança:**

A estrutura deve ter características de proximidade entre os pontos existentes, sendo por variável em comum entre os pontos.

- **Sistema de pertubação:**

  Definida a estrutura de vizinhança, o usuário deve escolher um mecanismo de destruição das variáveis do sistema, e reconstruí-las com operadores de Busca Local, ou baseado em alguma meta-heurística como o *Variable Neighbourhood Descent* (VND), explorado por Chen et al. (2010)) ou *Variable Neighborhood Search* (VNS), por Hansen et al. (2010)).

Dois trabalhos que emergiram com a Busca Local no PRDV foram: Branchini et al. (2009) e Xu et al. (2013). Branchini et al. (2009) adaptaram a técnica *Granular Local Search* proposta por Toth and Vigo (2003), onde a estrutura de vizinhança é gerada a partir de uma lista de candidatos que não excedem um limite de granularidade ϑ, definido pela Equação 2.23

$$\vartheta = \beta \frac{z'}{n+m} \tag{2.23}$$

Onde, z' é o valor da distância percorrida total obtida através de uma heurística tradicional, onde nesse caso a utilizada foi a heurística de Clarke and Wright (1964), n o número de clientes, m o número de veículos e β um parâmetro de ajuste do tamanho da vizinhança granular. A vizinhança granular elimina a possibilidade de passar por caminhos que superam o valor de ϑ. Os operadores utilizados para intra-rota foram *2-Opt* e $Or-$Opt e para inter-rota Or-Opt *insertion*, Or-Opt *exchange* e *Crossover*. Esse trabalho destacou-se por tratar de instâncias reais disponibilizadas por uma companhia de transporte de Brasília, os resultados demonstrados foram promissores quando comparados com outras duas estruturas de vizinhanças: *Nearest Neighbor* e *Best insertion*.

Xu et al. (2013) utilizaram VNS meta-heurística explorada por Hansen et al. (2010). A meta-heurística parte de uma solução inicial desenvolvida por um algoritmo de clusterização, e roteirizada pelo algoritmo de Clarke and Wright (1964). E adotaram como estrutura de vizinhança, os pacotes inseridos em uma rota com capacidade suficiente para ser atendidos, e a pertubação gerada pelo autor é realizada através dos operadores de troca, tais como: o operador p_{insert} que realiza a inserção de um pacote em uma rota mais próxima que não tenha a capacidade excedida, *cross-exchange*, *iCross-exchange*, o qual é uma extensão do operador *cross-exchange*, e a utilização de 2-Opt e 3-Opt nas intra-rotas. A maneira de escolher os pacotes perturbados é de forma randômica, porém com cada operador tendo uma porcentagem de chance de ocorrer. Nos experimentos realizados considerou-se $p_{insert} = 20\%$, $p_{cross} = 15\%$ e

$p_{iCross} = 10\%$ e para as intra-rotas as possibilidades do 2-Opt ou do 3-Opt ocorrer é em 50% para cada. Os resultados para o PRDV nas instâncias do Solomon (1987) demonstraram um ganho na distância percorrida dos veículos ao comparar uma solução sem o IVNS e outra com.

2.4.4 Síntese dos Trabalhos Relacionados

A partir dos trabalhos relacionados, a Tabela 2.1 foi gerada destacando fatores importantes para análise e comparação entre os trabalhos:

- Os métodos utilizados para resolver o problema.
- Se o problema de roteamento é estático ou dinâmico.
- O tamanho das das instâncias utilizadas para avaliação e validação das soluções.
- Se a distância entre os pontos é uma distância euclidiana.

Autor	Ano	Métodos	Dinâmica	Tamanho	Cálculo de Distancia
Clarke and Wright (1964)	1964	Heuristica		50-199	Euclidiana
Fisher and Jaikumar (1981)	1981	Kmeans, TSP		50-199	Euclidiana
Fahrion and Wrede (1990)	1990	Chain Exchange		100	Euclidiana
Taillard et al. (1997)	1997	Tabu Search		50-199	Euclidiana
Dorigo and Di Caro (1999)	1999	ACO		50-199	Euclidiana
Azi et al. (2007)	2007	Método Exato		100	Euclidiana
He et al. (2009)	2009	Kmeans, TSP		1882	Euclidiana
Hansen et al. (2010)	2010	VNS		50-199	Euclidiana
Reed et al. (2014)	2014	Kmeans		50-199	Euclidiana
Helsgaun (2017)	2017	LKH-3		50-199	Euclidiana
Comert et al. (2018)	2019	Kmeans		50-199	Euclidiana
Santos (2019)	2021	RF		10	Maritimo
de Menezes (2021)	2022	RF		20-100	Euclidiana
Le et al. (2022)	2022	Kmeans		39	Real
Este Trabalho	**2022**	**KRFO**		**300-4000**	**OSRM**
Larsen et al. (2002)	2002	Caso de estudo	X	50-199	Euclidiana
Alvarenga (2005)	2005	Algoritmo Genetico	X	50-199	Euclidiana
Montemanni et al. (2005)	2005	ACO	X	50-199	Euclidiana
Branchini et al. (2009)	2009	Granular Local Search	X	1100	Euclidiana
Xu et al. (2013)	2013	IVNS	X	1000	Euclidiana
Necula et al. (2017)	2017	ACO	X	50-199	Euclidiana
Singanamala et al. (2018)	2018	Kmeans, Heurística de C&W	X	13	Experimento
Xu et al. (2018)	2018	Kmeans, EACO	X	225-400	Euclidiana
Fonseca-Galindo et al. (2022)	2022	Árvore de Padrão Frequente	X	300-4000	OSRM
Este Trabalho	**2022**	**DSNR**	X	**300-4000**	**OSRM**

Tabela 2.1: Trabalhos relacionados.

Destaca-se que os métodos utilizados pelas propostas consistem em abordagens para encontrar as soluções adequadas para o problema. O tamanho das instâncias de entrada (tamanho do repositório utilizado) é fundamental para informar se a qualidade da solução é efetiva para problemas com características reais. Na tabela é possível observar que poucos trabalhos atuaram sobre instâncias com milhares de pontos. Da mesma forma, informar como a distância entre os pontos está sendo considerada é de suma importância, uma vez que, atualmente, abordagens baseadas em georeferenciamento estão se destacando por aproximarem de situações e contextos reais.

CAPÍTULO

3

Estratégia Para Problema de Roteamento Dinâmico de Veículos

Este capítulo apresenta o desenvolvimento de estratégias para solução do Problema de Roteamento Dinâmico de Veículos. O algoritmo para roteamento dinâmico é denominado *Dynamic Search per Neighbors Routes* (DSNR) e utiliza uma estratégia de reutilizar rotas previamente abertas por lotes de pacotes. Propõe-se também uma estratégia para o roteamento estático denominada *Kmeans, Relax-and-Fix and Optimizations* (K-RFO) baseada em etapas de gerenciamento de veículos e gerenciamento e pacotes. A Seção 3.1 apresenta o nível de dinamicidade do problema conforme os lotes de pacotes de entrega, a Seção 3.2 apresenta o algoritmo DSNR e a Seção 3.3 apresenta o algoritmo K-RFO.

3.1 Exploração das Estratégias

Na abordagem dinâmica de roteamento de veículos a re-otimização das rotas pode ocorrer desde que a rota não tenha sido despachada por algum dos veículos, o que tornaria fisicamente impossível a alteração de algum pacote. Nesta abordagem, um critério importante é a quantidade máxima de pacotes que podem ser entregues em uma jornada de trabalho. Outro critério é reconhecer o grau de dinamismo que o problema aborda.

O grau de dinamismo da solução proposta irá variar conforme o número de lotes de pacotes recebidos. A Figura 3.1 apresenta o grau de dinamismo conforme a fórmula 2.18, considerando que todos os lotes possuem a mesma quantidade de pacotes. Na perspectiva do primeiro lote de pacotes de entrega

a ser recebido pela estratégia de roteamento, tem-se um problema de roteamento estático, pois não haverá nenhuma rota construída. Na segunda iteração a abordagem se torna moderadamente dinâmica. Na N-ésima iteração, acima de cinco lotes, o problema se torna fracamente dinâmico, onde o grau de dinamismo adotado é apresentado na Subseção 2.3. Para $N \geq 5$ ao substituir na Equação 2.18, o total de demandas será N e o número de demandas dinâmicas será 1. Portanto $\leq 20\%$.

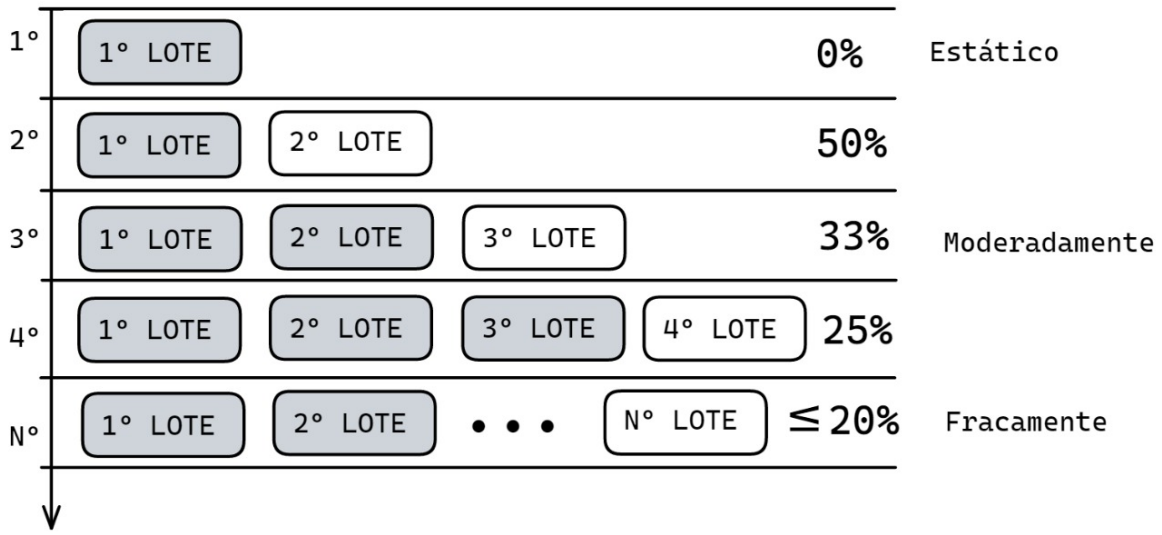


Figura 3.1: Ordem de chegada dos lotes com pacotes.

3.2 Estratégia para Roteamento Dinâmico

Considera-se a abordagem dinâmica conforme o cenário apresentado na Figura 3.2. Um novo lote de entregas traz consigo um novo conjunto de pacotes, representados pela cor preta. Lotes já despachados terão seus pacotes com rotas pré-definidas (mas que ainda não iniciaram o serviço de entrega) na cor branca. Os pacotes que já saíram para entrega serão representados na cor cinza. Cada novo lote libera uma quantidade de pacotes desconhecidos que devem ser alocados para os veículos de entrega. Quando um pacote for atribuído para uma rota (rota nova ou rota aberta), o veículo só será despachado se sua capacidade for suficientemente preenchida.

Nas próximas seções, os dois principais pseudo-algoritmos que compõem a estratégia de solução para o problema de roteamento dinâmico serão explicados: a estrutura de distribuição dos lotes na Seção 3.2.1 e como é feita a inserção dos pacotes dinâmicos na Seção 3.2.2.

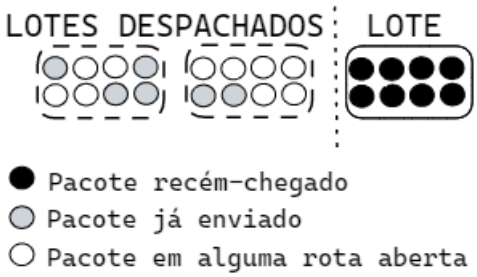


Figura 3.2: Lotes e pacotes para entregas no PRDV.

3.2.1 Distribuição dos Lotes

A estratégia de solução proposta inicia-se com o Algoritmo 1 que recebe como entrada: os pacotes da cor branca, o $LOTE$ com os pacotes pretos e um parâmetro T, o número de pacotes vizinhos considerados ao inserir-se um pacote preto e realizar uma busca de rota vizinha. Entre as linhas 3-15, cada pacote novo (preto) é alocado (branco) até o lote estar totalmente atribuído para as rotas. Na linha 4 o pacote p é retirado do lote. Ao modificar a estrutura do lote, o mesmo é armazenado em uma lista de possíveis configurações de lote, pois, na linha 6 é verificado se não há algum *loop* infinito. Caso seja uma nova configuração do lote, uma busca dos T pacotes mais próximos do pacote p é realizada e as rotas dos respectivos pacotes próximos serão listadas em uma lista de rotas vizinhas (linha 8). Entre as linhas 9-14 é feita uma tentativa de inserção na rota vizinha com o Algoritmo 3, descrito na Subseção 3.2.2. Caso não haja mais rotas vizinhas, uma nova rota é gerada com somente o pacote p e a heurística de melhoria 2Opt** (apresentada na Seção 3.4) é utilizada para verificar se existem trocas entre a nova rota e a rota vizinha mais próxima. Nas linhas 19 e 20 as rotas construídas são substituídas pelas anteriores e um novo pacote dinâmico do lote pode ser inserido, retornando para linha 4.

3.2.2 Inserção do Pacote Dinâmico

A inserção do pacote dinâmico considera um primeiro cenário como ilustrado na Figura 3.3. O melhor cenário é avaliado a partir de todas as possibilidades de inserção em uma rota vizinha, como mostram as linhas 2 e 3 do Algoritmo 2. Após a inserção do pacote dinâmico, existe a possibilidade da rota ser inviável, uma vez que pode exceder a capacidade suportada pelo veículo. Faz-se, então, necessária a consideração de dois possíveis cenários, o que é realizado nas linhas 4 e 5 do Algoritmo 3 e ilustrado na Figura 3.4. O cenário 1 trata de uma rota com o pacote dinâmico e sem o pior pacote da rota (pacote com mais alto *score*), e o cenário 2 considera que o pacote dinâmico não é inserido nesta rota, mantendo a rota original. Com isso, na linha 6 é feita uma comparação de qual rota possui a menor distância, critério este que

Algoritmo 1 Criação de rotas a partir de lotes e pacotes dinâmicos.

```
1: função DISTRIBUIRLOTE(pacotes, LOTE, T)        ▷ pacotes são os pacotes já inseridos no problema e LOTE o lote que precisa ser distribuído, e T um parâmetro de número de pacotes próximos a um pacote dinâmico
2:     configuracoesPossiveis ← LOTE
3:     enquanto NaoEstaVazio(LOTE) faça    ▷ Enquanto LOTE não está vazio
4:         p = TiraPrimeiroPacote(LOTE)
5:         configuracaoAtual ← LOTE
6:         se configuracaoAtual ∉ configuracoesPossiveis então
7:             configuracoesPossiveis ←Adiciona(configuracaoAtual)
8:             rotasVizinhas ←BuscarRotasVizinhas(p, pacotes, T)
9:             para rota ← rotasVizinhas até UltimaRotaVizinha faça
10:                novaRota ←InserirPacote(p, rota, pacotes)
11:                se novaRota = rota então
12:                    BuscaPróximaRotaVizinha
13:                fim se
14:            fim para
15:        fim se
16:        se NaoFoiInseridoEmNenhumaRota(p) então
17:            novaRota = CriaUmaRotaPara(p)
18:            rota1, rota2 = MelhorConf2OPT**(novaRota,rotaMaisProxima)
19:            RotasExcluidas(novaRota, rotaMaisProxima)
20:            RotasIncluidas(rota1, rota2)
21:        fim se
22:    fim enquanto
23: fim função
```

dita qual é a melhor rota. Caso o cenário 1 seja o melhor, o pacote considerado o pior da rota deve ser retirado da rota para que o cenário 1 prevaleça. A avaliação de um pacote dentro de uma rota é feita como mostra a Figura 3.5. O pacote que possuir o maior *score* é o pacote selecionado como pior pacote e expulso da lista de pacotes brancos e enviado para o final da lista dos pacotes pretos. Na linha 8 do Algoritmo 3 é feita uma nova tentativa de inserção do pacote p nesta mesma rota com o intuito de realmente acrescentar o pacote à rota.

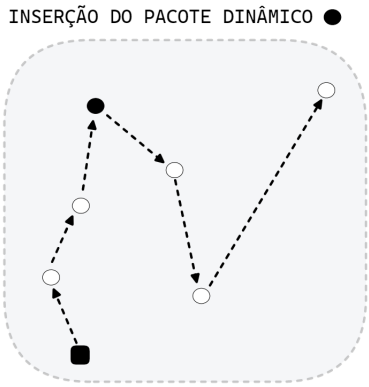


Figura 3.3: Inserção do pacote dinâmico com Algoritmo 2.

Algoritmo 2 Insere um pacote na melhor posição de uma rota, sem se importar com a capacidade.

1: **função** PACOTEINSERIDO($p, rota$) ▷ pacote **p** inserido na melhor posição da rota
2: $possibilidades \leftarrow$PesquisaPossibilidadesDasPosicoes($p, rota$)
3: $melhorRotaComPacket \leftarrow$AvaliaMelhorPossibilidade($possibilidades$)
4: **devolve** $melhorRotaComPacket$
5: **fim função**

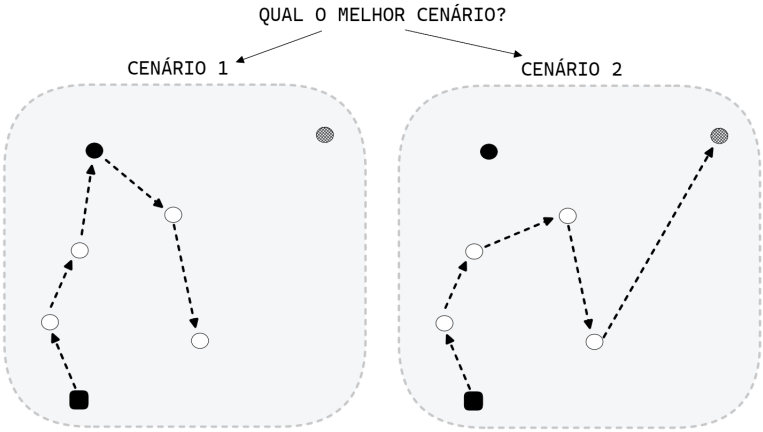


Figura 3.4: Comparação entre os cenários nas linhas 4 e 5 do Algoritmo 3.

A inserção pode ser fracassada caso o cenário 2 prevaleça. Neste caso existem duas novas situações ilustradas na Figura 3.6: 1) há uma rota vizinha

Algoritmo 3 Inserir pacote dinâmico em uma rota vizinha.

```
1: função INSERIRPACOTE(p, rota, pacotes)  ▷ pacote p inserido na rota, pacotes são os pacotes que ainda não foram entregues
2:     auxrota ←PacoteInserido(p, rota)
3:     se CapacidadeExcedida(auxrota) então
4:         cenario1 ← RotaComPacoteDinamicoSemPiorPacote(rota)
5:         cenario2 ← RotaSemPacoteDinamico(rota)
6:         se dist(cenario1) < dist(cenario2) então
7:             ExpulsaPiorPacote(rota, pacotes)
8:             InserirPacote(p, rota)
9:             devolve cenario1
10:        senão
11:            devolve cenario2        ▷ O cenário 2 = rota sem o pacote dinâmico
12:        fim se
13:    senão
14:        devolve auxrota
15:    fim se
16: fim função
```

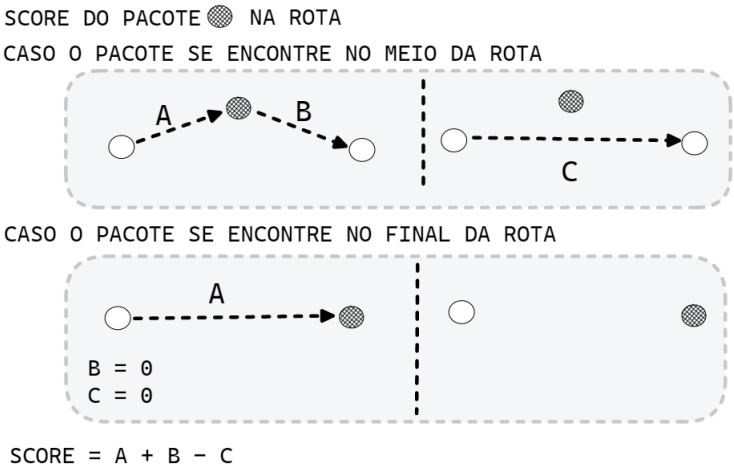


Figura 3.5: Cálculo de *score* de um pacote qualquer em uma rota.

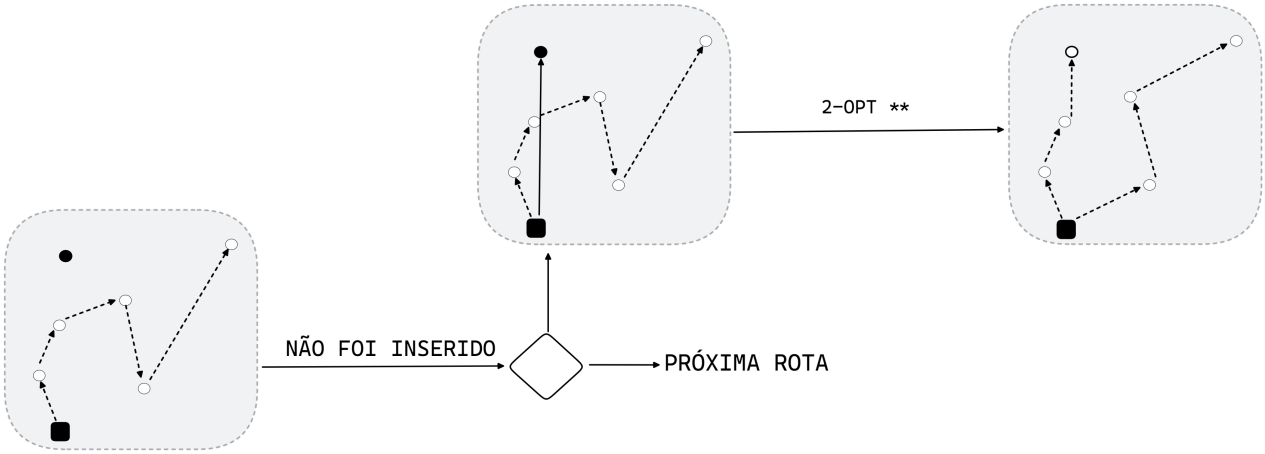


Figura 3.6: Pacote dinâmico não inserido na rota e busca da melhor configuração na linha 18 do Algoritmo 1.

com chance de ser visitada ou 2) não existem mais rotas para visitar. Se não existem mais rotas, cria-se uma rota para o pacote dinâmico e aplica-se a heurística 2-Opt** até encontrar a melhor configuração entre as duas rotas. Vale ressaltar, caso a condição da linha 6 não seja satisfeita, o código retorna para o Algoritmo 1 e a condição da linha 16 é avaliada.

3.3 Estratégia de Roteamento Estático de Veículos

A solução para a abordagem estática, utiliza um algoritmo sob as restrições do PRVC. Conforme o fluxograma apresentado, na Figura 3.7. A primeira etapa realiza um agrupamento com um número limitado de pacotes, conforme a proximidade pela distância Euclidiana. A segunda etapa, é responsável por selecionar os veículos mais adequados para atender cada grupo de pacotes criado. A terceira etapa é um gerenciador de pacotes, que seleciona os pacotes que cada veículo carrega. Na quarta e última etapa, é realizado o roteamento dos pacotes, por meio da heurística *Relax-and-Fix*, encontrando uma solução inicial.

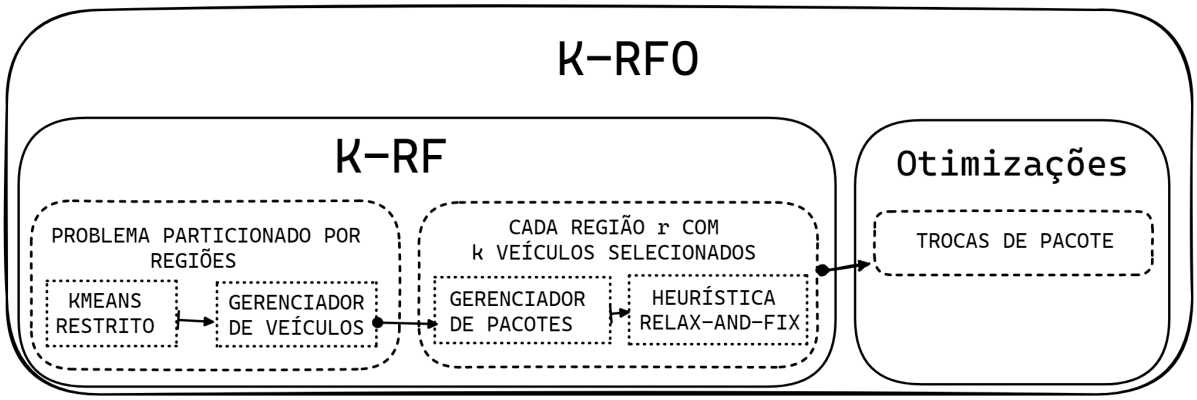


Figura 3.7: Fluxograma do algoritmo K-RFO.

O fluxo composto pelas etapas apresentadas em K-RF gera uma solução inicial para o roteamento de pacotes e atribuição de veículos. Contudo, é inevitável que essa solução inicial necessite de melhorias, uma vez que, a etapa do gerenciador de veículos, limita veículos para roteirizar pacotes de apenas uma região. Dessa forma, as melhorias aplicadas consistem na troca de pacotes entre duas rotas distintas entre si (heurística **2-Opt***). Uma nova melhoria **2-Opt**** implementada consiste em um aumento de possibilidades de troca, onde os dois pacotes selecionados podem pertencer a uma única rota. O fluxograma da Figura 3.7 é detalhado nas Subseções 3.3.1-3.3.4. As heurísticas de melhoria são detalhadas na Seção 3.4.

3.3.1 Kmeans Restrito

O *Kmeans* Restrito (He et al., 2009) aplica uma estratégia diferente para balancear as regiões geradas pelo *Kmeans* utilizando métricas da silhueta para encontrar a melhor partição de K. Na solução de roteamento estático, essa etapa visa restringir as regiões em um intervalo de número mínimo e máximo de pontos explorados por região. Os benefícios de realizar esse procedimento são devidos às seguintes características:

- Particionar o problema em subproblemas balanceados.
- Coligar pacotes próximos, descartando os pacotes distantes para cada subproblema.
- Distribuir veículos com custos e cargas distintas em um próximo passo.

Com alguns efeitos colaterais:

- Aumento de veículos utilizados no problema, devido à informação imprecisa sobre a carga dos pacotes.
- Restringir o percurso dos veículos em uma região.
- Em problemas menores, a clusterização não é bem vista.
- Custo operacional em calcular as regiões.

Para estabelecer o número de pacotes em cada região, é calculado um número médio de pacotes que um veículo consegue carregar. Portanto, é preciso extrair informações de carga dos pacotes já existentes. Neste trabalho, consideram-se as cargas conforme o repositório Loggibud (2021), onde os pacotes possuem cargas unidimensionais variando de 1 até 10. Definiu-se que, em média, os pacotes possuem 5 unidades de carga. Portanto, a quantidade de pacotes em cada região, estimada na variável, ΔA_p, deve pertencer ao intervalo das inequações 3.1

$$\begin{aligned} \lceil 70\%N_p \rceil \leq \quad \Delta A_p \quad &\leq \lceil 120\%N_p \rceil \\ N_p \quad &= \quad \frac{Q_k}{5} \end{aligned} \tag{3.1}$$

Considerando um cenário onde a capacidade máxima de um veículo k é dado por $Q_k = 180$, ou seja, o número médio de pacotes (N_p) que cada veículo transporta é $N_p = 36$, o número mínimo de pacotes de uma região é de 26 e o máximo é 43. Uma vez que cada região é estabelecida, a execução do algoritmo de roteamento é realizada pela etapa do gerenciador de veículos.

3.3.2 Gerenciador de Veículos

O Gerenciador de Veículos é responsável por receber as regiões geradas pelo *Kmeans* Restrito e selecionar os veículos mais adequados para cada região. Consideram-se, também, veículos heterogêneos, que possuem cargas e custos operacionais distintos. Na resolução desta etapa, a função objetivo minimiza a soma dos custos gerados pelos veículos, que partem do depósito e chegam a um centroide, como definido na Função (3.2). Utiliza-se um modelo de otimização com os seguintes dados de entrada, variáveis de decisões, restrições e função objetivo.

Dados de entrada

- R — Regiões indexadas por r.
- K — Veículos indexados por k.
- Q_r — Total de carga dos pacotes da região r.
- D_r — Distância do centroide da região r até o depósito.
- c_k — Custo de transporte do veículo k.
- Q_k — Quantidade máxima que o veículo k consegue carregar.

Variáveis

- $s_k^r \in \{0,1\}$ — Variável binária. Possui valor 1 caso o veículo k pertença à região r, e 0 caso contrário.

$$\min \sum_{r \in R} \sum_{k \in M} D_r s_k^r c_k \tag{3.2}$$

$$\textbf{sujeito a} \quad \sum_{k \in M} s_k^r Q_k \geq Q_r \qquad \forall r \in R \tag{3.3}$$

$$\sum_{r \in R} s_k^r \leq 1 \qquad \forall k \in K \tag{3.4}$$

O custo (Equação (3.2)) relaciona a distância do depósito até o centroide de uma região r, com o custo associado, em utilizar determinado veículo, em uma região. As restrições (3.3), indicam que os veículos utilizados de determinada região devem atender todos os pacotes da região. As restrições (3.4) restringem a possibilidade dos veículos percorrerem mais de uma região. O gerenciamento acontece pela variável s_k^r, que informa os veículos selecionados de cada região. Assim, na saída, são gerados R subproblemas de roteamento de veículos (um subproblema para cada região), gerados a partir do problema original. A partir da próxima etapa, cada uma das regiões é tratada de forma independente e tenta gerenciar e, depois, rotear os pacotes.

3.3.3 Gerenciador de Pacotes

Para cada região é executado o Gerenciador de Pacotes, cujo objetivo é determinar os pacotes selecionados para cada um dos veículos pertencentes à região, sem indicar a ordem de entrega dos pacotes para cada uma das rotas estabelecidas. A principal variável de decisão para esta etapa, é representada por w_j^k, ela indica se o pacote j pertence à rota do veículo k. Os dados de entrada são: uma matriz de distância d_{ij} (depósito na posição 0) e uma estimativa de preço P_j^k para cada pacote j entregue por um veículo k.

O cálculo de P_j^k é dado pela Equação (3.5) e ilustrado na Figura 3.8.

$$P_j^k = c_k(d_{0j} + \sum_{t}^{T} d_{jt}) \tag{3.5}$$

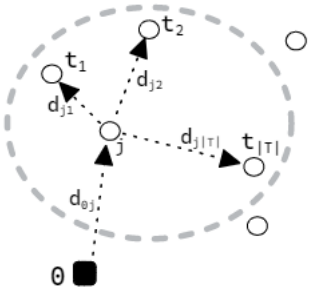


Figura 3.8: Ilustração da distância do veículo j entre seus vizinhos e o depósito.

Dado os custos dos veículos c_k, cada pacote é precificado, conforme a Equação 3.5. Para ilustrar na Figura 3.8, é exemplificada a precificação realizada no pacote j. Onde, depende diretamente do custo de transporte do veículo k e do somatório da distância do pacote j até os T pacotes mais próximos a ele, somado com a distância do depósito até o pacote j (d_{0j}). O tamanho de T é uma variável estabelecida pelo usuário, onde, na Figura 3.8 $T = 3$, vale notar, que o preço P_j^k não é um preço real ou final do pacote, pois, o preço final teria ape-

nas um caminho de chegada e um caminho de saída (com exceção do último pacote, em problemas com o roteamento aberto). A precificação dos pacotes serve para estimar a distância percorrida por determinado pacote, variando os custos de transporte distintos dos veículos. Dada uma matriz de precificação P, tem-se o modelo matemático a seguir.

Dados de Entrada

- P_j^k — Preço estimado para o veículo k entregar o pacote j.
- d_{ij} — Distância para percorrer o arco (i, j), utilizada para calcular a matriz de precificação dos pacotes P.

Variáveis de Decisão

- $w_j^k \in \{0,1\}$ — Variável binária. Possui valor 1 caso o pacote j pertença à rota do veículo k, e 0 caso contrário.

$$\min \sum_{j \in V} \sum_{k \in M} P_j^k w_j^k \tag{3.6}$$

$$\textbf{sujeito a} \quad \sum_{k \in M} w_j^k = 1 \qquad \forall j \in V \tag{3.7}$$

$$\sum_{i \in V} q_j w_j^k \leq Q_k \qquad \forall k \in M \tag{3.8}$$

A função objetivo (3.6) busca minimizar o custo envolvido no problema e utilizar os veículos mais baratos disponibilizados pela etapa 3.3.2. As primeiras restrições (3.7) indicam que todos os pacotes da região devem ser entregues. As restrições (3.8) limitam o número máximo de carga que cada veículo k suporta. A saída esperada do modelo é a variável de decisão w_j^k que informa os pacotes entregues por cada veículo k.

3.3.4 Heurística Relax-and-Fix

Nesta etapa, é recebida a variável de decisão w_j^k da etapa apresentada na Seção 3.3.3, e a utiliza para orientar a disposição dos pacotes na rota de cada veículo. Com a premissa de rotear os pacotes de cada veículo, a heurística *Relax-and-Fix* é utilizada. Com isso, o número de variáveis binárias existentes é reduzida, transformando-as em variáveis reais. O que diminui drasticamente o número de possibilidades para realizar o roteamento dos veículos. Tal heurística, foi explorada nos trabalhos de Santos (2019) e de Menezes (2021), de maneiras diferentes como apresentadas na Seção 2.4.2. A heurística *Relax-and-Fix*, apresentada neste trabalho, é baseada no modelo matemático a seguir.

Dados de Entrada

- p_i — Carga exigida pelo pacote i.
- v_k — Veículos disponibilizados pela saída do Gerenciador de Pacotes.

Variáveis de Decisão

- $x_{ij}^k \in \{0,1\}$ — Variável binária. Possui valor 1 caso o veículo k percorra o arco (i,j), e 0 caso contrário.
- $y_j^k \in \{0,1\}$ — Variável binária. Possui valor 1 caso o pacote j seja o último da rota do veículo k, e 0 caso contrário.
- $z_i^k \in \{0,1\}$ — Variável binária. Possui valor 1 caso o pacote i tenha um próximo destino, e 0 caso contrário.
- u_i^k — Variável auxiliar para eliminação de sub-rotas.

$$\min \sum_{(i,j)\in V^2} \sum_{k\in M} c_k d_{ij} x_{ij}^k \tag{3.9}$$

$$\textbf{sujeito a} \quad \sum_{k\in M} z_i^k + y_i^k = 1 \qquad \forall i \in V' \tag{3.10}$$

$$\sum_{i\in V} x_{ij}^k = w_j^k \qquad \forall k \in M, \forall j \in V' \tag{3.11}$$

$$\sum_{j\in V'} x_{ij}^k = z_j^k \qquad \forall k \in M, \forall i \in V \tag{3.12}$$

$$\sum_{j\in V'} x_{0j}^k = v^k \qquad \forall k \in M \tag{3.13}$$

$$\sum_{i\in V'} x_{i0}^k = 0 \qquad \forall k \in M \tag{3.14}$$

$$u_i^k \geq u_j^k + p_i - \quad Q_k \quad (1 - x_{ij}^k) \qquad \forall k \in M, \forall i,j \in V \tag{3.15}$$

A função (3.9) minimiza o custo percorrido total pela rota, considerando os custos de transporte de cada veículo. As restrições (3.10), indicam que o último pacote atendido pelo veículo k é o último da rota sem retornar ao depósito. As restrições (3.11), indicam se o pacote j é entregue pelo veículo k, o caminho x_{ij}^k deve existir. As restrições (3.12), permitem que um pacote j não tenha um destino definido, no caso da variável z_j^k indicar 0, caso contrário, o pacote j possuí um destino (não é o último pacote da rota). As restrições (3.13), informam que a saída do depósito implica que o veículo é utilizado. As equações (3.14) restringem os veículos de retornar ao depósito. As últimas restrições (3.15) são utilizadas para eliminação de possíveis sub-rotas criadas sem ultrapassar a capacidade máxima dos veículos.

Como visto, na Seção 2.4.2 é exigido um horizonte de planejamento, dividindo em t etapas. Para este caso, a implementação da *Relax-and-Fix* é baseada na variável de decisão x_{ij}^k, portanto, as seguintes etapas são realizadas:

- **Etapa 1:** Relaxe todas as variáveis x_{ij}^k.
- **Etapa 2:** Selecione as variáveis x_{0j}^k, responsáveis por indicar se o veículo k sai do depósito, e as transforme em binárias novamente.
- **Etapa 3:** Resolva o modelo matemático descrito, fixe os valores encontrados em x_{0j}^k e valores impossíveis de acontecer como, por exemplo, $x_{ij}^k = 0$, se $v_k = 0$.
- **Etapa 4:** Selecione as variáveis x_{jh}^k, onde h é qualquer outro pacote que não pertença a outro veículo e j o último pacote inserido no veículo k.
- **Etapa 5:** Resolva o modelo matemático descrito, e fixe os valores encontrados em x_{jh}^k e valores impossíveis de ocorrerem novamente, por exemplo, caso o valor de x_{jh}^k encontrado seja 1, então $x_{hj}^k = 0$ e $x_{ph}^k = 0$, onde p é qualquer outro pacote diferente de j.
- **Etapa 6:** Repita as etapas 4 e 5, considerando que o pacote h agora é j o último pacote da rota, até que todas as variáveis x_{ij}^k sejam resolvidas.

No final do processo, um subproblema terá seu roteamento resolvido. Tendo uma resolução parcial da região trabalhada, após encontrar a solução para todas as regiões é obtida uma solução inicial do problema.

3.4 Heurística de Melhoria 2-Opt**

Após o roteamento dos pacotes em cada veículo utilizado, são aplicados alguns operadores de busca local para explorar soluções melhores, a partir da solução inicial. O primeiro operador utilizado é o 2-Opt* (Fahrion and Wrede (1990)), e o segundo uma versão estendida da mesma (2-Opt**). A heurística 2-Opt* compara a troca de dois pacotes, cada um, em uma rota distinta. Caso a troca dos pacotes melhore a distância percorrida, a troca é realizada. Enquanto a 2-Opt* só verifica a possibilidade de troca, a versão estendida (2-Opt**) acrescenta a alternativa de um dos pacotes ser inserido na rota do outro pacote em sua melhor posição possível. Para ilustrar a versão implementada, considere a representação de como será o roteamento conforme a Figura 3.9. $r1$ representa a primeira rota e $r2$ a segunda rota.

Na Figura 3.9, os pacotes 2 e 3 são selecionados para explorar novos cenários. Ao aplicar a heurística 2-Opt*, um único cenário é possível, a troca

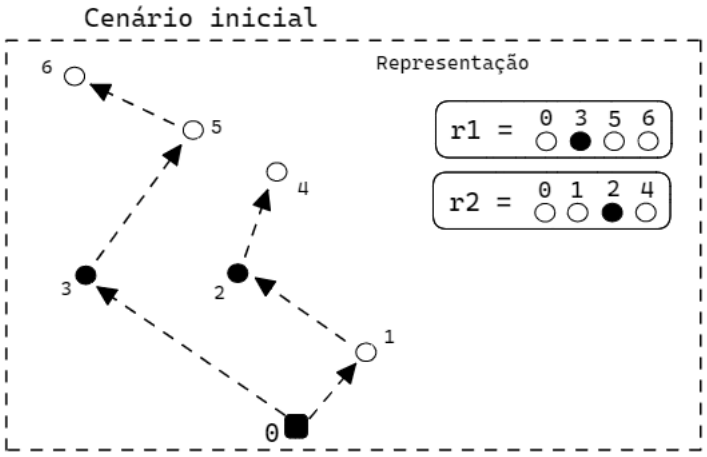


Figura 3.9: Rotas inicias para aplicação da heurística 2-Opt*.

entre os pacotes 2 e 3 (a.) da Figura 3.10. Entretanto, a modificação implementada acrescenta as possibilidades (b.-i.) de ambos os pacotes selecionados serem roteirizados em uma única rota e avalia a melhor posição em que um dos pacotes selecionados possa ser inserido na outra rota.

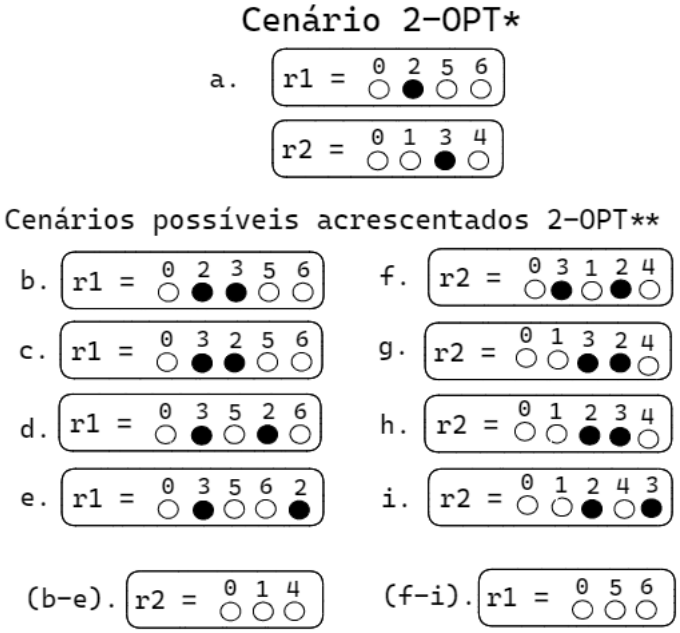


Figura 3.10: Cenários da 2-Opt**.

CAPÍTULO

4

Resultados

Este capítulo apresenta e discute os principais resultados obtidos com a aplicação dos algoritmos desenvolvidos neste trabalho. Nos experimentos realizados, as instâncias de entrada foram obtidas a partir do *benchmark* Loggibud (2021). Esse *benchmark* também disponibiliza algoritmos para solução tanto do PRV quanto do PRDV. Esses algoritmos foram utilizados para fins de validação e avaliação dos algoritmos propostos neste trabalho.

4.1 Repositório Loggibud

O repositório Loggibud considera que o processo de entrega segue um fluxo conforme apresentado em Fonseca-Galindo et al. (2022) que descreve três etapas para que uma encomenda possa ser encaminhada de um depósito até um destino. A primeira etapa (Primeira Milha) é responsável pela entrega dos produtos para os centros de distribuições (DC). A segunda etapa (Milha Intermediária) é responsável por retirar os produtos dos depósitos principais e enviar para centros de expedições (CEs). Por fim, a Última Milha é onde cada CE fica responsável por resolver um PRV, assim determinando as rotas para uma frota de veículos realizar a entrega dos produtos para os clientes (Figura 4.1).

Este capítulo apresenta resultados de roteamento de veículos com enfoque na resolução da Última Milha. Os pacotes de entrega que chegam ao DC são destinados para os CEs. Em uma abordagem estática do PRV, os CEs recebem todos os pacotes que precisam ser entregues e, após isso, é realizada a atribuição de veículos e o roteamento. Em uma abordagem dinâmica, considera-se que os CEs possuem restrições de tempo e de armazenamento,

de forma que todos os pacotes não podem ser armazenados no CE antes de realizar a atribuição de veículos e o roteamento. Nesse contexto dinâmico, atribui-se a cada pacote, dentro do CE, um destino temporário, antes mesmo de alocar os pacotes para os veículos.

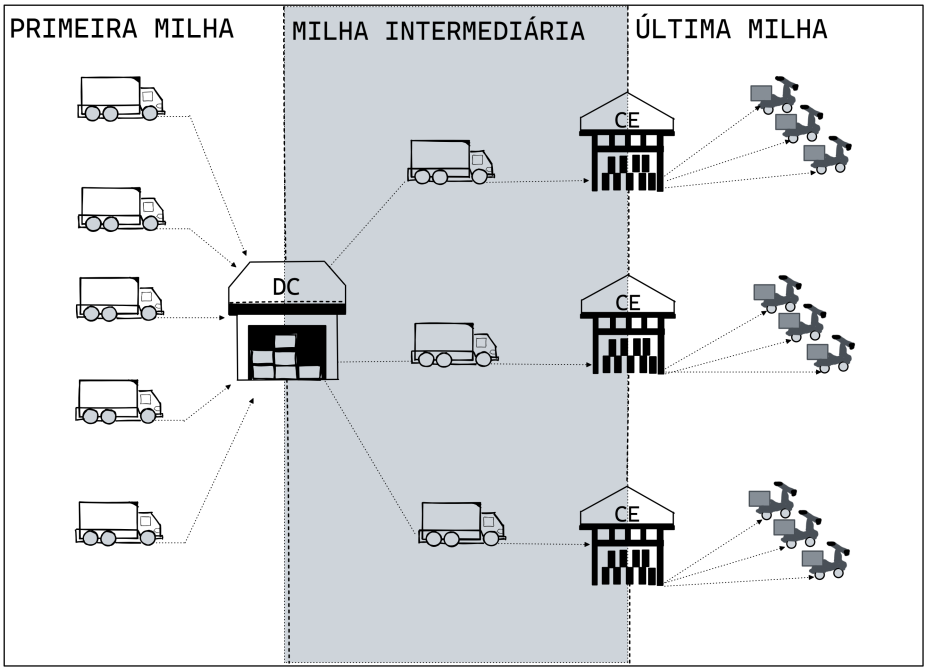


Figura 4.1: Fluxo de encaminhamento de um produto a partir do Centro de Distribuição (DC), passando pelo Centro de Expedição (CE) até a entrega ao cliente (Fonseca-Galindo et al., 2022).

O Loggibud (2021) é um repositório disponibilizado e mantido pela empresa Loggi® que contém instâncias de entrada com milhares de localizações de entrega, para fins de testes e validações com algoritmos de roteamento. Adicionalmente, o repositório disponibiliza uma infraestrutura de *benchmark* para avaliações e testes com algoritmos para os problemas de roteamento estático e dinâmico de veículos.

Os dados de entrada disponibilizados pelo *benchmark* possuem a coordenada georreferenciada dos pedidos e a quantidade de itens dos mesmos. Assim, devem ser considerados fatores de trânsito como sentido das ruas, ruas sem saídas, entre outros fatores que possam modificar as rotas. Tais fatores são considerados por uma *Application Program Interface* (API) disponibilizada pelo servidor *Open Source Routing Machine* (OSRM)[2], que fornece um serviço para calcular a menor distância entre dois pontos georreferenciados.

Para melhor ilustração do arquivo de entrada, a Figura 4.2 apresenta um mapa com as localizações do depósito (CE) e das entregas de quatro pedidos. Os números 7, 3, 5 e 8 referem-se à quantidade de itens de cada pedido. No arquivo de saída (Figura 4.3), tem-se, para cada veículo, uma sequência de

[2]Link para api: https://project-osrm.org.

clientes com a rota do veículo estabelecida.

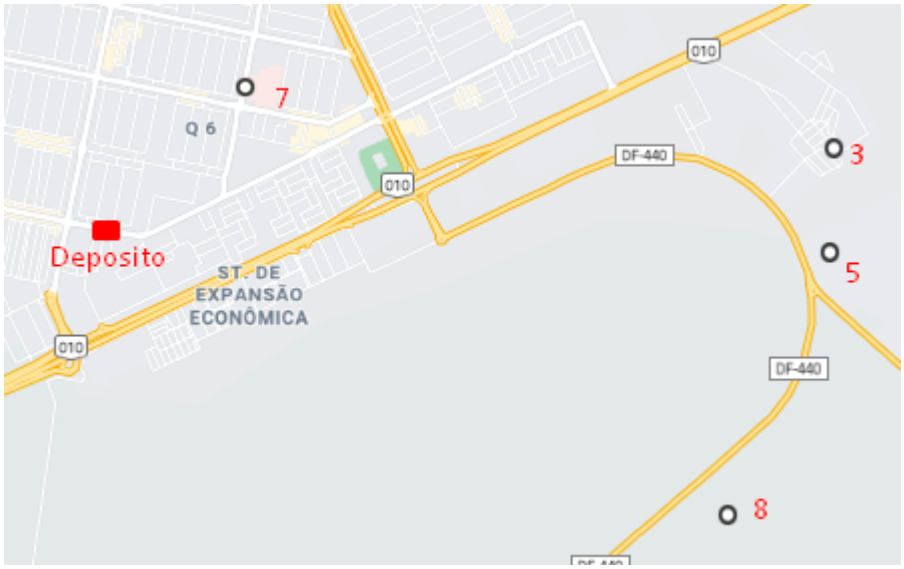


Figura 4.2: Exemplo de arquivo de entrada: localização dos pedidos e quantidade de itens a serem entregues.

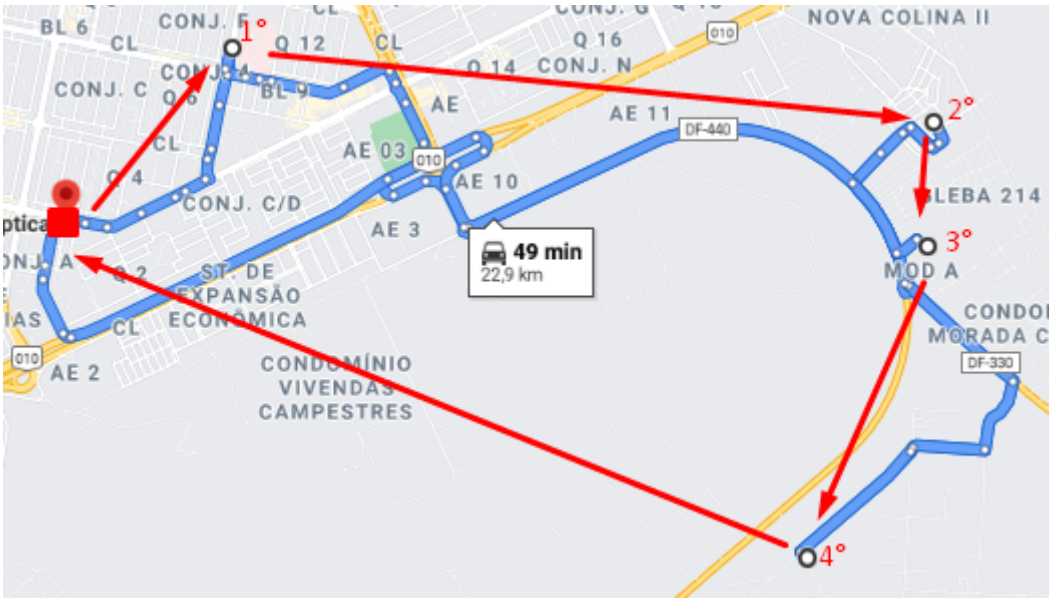


Figura 4.3: Exemplo de arquivo de saída: rota para entrega dos itens dos pedidos.

4.1.1 *Algoritmos do Benchmark Loggibud para o Roteamento Estático de Veículos*

O repositório possui a implementação de 3 algoritmos de roteamento estático. Tais algoritmos consideram que todos os pacotes de entrega são conhecidos *a priori* para atribuição dos veículos e construção de rotas:

- **Lin-Kernighan Helsgaun-3 (LKH-3)** O LKH-3 (Helsgaun, 2017) é uma variação generalizada da heurística LKH utilizada para resolver o TSP.

O algoritmo funciona a partir de particionamentos dos clientes (locais de entrega dos pacotes). Dado um grafo $G(V,E)$, onde V é o conjunto de vértices e E o conjunto de arestas, o conjunto V é dividido em dois subconjuntos quaisquer V_1 e V_2. Após isso, é feita uma avaliação de qual a melhor troca de vértices do subconjunto V_1 com V_2, aqueles dois vértices que tiverem o maior ganho são trocados de subconjuntos. Isso é feito até ter as melhores partições do problema. Para generalizar esta ideia para o PRV é considerado o número de veículos associados ao problema. Dado o somatório das cargas dos pacotes dividido pela carga máxima dos veículos, tem-se a quantidade mínima de rotas necessárias para resolver o problema. O particionamento dos pacotes nessas rotas é aplicado de forma que o número total de pacotes é dividido pelo número de rotas. Com isso, há vários subconjuntos com pequena variação na quantidade de pacotes. Então, o foco passa a ser minimizar o custo das rotas através das trocas entre os subconjuntos.

- **K*means-Partition* (KP)** Este algoritmo é baseado na implementação desenvolvida por He et al. (2009) mas sem a componente de balancear as regiões. Primeiro, o algoritmo particiona o problema em K regiões convexas e depois em cada região é usado o *solver OR-Tools* disponibilizado pela *Google Developer* para fornecer uma solução para roteamento de veículos. Os otimizadores internos utilizados pelo software são o Glop[3] e o SCIP[4].

- **K*means-Aggregation* (KA)** Da mesma forma que o KP, o algoritmo também é baseado nos conceitos de He et al. (2009) e particiona o problema em K regiões convexas. Entretanto, no lugar de aplicar o *Kmeans* de forma balanceada, aplica-se o algoritmo *Mini Batch Kmeans* que particiona primeiro o problema em pequenas regiões e realiza uma segunda camada de clusterização. O pressuposto é particionar o problema ainda mais estabelecendo o tamanho de um veículo em cada uma das regiões criadas pelo algoritmo *Mini Batch Kmeans*, e resolver o roteamento com o *OR-Tools*.

4.1.2 Algoritmos do Benchmark Loggibud para o Roteamento Dinâmico de Veículos

No repositório estão implementados 2 algoritmos que resolvem o problema de roteamento dinâmico:

[3]Otimizador desenvolvido pela Google.
[4]Otimizador disponível em https://www.scipopt.org/.

- ***QRP-Sweep* (QRPS)** Este algoritmo consiste em um particionamento de regiões capacitadas com um método de varredura (Bertsimas and Van Ryzin, 1993; Gillett and Miller, 1974). Primeiro, há uma fase de treinamento baseado nos dados históricos, onde as demandas de entrega têm suas coordenadas planares (x,y) convertidas em coordenadas polares (R,θ), com os ângulos θ no intervalo de $[-180,180]$ e $R=\sqrt{x^2+y^2}$. Com as coordenadas convertidas é aplicado o *Kmeans* sobre as coordenadas polares, obtendo regiões em formato de pizza onde o centro é o depósito. Geradas as regiões, cada pacote inserido tem seu destino classificado pelas regiões geradas na fase de treinamento. Assim que um veículo tiver alcançado sua capacidade máxima de pacotes, ou que historicamente não existam mais pacotes a serem enviados naquela região, o veículo responsável pela região é despachado. Antes de cada veículo ser despachado, resolve-se um TSP com os otimizadores do *OR-Tools* para cada um deles.

- **K*means-Greedy* (KG)** O algoritmo KG utiliza um treinamento baseado em dados históricos. A partir desses dados é possível criar padrões de regiões (He et al., 2009). Após o treinamento, o algoritmo insere um pacote de cada vez e a cada inserção o pacote é redirecionado para a região com o centroide mais próximo. Para cada uma das regiões é considerada um veículo distinto. Quando a região atinge um limite de pacote que poderia receber (conforme o treinamento), o veículo desta região é despachado. Da mesma forma que o QRPS, para cada veículo despachado aplica-se o *solver OR-Tools* para roteirizar os pacotes desta região gerada.

4.2 Experimento e Resultados com Algoritmos de Roteamento Dinâmico

A estratégia de roteamento dinâmico, proposta neste trabalho, pode receber um conjunto de pedidos de entrega (ou pacotes) em lotes, que pode estar organizado de acordo com três modos de ordenação. Esses modos estão relacionados com a ordem em que os pacotes são inseridos nas rotas. Considere que cada pacote é representado pelo índice i e possui um "peso" associado. O peso consiste na distância média desse pacote em relação aos demais pacotes do lote. Essa distância de um pacote i para um pacote j é dada por d_{ji}. Todos os pacotes, incluindo o depósito, pertencem ao conjunto V. O peso pode ser calculado pela Equação 4.1.

$$peso_i = \frac{\sum_{j \in V} d_{ji}}{|V|} \tag{4.1}$$

As ordenações podem ser:

- **Sem ordem (S):** Os pacotes são inseridos nas rotas conforme a ordem que aparecem nos lotes.
- **Ordem crescente (C):** Os pacotes do lote são inseridos nas rotas com os pesos em ordem crescente.
- **Ordem decrescente (D):** Os pacotes do lote são inseridos nas rotas com os pesos em ordem decrescente.

Para determinar a estratégia de ordenação de pacotes mais adequada, considera-se uma vizinhança igual a 30 $(T = 30)$, indicando que o cálculo do peso é realizado a partir dos 30 pacotes mais próximos do pacote (i) atual. Para determinar se as organizações de pacotes exercem algum impacto nos resultados do roteamento, os experimentos foram realizados alterando as ordenações dos pacotes. As Tabelas 4.1, 4.2 e 4.3 apresentam os resultados[5] de distância percorrida pelos veículos e o número máximo de veículos utilizados, para instâncias representando pontos de entrega de três cidades brasileiras: Bélem (PA), Distrito Federal (DF) e Rio de Janeiro (RJ).

Para cada tabela, a primeira coluna informa o nome da instância de entrada. As colunas DSNR-S-30, DSNR-D-30 e DSNR-C-30 representam, respectivamente, os resultados obtidos pelo algoritmo dinâmico *Dynamic Search per Neighbors Routes* (DSNR) utilizando a organização **sem ordem**, **ordem decrescente** e **ordem crescente**. Os resultados apresentam as diferenças que ocorrem de cidade para cidade em relação com as organizações. Os melhores resultados em distância percorrida e número de veículos estão destacados em negrito. No caso da distância percorrida ter uma diferença de, menor ou igual que $1km$ considera-se como melhor estratégia aquela que utiliza menos veículos. Para precificar os resultados de distância e número de veículos utilizados, as seguintes restrições foram consideradas:

- A distância total percorrida é dada em km;
- Cada instância representa 1 dia de trabalho;
- Os veículos possuem um custo de transporte médio de $R\$\ 1,92$ por km;
- O custo mínimo para os veículos saírem do depósito e entregarem ao menos 1 pacote é $R\$\ 13,90$.

[5] A DSNR foi executada conforme o código desenvolvido em https://github.com/Wiltoon/optimizeRoutesIntermediary.

	DSNR-S-30		DSNR-D-30		DSNR-C-30	
Instância	**Distancia P.**	**N veículos**	**Distancia P.**	**N veículos**	**Distancia P.**	**N veículos**
pa-90	608,0738	10	**594,8412**	**10**	669,7983	10
pa-91	706,3666	10	**699,3135**	**10**	769,5035	10
pa-92	**520,9199**	**10**	700,7986	10	533,984	10
pa-93	552,3717	10	593,3302	10	**513,0934**	**10**
pa-94	806,66	10	**733,3746**	**10**	806,6008	10
pa-95	**675,6512**	**10**	854,2433	10	804,4405	10
pa-96	836,7248	10	**784,3198**	**10**	843,2281	10
pa-97	**649,9002**	**10**	721,6059	10	698,2187	10
pa-98	780,0712	10	**696,0151**	**10**	765,2644	10
pa-99	**761,088**	**10**	908,8439	10	806,0248	10
pa-100	726,5846	10	**720,4573**	**10**	684,7977	10
pa-101	749,0886	10	745,8422	10	**703,1705**	**10**
pa-102	**627,6399**	**10**	769,2978	10	654,6388	10
pa-103	**598,0809**	**10**	601,3425	10	652,095	10
pa-104	**665,8891**	**10**	677,2377	10	726,9242	10
pa-105	869,7569	10	**697,618**	**10**	707,5161	10
pa-106	596,5353	10	667,5203	10	**581,5426**	**10**
pa-107	681,7921	10	823,2907	10	**558,7434**	**10**
pa-108	694,3758	10	**628,4816**	**10**	753,9011	10
pa-109	**610,1751**	**10**	662,7493	10	729,322	10
pa-110	632,9731	10	752,1817	10	**611,8721**	**10**
pa-111	708,7428	10	**635,5543**	**10**	714,0625	10
pa-112	783,9864	10	747,5442	10	**604,0927**	**10**
pa-113	**555,3273**	**10**	597,9199	10	651,7476	10
pa-114	**802,998**	**10**	825,9361	10	938,9519	10
pa-115	773,4343	10	700,176	10	**554,7784**	**10**
pa-116	574,6314	10	**560,9274**	**10**	565,5234	10
pa-117	**658,2698**	**10**	863,486	10	827,127	10
pa-118	738,168	10	**678,8788**	**10**	709,1594	10
pa-119	775,0179	10	682,4062	10	**646,5393**	**10**

Tabela 4.1: Comparação entre a organização do lote antes da entrega para cidade do Belém (PA).

	DSNR-S-30		DSNR-D-30		DSNR-C-30	
Instância	**Distancia P.**	**N veículos**	**Distancia P.**	**N veículos**	**Distancia P.**	**N veículos**
df-90	1465,3602	46	**1465,6016**	**44**	**1448,1838**	45
df-91	**1293,2252**	39	1344,3693	**37**	1331,8094	41
df-92	**1394,3307**	43	1402,7281	42	1409,9995	**40**
df-93	1356,5156	40	**1255,6552**	39	1269,2078	**38**
df-94	1621,5628	34	**1442,2466**	**31**	1693,7373	35
df-95	1582,2261	50	1602,6236	**48**	**1540,0608**	51
df-96	**1508,2228**	41	1647,593	**39**	1795,3243	39
df-97	**1303,7028**	39	1396,0494	**34**	1344,2486	39
df-98	1321,3048	**37**	**1261,3515**	38	1355,5237	41
df-99	1519,4293	31	1455,4398	**30**	**1441,562**	36
df-100	**1667,2149**	40	1928,9329	**39**	1748,5747	41
df-101	1522,286	57	**1415,8384**	47	1482,8491	**46**
df-102	1650,435	57	1512,146	50	**1500,3872**	**48**
df-103	1782,8452	44	1780,2872	46	**1709,727**	**43**
df-104	1543,4103	48	1585,4688	47	**1504,0321**	**47**
df-105	1550,6522	**26**	1400,6085	33	**1353,4692**	31
df-106	1621,0524	**46**	**1498,2364**	48	1562,8578	50
df-107	1794,7891	57	1706,7682	**54**	**1704,3721**	56
df-108	1649,9361	54	**1520,1903**	**48**	1552,6962	48
df-109	**1527,1772**	**37**	1658,4396	38	1550,3042	40
df-110	**1733,4873**	**38**	1754,5272	45	1970,5553	42
df-111	**1475,1695**	**46**	1565,3899	48	1555,1915	48
df-112	**1471,6955**	47	1578,3984	**45**	1539,5788	48
df-113	**1386,3061**	**32**	1633,7371	36	1397,083	35
df-114	**1694,4023**	36	1887,5359	**35**	1742,7437	36
df-115	1457,3697	51	1446,1815	47	**1373,3319**	**46**
df-116	1477,4656	40	**1432,1601**	**35**	1489,7657	38
df-117	**1239,546**	41	1261,3678	37	1303,8753	**35**
df-118	1574,3701	**31**	1593,2986	31	**1523,7405**	34
df-119	1438,2957	37	1539,0595	37	**1379,9869**	**31**

Tabela 4.2: Comparação entre a organização do lote antes da entrega para o Distrito Federal (DF).

	DSNR-S-30		DSNR-D-30		DSNR-C-30	
Instância	**Distancia P.**	**N veículos**	**Distancia P.**	**N veículos**	**Distancia P.**	**N veículos**
rj-90	**3896,2953**	**188**	3919,5326	193	4007,879	207
rj-91	4192,269	220	**3978,6227**	**205**	4134,6098	213
rj-92	4077,4316	**203**	**4019,7732**	207	4185,5283	213
rj-93	4017,9277	172	**3963,9913**	**158**	4118,2273	162
rj-94	**3881,8477**	**161**	3887,8537	164	3998,1586	171
rj-95	3455,3968	**113**	3451,9345	117	**3432,9097**	121
rj-96	3980,8674	157	**3801,6723**	147	3994,0159	**146**
rj-97	3371,5937	171	**3227,1748**	**156**	3498,3893	170
rj-98	3985,3786	**169**	**3979,583**	172	4096,3282	183
rj-99	**4221,7716**	**215**	4241,0023	217	4385,9949	230
rj-100	3858,4346	197	**3851,1248**	**196**	3895,6661	202
rj-101	**3575,4646**	118	3811,775	**112**	3636,2391	122
rj-102	**3428,8985**	**145**	3491,9486	146	3522,2415	154
rj-103	4448,8397	159	**4308,4206**	**157**	4337,7036	161
rj-104	**3298,0906**	**163**	3429,3289	168	3383,5743	167
rj-105	**4380,1876**	154	4612,3957	**151**	4410,5976	158
rj-106	3818,5585	**188**	**3802,6405**	194	3892,0807	204
rj-107	3619,4763	175	3607,0298	**172**	**3598,0252**	181
rj-108	3680,893	136	**3672,2694**	**129**	3818,3747	134
rj-109	3768,5167	183	3779,6604	191	**3712,6553**	**182**
rj-110	3476,6363	129	**3470,1277**	138	3475,6446	**127**
rj-111	**4150,5181**	**143**	4152,8012	144	4411,021	149
rj-112	**3742,6716**	**148**	3870,5992	150	3927,5087	148
rj-113	4378,65	**154**	**4315,9692**	157	4356,5558	156
rj-114	**4051,3221**	**201**	4090,3314	204	4132,4214	203
rj-115	4256,4052	157	4280,9831	156	**4144,2994**	**155**
rj-116	3755,1026	**139**	**3583,7977**	140	3788,0412	142
rj-117	4133,0698	217	**4073,8643**	**212**	4092,9941	213
rj-118	**4291,2398**	**221**	4301,624	226	4311,4067	228
rj-119	3808,9679	**191**	**3723,0957**	194	3743,6085	197

Tabela 4.3: Comparação entre a organização do lote antes da entrega para cidade do Rio de Janeiro (RJ).

Considerando a utilização de veículos homogêneos, a seguinte equação de custo total associado às entregas[6] foi adotada:

$$Custo_{Total} = R\$1,92 * Distancia_{Percorrida} + R\$13,90 * N_{Veiculos} \quad (4.2)$$

A Tabela 4.4 apresenta os custos calculados para os conjuntos de instâncias PA, DF e RJ conforme as distâncias e número de veículos apresentados nas Tabelas 4.1, 4.2 e 4.3.

	DSNR-S-30	**DSNR-C-30**	**DSNR-D-30**
CIDADE	**CUSTO MENSAL**		
PA-300	**R$ 43.954,89**	R$ 45.115,03	R$ 44.080,39
DF-1000	R$ 105.181,17	R$ 105.335,88	**R$ 104.850,78**
RJ-4000	R$ 295.354,53	**R$ 294.580,48**	R$ 299.676,08

Tabela 4.4: Custos para os conjuntos de instâncias alterando a organização inicial do lote de pacotes.

Observa-se a existência de uma pequena diferença de custo entre os tipos de organizações dos pacotes. Nesse sentido, testes estatísticos foram aplicados visando avaliar se a diferença estatística é significativa entre esses custos. O teste de normalidade de Shapiro-Wilk[7] foi aplicado sobre os custos dos três conjuntos de instâncias (PA, DF e RJ) indicando não haver efeitos significativos para rejeitar a hipótese de normalidade desses custos ($p-values > \alpha$), considerando um nível de significância de 5% ($\alpha = 0.05$). Após a validação da normalidade dos dados, o teste ANOVA[8] foi aplicado sobre cada conjunto de instâncias visando avaliar se há diferença significativa entre as organizações. O menor $p-value$ obtido pelo teste ANOVA foi $p-value > 0.41$ o que demonstra estatisticamente que não há efeito (pois o $p-value > \alpha$, $\alpha = 0.05$) nos dados para rejeitar a hipótese nula de igualdade entre as organizações dos pacotes. Portanto, optou-se por adotar a organização **Sem Ordem (S)** e utilizá-la para comparar com os outros algoritmos dinâmicos disponibilizados.

As Tabelas 4.5, 4.6 e 4.7 apresentam os resultados do algoritmo DSNR com organização sem ordem, e os algoritmos dinâmicos *QRP-Sweep* e K*means-Greedy*[9] disponíveis no *benchmark* Loggibud com as instâncias PA, DF e RJ. Nesse experimento, considerou-se que cada lote possui, no máximo, 75 pacotes. Para os algoritmos QRPS e KG considerou-se que o número de clusters utilizado para realizar o pré-treinamento dos dados foi o número mínimo de veículos que poderiam ser utilizado em uma instância de treinamento.

[6]Custos médios extraídos do site: https://www.montarumnegocio.com/quanto-ganha-um-motoboy-na-loggi/. Custos informados em 3 de agosto de 2020.

[7]https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_*test*

[8]https://en.wikipedia.org/wiki/Analysis_*of*_*variance*.

[9]Os algoritmos dinâmicos do repositório Loggibud foram executadas conforme a task2 do repositório https://github.com/loggi/loggibud.

	DSNR		***QRP-Sweep***		***Kmeans-Greedy***	
Instância	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**
pa-90	**608,0738**	**10**	718,2511	12	1198,7025	82
pa-91	**706,3666**	**10**	876,7989	12	1255,2553	81
pa-92	**520,9199**	**10**	816,7764	12	1322,2747	86
pa-93	**552,3717**	**10**	622,4138	12	1182,784	81
pa-94	**806,66**	**10**	807,3758	11	1295,3043	85
pa-95	**675,6512**	**10**	917,3683	13	1548,1802	88
pa-96	**836,7248**	**10**	952,1333	12	1524,2713	90
pa-97	**649,9002**	**10**	754,703	13	1135,2874	76
pa-98	780,0712	**10**	**772,7976**	12	1215,2671	78
pa-99	**761,088**	**10**	912,8184	12	1564,6098	88
pa-100	**726,5846**	**10**	831,1861	12	1283,9829	83
pa-101	**749,0886**	**10**	822,7584	12	1289,8666	83
pa-102	**627,6399**	**10**	736,1106	12	1281,8988	83
pa-103	**598,0809**	**10**	716,1063	12	1192,7135	84
pa-104	**665,8891**	**10**	895,8862	12	1443,4891	87
pa-105	**869,7569**	**10**	942,4216	13	1399,9278	84
pa-106	**596,5353**	**10**	792,3434	13	1311,9928	83
pa-107	**681,7921**	**10**	836,563	12	1301,1002	82
pa-108	**694,3758**	**10**	960,7496	12	1428,8094	87
pa-109	**610,1751**	**10**	756,2436	12	1119,5802	76
pa-110	**632,9731**	**10**	784,4334	12	1339,1521	82
pa-111	**708,7428**	**10**	771,8653	12	1422,0197	84
pa-112	**783,9864**	**10**	843,2348	13	1477,686	84
pa-113	**555,3273**	**10**	711,5191	13	1205,8396	80
pa-114	**802,998**	**10**	829,7362	12	1368,0337	83
pa-115	**773,4343**	**10**	839,3213	13	1276,9746	83
pa-116	**574,6314**	**10**	752,579	10	1131,9186	72
pa-117	**658,2698**	**10**	826,5882	13	1345,7334	87
pa-118	**738,168**	**10**	865,0067	12	1372,046	88
pa-119	**775,0179**	**10**	823,3594	12	1347,6168	89

Tabela 4.5: Resultados das instâncias do Belém (PA) com aproximadamente 300 pacotes em cada instância.

	DSNR		*QRP-Sweep*		*Kmeans-Greedy*	
Instância	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**
df-90	**1465,3602**	**46**	1748,4614	46	2549,9019	107
df-91	**1293,2252**	39	1540,5484	**37**	2231,7731	100
df-92	**1394,3307**	43	1681,646	**41**	2472,4867	108
df-93	**1356,5156**	40	1388,4163	**36**	2053,5336	92
df-94	**1621,5628**	**34**	1821,3157	42	2729,7832	110
df-95	**1582,2261**	50	1986,7212	**45**	2775,0112	114
df-96	**1508,2228**	**41**	1932,8853	46	2419,4917	106
df-97	**1303,7028**	39	1535,7124	**36**	2449,8851	106
df-98	**1321,3048**	**37**	1594,9961	38	2362,3821	103
df-99	**1519,4293**	**31**	1626,7085	39	2180,4183	98
df-100	**1667,2149**	**40**	2086,4619	48	2585,2083	109
df-101	**1522,286**	57	1790,1212	**47**	2315,8798	108
df-102	**1650,435**	57	1864,2116	**46**	2458,5582	109
df-103	**1782,8452**	**44**	2011,2075	48	2701,8874	114
df-104	**1543,4103**	48	1869,2803	**44**	2343,8053	104
df-105	1550,6522	**26**	**1532,9156**	36	2093,7692	100
df-106	**1621,0524**	46	1878,0686	**45**	2566,8251	109
df-107	**1794,7891**	57	2066,8405	**48**	2880,8175	117
df-108	**1649,9361**	54	1915,0758	**45**	2502,7153	105
df-109	**1527,1772**	**37**	1747,2256	43	2567,339	107
df-110	**1733,4873**	**38**	2034,7945	49	2708,6926	110
df-111	**1475,1695**	46	1950,9422	**45**	2664,1968	110
df-112	**1471,6955**	**47**	1984,9644	48	2605,6492	114
df-113	**1386,3061**	**32**	1602,0718	39	2405,6912	106
df-114	**1694,4023**	**36**	1990,5469	46	2698,2092	111
df-115	**1457,3697**	51	1784,1214	**42**	2410,7356	105
df-116	**1477,4656**	**40**	1678,1917	42	2282,5852	105
df-117	**1239,546**	41	1540,8555	**39**	2206,2277	99
df-118	**1574,3701**	**31**	1814,2064	40	2448,7881	108
df-119	**1438,2957**	**37**	1683,8156	40	2542,3668	109

Tabela 4.6: Resultados das instâncias do Distrito Federal (DF) com aproximadamente 1000 pacotes em cada instância.

	DSNR		*QRP-Sweep*		*Kmeans-Greedy*	
Instância	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**
rj-90	**3896,2953**	**188**	5655,5301	192	4528,968	216
rj-91	**4192,269**	220	5869,8616	**199**	4605,4447	216
rj-92	**4077,4316**	203	5732,8241	**199**	4575,7895	212
rj-93	**4017,9277**	**172**	5840,2369	202	4523,0917	215
rj-94	**3881,8477**	**161**	5344,9899	191	4327,3124	201
rj-95	**3455,3968**	**113**	4809,4203	160	3971,0325	184
rj-96	**3980,8674**	**157**	5560,2666	198	4379,8703	207
rj-97	**3371,5937**	171	4943,7088	**166**	4102,1336	190
rj-98	**3985,3786**	**169**	5628,1677	205	4463,2336	216
rj-99	**4221,7716**	215	5754,5064	**203**	4629,7398	219
rj-100	**3858,4346**	**197**	5775,4154	202	4472,6611	212
rj-101	**3575,4646**	**118**	4895,2905	163	4106,4834	191
rj-102	**3428,8985**	**145**	5141,8823	175	4213,6263	200
rj-103	**4448,8397**	**159**	5841,0507	206	4797,8711	227
rj-104	**3298,0906**	163	4579,0322	**156**	3938,106	185
rj-105	**4380,1876**	**154**	5909,0192	206	4667,8273	223
rj-106	**3818,5585**	**188**	5571,1459	196	4455,5092	215
rj-107	**3619,4763**	175	4969,7073	**174**	4220,1804	197
rj-108	**3680,893**	**136**	5083,1271	181	4144,6036	195
rj-109	**3768,5167**	**183**	5285,2285	192	4320,6089	207
rj-110	**3476,6363**	**129**	4848,6456	173	4041,1912	191
rj-111	**4150,5181**	**143**	5866,5579	201	4731,9292	215
rj-112	**3742,6716**	**148**	5594,8388	191	4564,3588	218
rj-113	**4378,65**	**154**	5835,0348	211	4769,6051	227
rj-114	**4051,3221**	**201**	5953,4524	201	4646,9258	217
rj-115	**4256,4052**	**157**	5863,9875	202	4619,3945	222
rj-116	**3755,1026**	**139**	5243,5598	180	4363,8498	203
rj-117	**4133,0698**	217	5820,6683	**201**	4641,3202	220
rj-118	**4291,2398**	221	5942,4162	**206**	4790,1975	229
rj-119	**3808,9679**	**191**	5344,5916	192	4333,6692	207

Tabela 4.7: Resultados das instâncias do Rio de Janeiro (RJ) com aproximadamente 4000 pacotes em cada instância.

A Tabela 4.8 apresenta os custos totais de entrega, para cada conjunto de instâncias, das três estratégias dinâmicas avaliadas. Observa-se que a estratégia proposta neste trabalho (DSNR) possui custos finais significativamente menores quando comparados com os algoritmos *QRP-Sweep* (QRPS) e K*means-Greedy* (KG).

	DSNR	***QRP-Sweep***	**K*means-Greedy***
CIDADES	**CUSTO MENSAL**		
PA-300	**R$ 43.954,89**	R$ 52.093,24	R$ 110.734,15
DF-1000	**R$ 104.850,78**	R$ 120.947,39	R$ 187.013,76
RJ-4000	**R$ 294.580,48**	R$ 395.411,60	R$ 342.507,65

Tabela 4.8: Custo total mensal das estratégias dinâmicas para entrega dos pacotes.

Comparando-se os custos apresentados na Tabela 4.8 nota-se que os ganhos (em porcentagem) do algoritmo DSNR em relação aos algoritmos QRPS e KG, para cada instância, são respectivamente:

PA - *QRP-Sweep* $= 17,81\%$

DF - *QRP-Sweep* $= 14,98\%$

RJ - *K*means-Greedy* $= 15,85\%$

Ou seja, a economia gerada pela estratégia DSNR representam mais de 14,98% para todas as cidades sem precisar de dados históricos, ou realizar algum tipo de *Kmeans*. Isso indica que a estratégia pode ser explorada ao mesclar com outras informações do passado, com uma alta probabilidade de aumento de economia.O teste de Shapiro-Wilk foi aplicado sobre os custos entre os algoritmos, apresentado nas Tabelas A.13, A.17 e A.21, nos estados do PA e do DF o menor $p-value$ encontrado foi de 32,78%, superior a α ($\alpha > 5\%$), indicando não haver efeitos significativos para rejeitar a hipótese de normalidade. Após a validação da normalidade dos dados, o teste ANOVA foi aplicado sobre cada conjunto de instâncias visando avaliar se há diferença significativa entre os algoritmos. O maior $p-value$ obtido pelo teste ANOVA foi $p-value < 2.2e^{-16}$, demonstrando estatisticamente que há efeito nos dados para rejeitar a hipótese nula de igualdade entre os algoritmos. Validando os ganhos encontrados.

Posteriormente a esses resultados, novos experimentos foram explorados. Os algoritmos dinâmicos QRPS e KG baseiam-se em um número inicial de clusters previstos pelo modelo. Os resultados que obtiveram maiores destaques perante a função de custo para o PA utilizando o QRPS foi com número de clusters igual a 10 e para o KG igual a 8. Para o DF, no algoritmo QRPS o número de clusters definido foi 40 e para o KG 10. No RJ, os melhores resultados encontrados foram baseados no número mínimo de veículos utilizados nos pré-treinamentos dos algoritmos. Os resultados obtidos sobre dis-

	DSNR	QRPSWEEP	KMEANSGREEDY
PA	R$ 43.954,89	R$ 48.604,20	**R$ 43.896,49**
DF	**R$ 105.181,17**	R$ 120.947,39	R$ 107.731,30
RJ	**R$ 295.354,53**	R$ 395.411,60	R$ 342.507,65

Tabela 4.9: Resultados comparativos de custo mensal.

tância, número de veículos estão disponíveis em https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas. Perante aos resultados foi extraído uma nova comparação de custo mensal apresentado na Tabela 4.9.

4.3 Resultados com Roteamento Estático

Os três algoritmos para roteamento estático disponibilizados no *benchmark* Loggibud são: KA, KP e LKH-3. As instâncias de dados de entrada para os algoritmos são as mesmas utilizadas na Seção 4.2. Os algoritmos disponibilizados foram comparados com K-RFO e DSNR, desenvolvidos neste trabalho. O K-RFO é construído a partir do fluxograma representado na Figura 3.7. O DSNR mesmo sendo um algoritmo dinâmico, é possível a aplicação no roteamento estático, considerando que os veículos serão despachados após a inserção de todos os pacotes.

O algoritmo DSNR possui dois parâmetros (T, L) que podem influenciar no desempenho e resultado do algoritmo. T é o número de pacotes do lote que estão próximos e que serão utilizados para pesquisa das rotas vizinhas. L é o número de lotes que compõem a instância. A diferença entre a abordagem estática na divisão dos lotes é que não existe preocupação em enviar os veículos o mais rápido possível. Portanto, a cada lote recebido todas as rotas podem ser re-otimizadas pelo algoritmo. Nos resultados apresentados a seguir, os parâmetros T e L foram utilizados em diferentes intervalos para cada cidade e 3 modos distintos de ordenação dos pacotes, sendo:

- **PA:** T no intervalo de $[9, 15]$ e L no intervalo de $[1, 5]$, gerando 3150 soluções com 300 pacotes cada instância.
- **DF:** T no intervalo de $[27, 29]$ e $L = 1$, gerando 270 soluções com 1000 pacotes cada instância.
- **RJ:** $T = 30$ e $L = 1$, gerando 90 soluções com 4000 pacotes cada instância.

A escolha do intervalo foram baseadas em um tempo computacional próximo de 600 segundos por instância, por tal motivo, para as instâncias do PA se teve uma maior liberdade em variar os parâmetros. Baseado nos resultados do PA, foi feita uma estimativa de tempo para execução das instâncias do

DF reduzindo o número de possibilidade de T de 7 para 3 e o L reduzido de 5 para 1 possibilidade. O L foi estabelecido em 1, pois, 50% dos resultados do PA mostraram uma distância menor para soluções com 1 lote. No RJ a liberdade de variação dos parâmetros foi reduzida apenas para os três modos de ordenação dos pacotes.

Os resultados com roteamento estático obtidos pelos DSNR, K-RFO, LKH-3, KP e KA[10] são apresentados nas Tabelas 4.10, 4.11 e 4.12.

	DSNR		**K-RFO**		**LKH-3**		***Kmeans-Partition***		***Kmeans-Aggregation***	
Instância	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**
pa-90	**502,1782**	13	651,2639	15	538,0634	**10**	557,5792	10	550,7397	10
pa-91	609,0556	20	614,2588	12	**595,3111**	**10**	629,1676	10	612,7826	11
pa-92	**503,8677**	16	586,007	13	536,6372	**9**	591,5251	9	565,6479	9
pa-93	**467,0969**	13	557,2993	12	525,1455	**8**	519,7836	8	574,0682	8
pa-94	609,6608	14	712,98	14	**571,4023**	**9**	633,096	9	575,3919	10
pa-95	**603,2821**	16	728,5079	16	627,202	**10**	639,6516	10	628,1731	10
pa-96	699,6264	19	698,5624	16	**618,9687**	**10**	670,9439	10	639,8772	10
pa-97	**484,8118**	17	587,0453	17	528,8672	11	583,171	**10**	559,3691	12
pa-98	**487,3562**	19	657,3075	14	525,7624	**10**	563,8809	10	561,7088	11
pa-99	**596,7313**	20	718,3267	17	610,9431	11	675,7186	**10**	615,9799	11
pa-100	624,3623	17	720,0813	16	593,9225	**9**	615,9026	9	**572,7673**	10
pa-101	617,5812	13	702,9084	14	**536,9863**	**10**	546,7908	10	552,9744	10
pa-102	**547,1995**	18	616,6023	13	554,5112	**10**	562,4149	10	566,8448	10
pa-103	**551,5153**	15	568,188	17	534,4179	**11**	573,3692	11	563,3095	11
pa-104	583,6283	20	694,6307	15	670,8741	11	695,4092	11	**579,254**	**11**
pa-105	**625,4552**	21	729,1639	15	693,2719	11	695,1197	11	673,3286	**11**
pa-106	554,5131	18	719,6568	16	**550,5526**	**10**	609,9295	11	564,9806	11
pa-107	**538,8794**	15	646,1848	15	571,9787	**9**	583,1595	9	552,8074	10
pa-108	617,1169	17	776,0148	13	**607,7012**	**10**	639,73	10	690,1041	10
pa-109	505,6675	17	575,2224	14	**477,4237**	**9**	534,9432	9	514,8884	9
pa-110	**538,9774**	15	674,8988	15	596,1529	10	610,437	**9**	575,1223	10
pa-111	566,0644	17	663,7678	13	**507,3777**	**9**	567,5988	9	591,2408	9
pa-112	639,0365	17	684,8454	18	**579,7817**	**11**	654,7679	11	622,4172	12
pa-113	539,4501	20	559,2815	16	**528,6599**	**10**	543,4759	10	538,29	11
pa-114	640,1801	16	706,389	16	**611,4943**	**10**	634,2356	10	619,9934	11
pa-115	542,7957	17	592,0613	16	**507,8243**	**11**	565,4232	11	565,4284	11
pa-116	**480,5107**	12	585,6588	10	557,576	8	572,5437	**7**	598,5518	7
pa-117	647,7763	19	702,8613	17	579,0026	11	596,2579	11	**574,4921**	**11**
pa-118	**583,4521**	19	712,4457	16	589,1203	**10**	610,9078	10	633,856	11
pa-119	606,5609	18	706,5638	13	**518,9872**	**10**	600,787	10	610,0277	10

Tabela 4.10: Resultados obtidos pelos algoritmos utilizando instâncias PA.

A partir dos resultados apresentados nas Tabelas 4.10, 4.11 e 4.12 observa-se que os três algoritmos já disponibilizados no *benchmark* Loggibud apresentam melhores resultados que as soluções propostas neste trabalho, mesmo assim, o teste de Shapiro-Wilk foi aplicado sobre os custos dos três conjuntos de instâncias (PA, DF, RJ) indicando não haver efeitos significativos para rejeitar a hipótese de normalidade desses custos ($p-value > \alpha$), considerando um nível de significância de 5% ($\alpha = 5\%$). Após a validação dos dados, o teste ANOVA foi aplicado sobre cada conjunto de instâncias visando avaliar se há diferença significativa entre os algoritmos comparados (DSNR, K-RFO, LKH-3, KP e KA). Conforme as Tabelas A.2, A.6 e A.10, observou-se que todos os $p-value$ são menores que 5%, indicando haver efeito nos dados para rejeitar a hipótese nula de igualdade entre os algoritmos comparados. Portanto, foi aplicado o teste de Shapiro-Wilk sobre os resíduos encontrados no teste da ANOVA, e observou-se que existe normalidade dos dados nos estados do PA

[10]O algoritmo KRFO foi executado conforme o algoritmo disponibilizado no link https://github.com/Wiltoon/CVRP-heteregeneos, e os do repositório Loggibud, estão na task1 do link https://github.com/loggi/loggibud.

	DSNR		K-RFO		LKH-3		K*means-Partition*		K*means-Aggregation*	
Instância	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**
df-90	1355,5303	56	1703,9405	50	1406,9901	31	1302,3282	**31**	**1221,0963**	32
df-91	1170,6157	41	1483,9822	39	1223,3609	26	1179,3694	**25**	**1157,7031**	26
df-92	1294,7129	38	1554,089	42	1289,9647	27	**1232,2215**	**27**	1310,9162	28
df-93	1193,3068	41	1302,0775	37	1108,805	**24**	1110,6754	25	**1046,7733**	25
df-94	1329,8288	41	1742,9646	41	1359,7764	27	**1310,117**	**27**	1319,2238	28
df-95	1458,1513	49	1769,1566	47	1500,8011	32	**1381,1981**	**32**	1418,719	33
df-96	1443,1211	45	1767,1022	46	1384,3199	31	**1307,9033**	**30**	1342,0728	33
df-97	1215,466	38	1540,3804	37	1241,9613	25	1176,5852	**25**	**1148,0613**	27
df-98	1182,7343	34	1476,8866	38	1265,3963	25	1222,3422	**25**	**1152,1412**	26
df-99	1162,6857	37	1415,5286	37	1260,749	27	1178,0501	26	**1131,4321**	**26**
df-100	1483,3855	47	1915,4525	49	1557,0143	35	**1415,3397**	**35**	1459,844	36
df-101	1330,9703	48	1635,1893	50	1275,2915	32	1282,4271	**31**	**1211,5032**	33
df-104	1291,0935	43	1583,9213	45	1342,3822	31	1277,5496	**30**	**1211,6462**	33
df-105	1198,9164	39	1523,4451	38	1215,5885	25	1188,915	**25**	**1164,0346**	26
df-106	1499,7017	48	1816,046	49	1426,2343	31	1397,797	**31**	**1323,0836**	33
df-108	1404,8589	49	1719,4851	45	1450,1971	32	1406,9237	**32**	**1308,7067**	33
df-109	1380,2784	48	1691,1042	49	1438,4284	33	1385,8273	**32**	**1294,2559**	33
df-111	1404,0012	44	1706,1148	46	1395,2662	**31**	1326,3516	32	**1284,8901**	32
df-112	1399,1351	47	1712,3732	45	1413,0072	**30**	1413,3309	31	**1349,5378**	31
df-113	1272,7549	43	1517,7505	39	1261,3397	28	1213,7866	28	**1170,0331**	**28**
df-114	1437,9012	43	1821,9135	49	1544,9949	**32**	1401,9741	33	**1372,2636**	34
df-115	1361,9951	50	1600,9243	48	1389,4464	31	1357,6678	**30**	**1317,0106**	32
df-116	1318,6134	43	1568,651	45	1299,5217	29	1308,842	30	**1259,7098**	**30**
df-117	1195,983	35	1453,9709	39	1162,4692	**26**	1197,3655	26	**1136,5195**	27
df-118	1387,8154	40	1596,0167	44	1303,9595	28	**1281,5457**	**28**	1315,2874	30
df-119	1282,3624	41	1624,0705	38	1272,7637	27	**1228,328**	**27**	1266,9108	27

Tabela 4.11: Resultados obtidos pelos algoritmos com instâncias DF.

	DSNR		K-RFO		LKH-3		K*means-Partition*		K*means-Aggregation*	
Instância	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**	**Distância P.**	**N veículos**
rj-90	3621,9236	188	4491,6945	200	5388,385	136	**3073,3578**	**135**	3118,1265	141
rj-91	3791,7619	207	4628,3242	217	6511,6793	**142**	**3251,4736**	143	3318,6726	151
rj-92	3764,2875	203	4610,9433	203	4094,569	151	**3204,6516**	**141**	3246,275	148
rj-93	3713,6526	193	4559,3406	204	6653,945	**134**	**3120,6614**	137	3246,8159	140
rj-94	3618,0051	190	4356,6485	184	4926,4711	130	**3004,0154**	**127**	3056,4748	132
rj-95	3123,1155	148	3793,7516	161	5251,4124	110	2710,6255	**110**	**2677,1376**	115
rj-96	3485,5196	185	4334,2292	203	6072,2777	**134**	**2976,8039**	135	3039,9444	139
rj-97	3191,1133	169	3886,78	171	4607,8849	117	**2720,5106**	**114**	2787,3641	117
rj-98	3793,7492	202	4537,9622	201	4954,5428	140	**3188,7021**	**139**	3222,2083	146
rj-99	3922,4207	205	4708,9252	208	6920,9696	**144**	**3272,7893**	147	3315,049	153
rj-100	3605,8336	183	4367,4413	194	3874,1759	138	**3038,0603**	**132**	3120,8002	136
rj-102	3280,0545	171	3907,9502	169	6596,2628	135	**2801,9006**	**119**	2903,0729	122
rj-103	3938,8263	220	4732,8593	222	7523,1736	144	**3312,7488**	**147**	3327,6821	154
rj-104	3066,2803	150	3726,3576	163	4423,6751	111	**2659,3173**	**109**	2701,4657	113
rj-105	3899,6712	208	4747,2082	213	7410,4178	143	**3245,8427**	**146**	3291,3155	151
rj-106	3657,7144	206	4432,1513	199	3612,4803	138	3085,9083	**133**	**3072,9396**	137
rj-107	3253,8857	171	3989,0186	170	5968,0494	129	**2797,0047**	**118**	2846,3987	123
rj-108	3379,4022	173	4033,659	185	4859,8954	123	**2847,2968**	**121**	2918,6149	126
rj-109	3407,9128	182	4161,502	184	4912,9877	**127**	**2949,0842**	128	3030,9896	129
rj-110	3163,0801	165	3811,0017	168	3517,4222	125	**2740,0148**	**116**	2754,5577	119
rj-111	3804,9045	197	4577,1443	210	6251,1822	**138**	**3247,9415**	141	3270,6813	144
rj-112	3527,5469	195	4293,9284	195	6091,9681	**131**	**3070,1194**	132	3095,227	138
rj-113	3872,7401	209	4738,9737	220	7435,4522	**145**	**3289,4262**	148	3386,3312	154
rj-114	3887,2399	202	4689,5725	212	6917,9895	**139**	3277,4658	143	**3275,7415**	148
rj-115	3781,2642	199	4666,6679	210	7275,6584	143	**3281,6868**	**147**	3281,3444	151
rj-116	3371,1453	171	4062,4955	179	3474,8104	128	**2969,0476**	**122**	3000,9243	130
rj-117	3697,9741	208	4601,8075	215	6021,094	144	**3254,6837**	**143**	3256,3734	150
rj-118	3932,349	215	4672,427	226	4113,4376	157	**3330,8096**	**149**	3419,3642	158
rj-119	3461,4498	192	4313,6687	197	5553,6329	**130**	**2994,9552**	131	3080,8431	139

Tabela 4.12: Resultados obtidos pelos algoritmos com instâncias RJ.

e DF. Com isso, aplicou-se o teste de Tukey, visando encontrar se existem correlações entre os algoritmos. Observou-se, na Tabela A.4, que ao utilizar o algoritmo DSNR os custos se aproximaram dos custos obtidos pelos algoritmos KA e KP e os algoritmos KA e KP foram fortemente correlacionados, o algoritmo K-RFO foi o único que ficou distante da normalidade de modo negativo em relação aos custos dos outros algoritmos.

No estado do DF, aplicou-se também o teste de Tukey, e observou-se, na Tabela A.8, que ao comparar os custos dos algoritmos, os dados obtiveram uma igualdade de 11,09% entre LKH-3-DSNR, 97,34% entre KP-KA 28,37% entre LKH-3-KA e 65,01% entre LKH-3, isso demonstra uma aproximação nos resultados entre os algoritmos disponibilizados pelo repositório e o DSNR se distanciando negativamente em relação aos custos, devido ao número de veículos utilizados. No estado do RJ, aplicou-se o teste de Shapiro-Wilk nos residuais, obtendo um $p - value$ próximo de 0%, portanto, demonstrou-se que não existe normalidade de igualdade entre os algoritmos, quando comparados neste estado. Mesmo assim, aplicou-se o teste de Tukey, visando verificar alguma possível igualdade. Apenas os algoritmos KP-KA, obtiveram uma igualdade, sendo os dois melhores algoritmos para este estado, e o algoritmo DSNR sendo o terceiro melhor. O LKH-3 neste estado obteve o pior dos resultados neste estado, inclusive atrás do algoritmo K-RFO, que nos estados anteriores estava na última colocação. No estado do PA pode-se observar que a estratégia DSNR possuí uma distância menor para 14 de 30 instâncias, apesar disso, o número de veículos utilizado é significativamente superior aos algoritmos disponibilizados do repositório. Com isso, os custos da utilização dos veículos superou os custos de transportes dos outros algoritmos. Os resultados do K-RFO se mostraram mais custosos, sendo mais econômico apenas para a cidade do RJ contra o LKH-3. A justificativa para esse algoritmo não ter se saído bem é que possuí muitas fases com otimizações como o Gerenciador de Veículos e o Gerenciador de Pacotes, com tempo limitado. Além disso, o Gerenciador de Pacotes, tinha como objetivo de prever o preço final de cada pacote em cada veículo, entretanto, a previsão não é a realizada, acarretando um aumento significativo no número de veículos ociosos. Como consequência direta, as distâncias percorridas também são menores. A Tabela 4.13 apresenta os custos de entrega desses cinco algoritmos. Os dados apresentados nesta tabela ratificam a percepção de que o menor número de veículos utilizados gera menores custos totais de entrega.

	DSNR	**K-RFO**	**LKH-3**	***Kmeans-Partition***	***Kmeans-Aggregation***
PA-300	R$ 39.920,83	R$ 44.281,65	**R$ 36.870,37**	R$ 38.809,72	R$ 38.158,48
DF-1000	R$ 81.834,57	R$ 96.840,47	R$ 77.305,26	R$ 74.771,34	**R$ 73.641,08**
RJ-4000	R$ 276.255,76	R$ 321.747,81	R$ 363.827,93	**R$ 223.893,16**	R$ 228.576,05

Tabela 4.13: Custos totais gerados pelos algoritmos estáticos.

CAPÍTULO

5

Conclusões e Trabalhos Futuros

Este trabalho de mestrado objetivou o estudo, projeto e desenvolvimento de uma estratégia algorítmica para solução do problema de roteamento dinâmico de veículos. A solução proposta, denominada de *Dynamic Search per Neighbors Routes* (DSNR) consegue reutilizar rotas existentes de forma que, a cada lote de pacotes recebidos para roteirização, a solução possa progredir na resolução sem reconstruir rotas e pacotes já roteirizados. Também foi desenvolvida uma estratégia para resolver o lote de pacotes com a abordagem estática, denominada de estratégia *Kmeans, Relax-and-Fix and Optimizations* (K-RFO). O algoritmo K-RFO utiliza uma abordagem de particionamento junto com uma heurística *Relax-and-Fix* para resolução da roteirização dos pacotes.

A estratégia dinâmica desenvolvida alcançou melhorias da ordem de 15% a 17% em relação ao algoritmo dinâmico disponibilizado por Loggibud que apresentou os melhores resultados. O algoritmo DSNR também superou o algoritmo K-RFO quando aplicado no PRV estático. Entretanto, em relação ao número de veículos utilizados, a abordagem dinâmica se mostrou desvantajosa, afetada diretamente pelos custos operacionais dos veículos.

Uma limitação do algoritmo estático K-RFO reside na utilização de várias fases, afetando significativamente o tempo de execução desse algoritmo. Uma das limitações do algoritmo dinâmico DSNR é que o dinamismo abordado ainda necessita de um "buffer" de pacotes que, por menor que seja, pode prejudicar no tempo de entrega, em situações onde esse tempo pode ser muito restrito.

Como trabalhos futuros sugere-se aplicar a estratégia *Kmeans* em dados históricos, assim como os algoritmos *QRP-Sweep* e K*means-Greedy* e inserir os pacotes com a estratégia DSNR. Outro ponto incitador seria avaliar a inserção

de um primeiro lote com um algoritmo estático mais eficiente, com os próximos lotes sendo inseridos com a estratégia DSNR. Ambas concepções aparentam trazer ganhos significativos.

Referências Bibliográficas

Alvarenga, G. B. (2005). Um algoritmo híbrido para os problemas de roteamento de veículos estático e dinâmico com janela de tempo.

Azi, N., Gendreau, M., e Potvin, J.-Y. (2007). An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European journal of operational research*, 178(3):755–766.

Bertsimas, D. J. e Van Ryzin, G. (1993). Stochastic and dynamic vehicle routing with general demand and interarrival time distributions. *Advances in Applied Probability*, 25(4):947–978.

Branchini, R. M., Armentano, V. A., e Løkketangen, A. (2009). Adaptive granular local search heuristic for a dynamic vehicle routing problem. *Computers & Operations Research*, 36(11):2955–2968.

Bräysy, O. e Gendreau, M. (2005a). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, 39(1):104–118.

Bräysy, O. e Gendreau, M. (2005b). Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, 39(1):119–139.

Chen, P., Huang, H.-k., e Dong, X.-Y. (2010). Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. *Expert Systems with Applications*, 37(2):1620–1627.

Chen, S., Chen, R., e Gao, J. (2017). A monarch butterfly optimization for the dynamic vehicle routing problem. *Algorithms*, 10(3):107.

Clarke, G. e Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581.

Comert, S. E., Yazgan, H. R., Kır, S., e Yener, F. (2018). A cluster first-route second approach for a capacitated vehicle routing problem: a case study. *International Journal of Procurement Management*, 11(4):399–419.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations research*, 6(6):791–812.

Dantzig, G. B. e Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91.

de Menezes, G. G. S. (2021). A relax-and-fix based approach for solving heterogenous fleet vehicle routing problem with three-dimensional loading constraints.

Dillenberger, C., Escudero, L. F., Wollensak, A., e Zhang, W. (1994). On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research*, 75(2):275–286.

Dorigo, M. e Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 2, páginas 1470–1477. IEEE.

Fahrion, R. e Wrede, M. (1990). On a principle of chain-exchange for vehicle-routeing problems (1-vrp). *Journal of the Operational Research Society*, 41(9):821–827.

Fisher, M. L. e Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124.

Fonseca-Galindo, J. C., de Castro Surita, G., Neto, J. M., de Castro, C. L., e Lemos, A. P. (2022). A multi-agent system for solving the dynamic capacitated vehicle routing problem with stochastic customers using trajectory data mining. *Expert Systems with Applications*, 195:116602.

Gillett, B. E. e Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2):340–349.

Hannan, M., Akhtar, M., Begum, R., Basri, H., Hussain, A., e Scavino, E. (2018). Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using pso algorithm. *Waste management*, 71:31–41.

Hansen, P., Mladenović, N., e Pérez, J. A. M. (2010). Variable neighbourhood search: Methods and a. *Annals of Operations Research*, 175(1):367–407.

He, R., Xu, W., Sun, J., e Zu, B. (2009). Balanced K-Means Algorithm for Partitioning Areas in Large-Scale Vehicle Routing Problem. *3rd International Symposium on Intelligent Information Technology Application*, 3:87–90.

Helsgaun, K. (1998). An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. *European Journal of Operational Research*, 126(81):106–130.

Helsgaun, K. (2017). An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*.

Larsen, A., Madsen, O., e Solomon, M. (2002). Partially Dynamic Vehicle Routing - Models and Algorithms. *Journal of the Operational Research Society*, 53(6):637–646.

Le, T. D. C., Nguyen, D. D., Oláh, J., e Pakurár, M. (2022). Clustering algorithm for a vehicle routing problem with time windows. *Transport*, 37(1):17–27.

Lenstra, J. K. e Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

Loggibud (2021). loggibud: Loggi benchmark for urban deliveries. GitHub.

Marinakis, Y. e Marinaki, M. (2010). A hybrid genetic–particle swarm optimization algorithm for the vehicle routing problem. *Expert Systems with Applications*, 37(2):1446–1455.

Marinakis, Y., Migdalas, A., e Pardalos, P. M. (2009). Multiple phase neighborhood search—grasp based on lagrangean relaxation, random backtracking lin–kernighan and path relinking for the tsp. *Journal of combinatorial optimization*, 17(2):134–156.

Montemanni, R., Gambardella, L. M., Rizzoli, A. E., e Donati, A. V. (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of combinatorial optimization*, 10(4):327–343.

Nazari, M., Oroojlooy, A., Snyder, L., e Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31.

Necula, R., Breaban, M., e Raschip, M. (2017). Tackling dynamic vehicle routing problem with time windows by means of ant colony system. In *Proceedings of the 2017 IEEE Congress on Evolutionary Computation*, páginas 2480–2487. IEEE.

Potvin, J.-Y. e Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46(12):1433–1446.

Psaraftis, H. N. (1988). Dynamic vehicle routing problems. *Vehicle routing: Methods and studies*, 16:223–248.

Psaraftis, H. N. (1995). Dynamic vehicle routing: Status and prospects. *Annals of operations research*, 61(1):143–164.

Ralphs, T. K., Kopman, L., Pulleyblank, W. R., e Trotter, L. E. (2003). On the capacitated vehicle routing problem. *Mathematical programming*, 94(2):343–359.

Reed, M., Yiannakou, A., e Evering, R. (2014). An ant colony algorithm for the multi-compartment vehicle routing problem. *Applied Soft Computing*, 15:169–176.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Santos, P. T. G. d. (2019). Uma heurística relax-and-fix para o vehicle routing problem with pickup and delivery with time windows aplicado a um problema de transporte marítimo.

Singanamala, P., Reddy, D., e Venkataramaiah, P. (2018). Solution to a multi depot vehicle routing problem using k-means algorithm, clarke and wright algorithm and ant colony optimization. *International Journal of Applied Engineering Research*, 13(21):15236–15246.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., e Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2):170–186.

Thorndike, R. L. (1953). Who belongs in the family. In *Psychometrika*. Citeseer.

Toth, P. e Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*, 15(4):333–346.

Xu, H., Pu, P., e Duan, F. (2018). Dynamic vehicle routing problems with enhanced ant colony optimization. *Discrete Dynamics in Nature and Society*, 2018.

Xu, Y., Wang, L., e Yang, Y. (2013). Dynamic vehicle routing using an improved variable neighborhood search algorithm. *Journal of Applied Mathematics*, 2013.

Zirour, M. (2008). Vehicle routing problem: Models and solutions. *Journal of Quality Measurement and Analysis JQMA*, 4(1):205–218.

ANEXO

A

Análises Estatísticas

Shapiro					
	DSNR	**K-RFO**	**LKH-3**	**KP**	**KA**
W	98,12%	91,29%	96,15%	97,63%	92,37%
P-VALUE	85,67%	**1,76%**	33,86%	72,23%	33,55%

Tabela A.1: Aplicação do teste de Shapiro nos algoritmos estáticos da cidade de Belém (PA).

ANOVA					
Response = custo					
	DF	**Sum sq**	**Mean Sq**	**F value**	**Pr(>F)**
Algoritmo	4	1072468	268117	22,925	**1,08e-14**
Residuals	145	1695798	11695		

Tabela A.2: Aplicação da ANOVA nos algoritmos estáticos na cidade de Belém (PA).

Shapiro	
Residuals ANOVA	
p-value	**w**
66,33%	99,29%

Tabela A.3: Aplicação do teste de Shapiro nos Residuais da ANOVA dos algoritmos estáticos, na cidade de Belém (PA).

Tukey				
95 % family-wise confidence level				
	diff	lwr	upr	p adj
KA-DSNR	-58,74667	-135,88051	18,38717	**22,40%**
KP-DSNR	-37,038	-114,17184	40,09584	**67,52%**
KRFO-DSNR	145,36033	68,22649	222,49417	0,00%
LKH3-DSNR	-101,68267	-178,81651	-24,54883	0,34%
KP-KA	21,70867	-55,42517	98,84251	**93,67%**
KRFO-KA	204,107	126,97316	281,24084	0,00%
LKH3-KA	-42,936	-120,06984	34,19784	**53,98%**
KRFO-KP	182,39833	105,26449	259,53217	0,00%
LKH3-KP	-64,64467	-141,77851	12,48917	**14,60%**
LKH3-KRFO	-247,043	-324,17684	-169,90916	0,00%

Tabela A.4: Aplicação do teste de Tukey, nos algoritmos estáticos, na cidade de Belém (PA).

Shapiro					
	DSNR	**K-RFO**	**LKH-3**	**KP**	**KA**
W	93,23%	98,19%	98,46%	94,50%	97,01%
P-VALUE	8,79%	91,15%	95,30%	17,63%	62,46%

Tabela A.5: Aplicação do teste de Shapiro nos algoritmos estáticos da cidade do Distrito Federal (DF).

ANOVA					
Response = custo					
	DF	**Sum sq**	**Mean Sq**	**F value**	**Pr(>F)**
Algoritmo	4	13774760	3443690	52.08	**2.2e-16**
Residuals	125	8265375	66123		

Tabela A.6: Aplicação da ANOVA nos algoritmos estáticos na cidade do Distrito Federal (DF).

Shapiro	
Residuals anova	
p-value	**w**
27,98%	98,74%

Tabela A.7: Aplicação do teste de Shapiro nos Residuais da ANOVA dos algoritmos estáticos, na cidade do Distrito Federal (DF).

Tukey				
95 % family-wise confidence level				
	diff	lwr	upr	p adj
KA-DSNR	-315,13423	-512,54495	-117,72351	0,02%
KP-DSNR	-271,66192	-469,07264	-74,25121	0,20%
KRFO-DSNR	577,14962	379,7389	774,56033	0,00%
LKH3-DSNR	-174,20462	-371,61533	23,2061	**11,09%**
KP-KA	43,47231	-153,93841	240,88303	**97,34%**
KRFO-KA	892,28385	694,87313	1089,69456	0,00%
LKH3-KA	140,92962	-56,4811	338,34033	**28,37%**
KRFO-KP	848,81154	651,40082	1046,22226	0,00%
LKH3-KP	97,45731	-99,95341	294,86803	**65,01%**
LKH3-KRFO	-751,35423	-948,76495	-553,94351	0,00%

Tabela A.8: Aplicação do teste de Tukey, nos algoritmos estáticos, na cidade do Distrito Federal (DF).

Shapiro					
	DSNR	**KRFO**	**LKH3**	**KPARTITION**	**KAGGREGATION**
W	93,93%	90,57%	95,01%	91,96%	93,75%
P-VALUE	9,59%	**1,34%**	18,44%	**2,98%**	8,60%

Tabela A.9: Aplicação do teste de Shapiro nos algoritmos estáticos da cidade do Rio de Janeiro (RJ).

ANOVA					
Response = custo					
	DF	**Sum sq**	**Mean Sq**	**F value**	**Pr(>F)**
Algoritmo	4	501224682	125306171	75.595	**2.2e-16**
Residuals	140	232062489	1657589		

Tabela A.10: Aplicação da ANOVA nos algoritmos estáticos na cidade do Rio de Janeiro (RJ).

Shapiro	
Residuals anova	
p-value	**w**
0,00%	93,54%

Tabela A.11: Aplicação do teste de Shapiro nos Residuais da ANOVA dos algoritmos estáticos, na cidade do Rio de Janeiro (RJ).

Tukey				
95 % family-wise confidence level				
	diff	**lwr**	**upr**	**p adj**
KA-DSNR	-1644,128	-2578,5388	-709,7164	0,00%
KP-DSNR	-1805,607	-2740,0177	-871,1954	0,00%
KRFO-DSNR	1568,692	634,2806	2503,1029	0,01%
LKH3-DSNR	3019,73	2085,3185	3954,1408	0,00%
KP-KA	-161,479	-1095,8901	772,9322	**98,93%**
KRFO-KA	3212,819	2278,4081	4147,2305	0,00%
LKH3-KA	4663,857	3729,4461	5598,2684	0,00%
KRFO-KP	3374,298	2439,8871	4308,7094	0,00%
LKH3-KP	4825,336	3890,925	5759,7474	0,00%
LKH3-KRFO	1451,038	516,6268	2385,4491	0,03%

Tabela A.12: Aplicação do teste de Tukey, nos algoritmos estáticos, na cidade do Rio de Janeiro (RJ).

Shapiro			
	DSNR	**QRPS**	**KMEANS**
W	98,89%	97,70%	97,68%
P-VALUE	79,51%	74,25%	73,51%

Tabela A.13: Aplicação do teste de Shapiro nos algoritmos dinâmicos na cidade de Belém (PA).

ANOVA					
Response = custo					
	DF	**Sum sq**	**Mean Sq**	**F value**	**Pr(>F)**
Algoritmo	2	88493968	44246984	998.69	**2.2e-16**
Residuals	87	3854532	44305		

Tabela A.14: Aplicação da ANOVA nos algoritmos dinâmicos na cidade de Belém (PA).

Shapiro	
Residuals anova	
p-value	**w**
88,16%	99,23%

Tabela A.15: Aplicação do teste de Shapiro nos Residuais da ANOVA dos algoritmos dinâmicos, na cidade de Belém (PA).

Tukey				
95 % family-wise confidence level				
	diff	**lwr**	**upr**	**p adj**
KMEANS-DSNR	2225,975	2096,3842	2355,5658	0,00
QRPS-DSNR	271,278	141,6872	400,8688	0,00
QRPS-KMEANS	-1954,697	-2084,2878	-1825,1062	0,00

Tabela A.16: Aplicação do teste de Tukey, nos algoritmos dinâmicos, na cidade de Belém (PA).

Shapiro			
	DSNR	**QRPS**	**KMEANS**
W	97,71%	96,10%	98,69%
P-VALUE	74,48%	32,78%	96,44%

Tabela A.17: Aplicação do teste de Shapiro nos algoritmos dinâmicos na cidade de Distrito Federal (DF).

ANOVA					
Response = custo					
	DF	**Sum sq**	**Mean Sq**	**F value**	**Pr(>F)**
Algoritmo	2	125665510	62832755	384.87	**2.2e-16**
Residuals	87	14203403	163258		

Tabela A.18: Aplicação da ANOVA nos algoritmos dinâmicos na cidade do Distrito Federal (DF).

Shapiro	
Residuals anova	
p-value	w
79,74%	99,09%

Tabela A.19: Aplicação do teste de Shapiro nos Residuais da ANOVA dos algoritmos dinâmicos, na cidade do Distrito Federal (DF).

Tukey				
95 % family-wise confidence level				
	diff	lwr	upr	p adj
KMEANS-DSNR	2727,751	2478,989	2976,5136	0
QRPS-DSNR	525,541	276,7787	774,3033	0
QRPS-KMEANS	-2202,21	-2450,9726	-1953,448	0

Tabela A.20: Aplicação do teste de Tukey, nos algoritmos dinâmicos, na cidade do Distrito Federal (DF).

Shapiro			
	DSNR	QRPS	KMEANS
W	95,72%	88,05%	97,68%
P-VALUE	26,22%	**0,29%**	12,84%

Tabela A.21: Aplicação do teste de Shapiro nos algoritmos dinâmicos na cidade do Rio de Janeiro (RJ).

ANOVA					
Response = custo					
	DF	Sum sq	Mean Sq	F value	Pr(>F)
Algoritmo	2	167040680	83520340	114.65	**2.2e-16**
Residuals	87	63377019	728471		

Tabela A.22: Aplicação da ANOVA nos algoritmos dinâmicos na cidade do Rio de Janeiro (RJ).

Shapiro	
Residuals anova	
p-value	w
0,23%	95,22%

Tabela A.23: Aplicação do teste de Shapiro nos Residuais da ANOVA dos algoritmos dinâmicos, na cidade do Rio de Janeiro (RJ).

Tukey				
95 % family-wise confidence level				
	diff	lwr	upr	p adj
KMEANS-DSNR	1571,771	1046,293	2097,248	0
QRPS-DSNR	3335,236	2809,758	3860,713	0
QRPS-KMEANS	1763,465	1237,987	2288,943	0

Tabela A.24: Aplicação do teste de Tukey, nos algoritmos dinâmicos, na cidade do Rio de Janeiro (RJ).