UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
MESTRADO EM COMPUTAÇÃO APLICADA

TIAGO FUNK

PROPOSTA DE ALGORITMO DE BUSCA LARGA EM VIZINHANÇA ADAPTATIVA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM SUPORTE A JANELAS DE TEMPO E COLETA DE RECOMPENSA

JOINVILLE

2023

TIAGO FUNK

PROPOSTA DE ALGORITMO DE BUSCA LARGA EM VIZINHANÇA ADAPTATIVA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM SUPORTE A JANELAS DE TEMPO E COLETA DE RECOMPENSA

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Dr. Adriano Fiorese
Coorientador: Dr. Fabiano Baldo

JOINVILLE

2023

Funk, Tiago
PROPOSTA DE ALGORITMO DE BUSCA LARGA EM VIZINHANÇA ADAPTATIVA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM SUPORTE A JANELAS DE TEMPO E COLETA de RECOMPENSA / Tiago Funk. -- 2023.
76 p.

Orientador: Adriano Fiorese
Coorientador: Fabiano Baldo
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2023.

1. Otimização. 2. Problema de roteamento de veículos. 3. Metaheurísticas. 4. Busca larga adaptativa em vizinhança. 5. Simulated Annealing. I. Fiorese, Adriano. II. Baldo, Fabiano. III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada. IV. Título.

Tiago Funk

PROPOSTA DE ALGORITMO DE BUSCA LARGA EM VIZINHANÇA ADAPTATIVA PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM SUPORTE A JANELAS DE TEMPO E COLETA DE RECOMPENSA

Esta dissertação foi julgada adequada para a obtenção do título de **Mestre em Computação Aplicada** área de de concentração em "Sistemas de Computação", e aprovada em sua forma final pelo Curso de Mestrado em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina.

Banca Examinadora

Prof. Dr. Adriano Fiorese
DCC/CCT/UDESC
(Presidente/Orientador)

Membros:

Prof. Dr. Adriano Fiorese
DCC/CCT/UDESC
(Presidente/Orientador)

Prof. Rafael de Santiago
INE/CTC/UFSC

Prof. Dr. Rafael Stubs Parpinelli
DCC/CCT/UDESC

Joinville, 12 de julho de 2023

“A mais antiga e mais forte emoção da humanidade é o medo, e o mais antigo e mais forte tipo de medo é o medo do desconhecido”

Howard Phillips Lovecraft

”De nada tenho certeza, mas a visão das estrelas me faz sonhar”

Vincent Van Gogh

RESUMO

Os problemas de roteamento de veículo são uma área de estudo bastante abrangente, geralmente abordados como parte dos sistemas de gerenciamento de transporte. Na literatura, existem duas abordagens predominantes para a solução de problemas de roteamento de veículos. A primeira é o *Vehicle Routing Problem* (VRP), que busca alocar a frota de forma a minimizar o seu custo total. A segunda é o *Team Orienteering Problem* (TOP), cuja solução busca maximizar a recompensa coletada pela frota. A abordagem que busca unir custo e recompensa é a *Vehicle Routing Problem with Profit* (VRPP), onde ambos são considerados objetivo de otimização. Porém, essa variante do problema é pouco abordada. Este trabalho vai abordar o *Vehicle Routing Problem with Profit and Time Windows* (VRPPTW), onde são adicionadas janelas de atendimento à modelagem do problema. Metaheurísticas estão sendo utilizadas nos últimos anos dada a característica de fornecer soluções suficientemente boas em tempo aceitável para problemas de otimização. Este trabalho utiliza um algoritmo baseado em *Adaptive Large Neighborhood Search* (ALNS) com uma lógica de aceitação de possíveis soluções do *Simulated Anneling* (SA) para resolver o VRPPTW. Realizou-se um estudo com um *benchmark* com 168 instâncias do Vehicle Routing Problem with Time Windows (VRPTW) adaptadas para cada requisição possuir uma recompensa atrelada. Em um primeiro experimento para comparar com a literatura, observou-se que em 63 instâncias foi alcançado o resultado da literatura, indicando resultados promissores do algoritmo. No segundo experimento, foi realizada a adaptação do *benchmark* e o algoritmo ALNS foi executado com função objetivo apenas considerando custo e uma segunda execução considerando custo e recompensa. O experimento mostrou que houve aumento do custo e diminuição da recompensa não coletada. Ainda, foi realizada uma análise qualitativa do comportamento do algoritmo durante sua execução.

Palavras-chaves: Otimização, Problema de roteamento de veículos, Metaheurísticas, Busca larga adaptativa em vizinhança, Simulated Annealing.

ABSTRACT

Vehicle routing problems are a very broad area of study, generally considered as part of transportation management systems. In the literature, there are two predominant approaches to solving vehicle routing problems. The first is *Vehicle Routing Problem* (VRP), which seeks to allocate the fleet in such a way as to minimize its total cost. The second is *Team Orienteering Problem* (TOP), whose solution seeks to maximize the reward collected by the fleet. The approach that seeks to unite cost and reward is *Vehicle Routing Problem with Profit* (VRPP), where both are considered optimization objectives. However, this variant of the problem is little addressed. This work will address the *Vehicle Routing Problem with Profit and Time Windows* (VRPPTW), where service windows are added to the modeling of the problem. Metaheuristics have been used in recent years due to the characteristic of providing good enough solutions in acceptable time for optimization problems. This work uses an algorithm based on *Adaptive Large Neighborhood Search* (ALNS) with an acceptance logic of possible solutions from *Simulated Anneling* (SA) to solve VRPPTW. A study was carried out with a *benchmark* with 168 instances of VRPTW adapted for each request to have a reward attached. In a first experiment to compare with the literature, it was observed that in 63 instances the literature result was achieved, indicating promising results of the algorithm. In the second experiment, the *benchmark* adaptation was performed and the ALNS algorithm was executed with the objective function only considering cost and a second execution considering cost and reward. The experiment showed that there was an increase in the cost and a decrease in the reward not collected. Also, a qualitative analysis of the behavior of the algorithm during its execution was carried out.

Keywords: Otimization, Vehicle routing problem, Metaheuristics, Adaptive large neighborhood search, Simulated Annealing.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ALNS	Adaptive Large Neighborhood Search
ATS	Adaptive Tabu Search
CPTP	capacitated profitable tour problem
CRPPSLR	Capacitated Routing Problem with Profits and Service Level Requirements
CTOP	Capacitated Team Orienteering Problem
CVRP	Capacitated Vehicle Routing Problem
GRASP	Greedy Randomized Adaptive Search Procedures
LNS	Large Neighborhood Search
LS	Local Search
MBSU	Metaheurística Baseada em Solução Única
MBSUs	Metaherísticas Baseadas em Solução Única
PSO	Particle Swarm Optimization
PTP	Profitable Tour Problem
RPD	Relative Percent Difference
SA	Simulated Anneling
TOP	Team Orienteering Problem
TOPTW	Team Orienteering Problem with Time Windows
TRPP	Traveling Repairman Problem with Profits
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem
VRPP	Vehicle Routing Problem with Profit
VRPPTW	Vehicle Routing Problem with Profit and Time Windows
VRPTW	Vehicle Routing Problem with Time Windows
TMS	Transport Management System

SUMÁRIO

1 INTRODUÇÃO

Com a evolução constante e crescimento das cidades e dos sistemas de comércio eletrônico, o número de encomendas transportadas vem crescendo consideravelmente, exigindo melhores sistemas de roteamento de veículos para realização das entregas (Steve Banker, 2018). Os sistemas de gerenciamento de transporte (*Transport Management System* (TMS)) vêm recebendo muita atenção de empresas como Uber, por exemplo, e o aprofundamento dos avanços no desenvolvimento dos TMS's deu início a um novo ramo de serviços, os chamados TaaS (Transport-as-a-Service), sendo que é esperado que tais serviços ultrapassem a receita de US$ 52 bilhões em 2025 (Bridget McCrea, 2019). Soma-se a esses fatores ainda o impacto da pandemia de COVID-19. Se por meados de 2015 a 2019, compras online, serviços de entrega de comida e aplicativos de carona já exerciam um forte impacto na vida das pessoas, a chegada da pandemia mostrou um crescimento desse tipo de atividade. Por exemplo, entre fevereiro e maio de 2020, houve um aumento de 71% em compras online comparando com o mesmo período de 2019 (Rodrigo Martucci, 2021). Dados como esses demonstram, mesmo que indiretamente, que o assunto de roteamento de veículos apresenta-se cada vez mais importante para a sociedade. Para tanto é necessário que o mesmo percorra um caminho desde onde encontra-se originalmente armazenado até o domicílio do comprador.

O problema de roteamento de veículos consiste em criar rotas para uma frota que percorre localizações diferentes geograficamente. O desafio é criar rotas que sejam as mais otimizadas possível de acordo com as necessidades que atendam às restrições impostas pelo cenário trabalhado. Uma restrição é uma representação de uma limitação. Por exemplo, a carga máxima de um veículo. Restrições podem ser físicas (carga máxima, autonomia máxima dos veículos), legais (funcionários não podem trabalhar mais de 8 horas por dia) ou práticas (uma carga não pode ser coletada/entregue fora de uma janela de tempo de atendimento). Assim, para a solução desse desafio de otimização, é importante citar a forma como busca-se alcançá-la. Primeiro é necessário definir qual critério deve ser otimizado. Pode-se utilizar a distância/custo da rota ou ainda recompensa/lucro. No primeiro caso, busca-se minimizar e no segundo busca-se maximizar o valor do critério (GENDREAU; POTVIN et al., 2010).

O *Vehicle Routing Problem* (VRP) (CHRISTOFIDES, 1976) é um exemplo de problema de roteamento de veículos. Nele o objetivo da otimização é minimizar o custo da rota de cada veículo da frota visitando todas as localizações. As rotas devem iniciar e finalizar no mesmo local e todos os locais de parada são conhecidos antes de se iniciar o trajeto. Ainda, podem existir restrições que definem janelas de

atendimento, capacidade do veículo, distância máxima do percurso, etc. As restrições são opcionais e quando utilizadas, definem subproblemas do VRP. Um exemplo de variante com restrições é o *Vehicle Routing Problem with Time Windows* (VRPTW) (KONTORAVDIS; BARD, 1995; DESROCHERS; DESROSIERS; SOLOMON, 1992). As janelas de tempo são restrições com limitadores temporais para visitar o local, representando o início e o final do horário de atendimento. Por exemplo, imagine um supermercado que recebe suas encomendas entre 9:00 e 13:00 horas, assim sua janela de atendimento deve respeitar esse intervalo [9:00-13:00]. Outra variante do VRP é a *Capacitated Vehicle Routing Problem* (CVRP) (LYSGAARD; LETCHFORD; EGLESE, 2004). Esta variante adiciona uma restrição que indica a capacidade da cada veículo da frota e uma demanda (que seria o peso ou volume da carga) em cada local que se visita. Assim, a capacidade máxima de cada veículo não pode ser ultrapassada, pois o veículo não seria fisicamente capaz de transportar as cargas.

A variante *Vehicle Routing Problem with Profit* (VRPP) adiciona recompensas a cada local de visita. A recompensa é um incentivo para a visita, que é coletada apenas quando daquela visita é realizada. Cada local de visita possui um valor de recompensa diferente. A recompensa pode ser representada por uma vantagem financeira (receita, por exemplo). Outra forma de representar a recompensa seria uma pontuação que se pode receber ao visitar determinado ponto no contexto de uma competição esportiva, por exemplo. Por representar uma vantagem, a solução do VRPP busca maximizar a recompensa coletada. Assim, a recompensa pode fazer parte do objetivo da otimização como em Aksen e Aras (2006), Archetti, Speranza e Vigo (2014), Stavropoulou, Repoussis e Tarantilis (2019), buscando maximizar a recompensa coletada e minimizar o custo total da rota.

Outra abordagem é usado em problemas de roteamento de veículos é a *Team Orienteering Problem* (TOP) (CHAO; GOLDEN; WASIL, 1996). Nela, busca-se maximizar a recompensa coletada, mesmo que não sejam visitadas todas as localizações. O custo da rota pode ser limitado por uma restrição que limita máximo da distância percorrida. O TOP também possui variações que podem adotar janelas de tempo e capacidade. A diferença entre *Vehicle Routing Problem* (VRP) e *Team Orienteering Problem* (TOP) é que, no primeiro, busca-se minimizar o custo, na segunda busca-se maximizar a recompensa coletada.

A recompensa pode ser um fator importante em determinados tipos de operações de VRP. Por exemplo, imagine o seguinte cenário: uma equipe que trabalha em uma companhia elétrica realizando manutenção de unidades consumidoras. Cada unidade consumidora possui um gasto mensal diferente ou ainda, cada subestação de fornecimento de energia possui um número de unidades. Se ocorrer algum problema técnico em alguma dessas unidades, é mais inteligente dar prioridade para o local que

afeta mais unidades, mesmo que esse local seja o mais distante. Mesmo raciocínio pode ser feito para uma empresa de TV a cabo, onde seria mais interessante dar prioridade para prestação de serviço em um prédio com vários moradores em detrimento de uma única casa.

Este trabalho aborda a variante *Vehicle Routing Problem with Profit and Time Windows* (VRPPTW), que tem como objetivo a minimização da soma entre o custo e a recompensa não coletada. A motivação para se trabalhar com essa variante se dá por dois motivos: i.O objetivo da otimização, quando considera custo e recompensa, abrange novos e relevantes cenários no assunto roteamento de veículos; ii.A pouca atenção que recebe, pois é mais comum utilizar a abordagem TOP para lidar com a maximização das recompensas coletadas com ou sem restrição de custo e janela de tempo; ou o *Vehicle Routing Problem with Time Windows* (VRPTW) para minimização do custo das rotas com restrição de capacidade e eventualmente janela de tempo. Assim, neste caso, o problema consiste em criar rotas para uma frota de veículos com capacidade homogênea. As rotas devem percorrer locais que estão geograficamente distantes entre si. Cada local tem um valor de recompensa associado, que representa a vantagem (que pode ser financeira) em visitá-lo. O custo é proporcional à distância entre os locais. Não é necessário visitar todas as localidades. O tamanho da frota (número de veículos utilizados) não é fixo; assim busca-se minimizar o número de veículos utilizados. Além disso, leva-se em conta restrições de janelas de tempo única para cada local de visita.

Problemas de otimização, dentre eles o caso da variante VRPPTW, uma vez que herda a característica NP-difícil do VRP, podem necessitar de muitos recursos computacionais quando resolvidos por métodos exatos. Este fator estimula a utilização de metaheurísticas para a sua resolução. Metaheurísticas fornecem soluções satisfatórias para problemas mais complexos em um tempo razoável. Porém, ao contrário de métodos exatos, não há garantia que encontrem soluções ótimas, pois possuem um comportamento estocástico que não garante que todo o espaço de busca seja explorando (TALBI, 2009).

Este trabalho utiliza a metaheurística *Adaptive Large Neighborhood Search* (ALNS), que foi definida em (ROPKE; PISINGER, 2006). A escolha de tal metaheurística se dá pelos bons resultados alcançados em SCHMITT (2020), BAYER (2021), Sacramento, Pisinger e Ropke (2019), Liu, Tao e Xie (2019), onde também trabalharam-se problemas de roteamento de veículos. O ALNS é uma metaheurística de solução única. O seu funcionamento básico consiste em utilizar operadores que destroem e constroem soluções a cada iteração do algoritmo.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Este trabalho tem como objetivo a proposta de um método de solução do *Vehicle Routing Problem with Profit and Time Windows* (VRPPTW) utilizando como base a meta-heurística *Adaptive Large Neighborhood Search* (ALNS).

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

1. Apresentar uma revisão bibliográfica sobre os problemas de roteamento de veículos que utilizam custo e recompensa em suas funções objetivas.
2. Apresentar uma modelagem matemática para o *Vehicle Routing Problem with Profit and Time Windows* (VRPPTW).
3. Especificar e implementar um algoritmo baseado na metaheurística ALNS para resolver *Vehicle Routing Problem with Profit and Time Windows* (VRPPTW).
4. Definir a configuração dos parâmetros do ALNS que melhor resolvam a variante *Vehicle Routing Problem with Profit and Time Windows* (VRPPTW).
5. Realizar testes com *benchmark* da literatura para avaliar o desempenho do algoritmo desenvolvido.

1.2 METODOLOGIA

De acordo com a classificação metodológica encontrada na literatura, o tipo de pesquisa relacionado a este trabalho é de cunho exploratório, onde busca-se desenvolver algo melhor que o existente pois será proposto um método computacional diferente para a resolução de um problema da literatura (LAKATOS; MARCONI, 2007). Quanto ao procedimento técnico adotado, este trabalho se enquadra como uma Pesquisa-Ação, ou seja, a investigação e a experimentação são aplicadas de forma cíclica onde cada iteração gera informações aplicadas à evolução da próxima iteração (TRIPP, 2005).

Este trabalho inicia realizando uma revisão da literatura sobre problemas de roteamento de veículos, focando especificamente em VRP e TOP com a intenção de investigar quais características e/ou restrições são utilizadas em problemas de roteamento de veículos com recompensa. Com essas informações, este trabalho vai realizar a modelagem matemática do VRRPTW focado na função objetivo que minimize a soma entre o custo de deslocamento e a recompensa não coletada. Essa modelagem

vai contemplar também as restrições de capacidade, janela de tempo e tempo máximo da rota.

Com a definição, partiu-se para a modelagem matemática do problema e para a implementação do algoritmo *Adaptive Large Neighborhood Search* (ALNS). Neste trabalho foi utilizado uma função resfriamento inspirada no *Simulated Anneling* (SA). O *Adaptive Large Neighborhood Search* (ALNS) funciona selecionando operadores de destruição e reparo. Cada tipo de operador vai possuir um peso. O valor do peso indica o quanto aquele operador obteve sucesso anteriormente em melhorar a solução. Assim o algoritmo pode escolher operadores diferentes dependendo a instância do problema, permitindo que o algoritmo se adapte.

Com o algoritmo pronto, foi utilizado um *benchmark* da literatura para testar a qualidade das soluções geradas. O *benchmark* que será utilizado foi proposto por Solomon (1987). O *benchmark* foi adaptado, onde houve a adição de recompensa proporcional a distância entre a requisição e o deposito. Um segundo experimento foi realizado, onde se executou o algoritmo com a função objetivo levando em consideração apenas o custo e a coleta dos valores de custo e recompensa não coletada. Após isso, houve a execução o algoritmo com função objetivo considerando a soma do custo e a recompensa não coletada. A intenção da comparação entre as duas funções objetivo é verificar o desempenho do algoritmo nessas duas situações. A hipótese é que o custo de deslocamento vai aumentar, mas a recompensa coletada vai aumentar também. A fim de dar maior relevância estatística dos resultados da comparação, foi utilizado o teste estatístico de Wilcoxon. Além disso houve uma análise qualitativa, onde foi analisada as curvas de convergência e a curva de resfriamento do *Simulated Anneling* (SA).

1.3 CONTRIBUIÇÕES

Este trabalho apresentas as seguintes contribuições:

- Formular a variante *Vehicle Routing Problem with Profit and Time Windows* (VRPPTW) com restrições de capacidade e tempo máximo da rota, além das janelas de atendimento. Na literatura existe apenas a variante *Vehicle Routing Problem with Profit* (VRPP).
- Adaptar um *dataset* de instâncias, originalmente desenvolvidas para o *Vehicle Routing Problem with Time Windows* (VRPTW), para o *Vehicle Routing Problem with Profit and Time Windows* (VRPPTW). As instâncias do VRPTW não tem recompensa atrelada a cada requisição. Assim, foram acrescentados valores de

recompensa para cada instância, baseados em um cálculo de recompensa segundo a literatura da área.

1.4 ESTRUTURA DO TRABALHO

Esse trabalho está organizado da seguinte forma. O Capítulo 2 apresenta uma revisão dos conceitos necessários para o desenvolvimento dessa pesquisa. Nele é revisada a bibliografia sobre os problemas de roteamento de veículos, focando naquelas que apresentam custo ou recompensa em suas funções objetivos. O Capítulo 3 apresenta o método proposto utilizando o algoritmo ALNS aplicado ao problema VRPPTW. O Capítulo 4 apresenta os resultados dos experimentos realizados, incluindo a adaptação de uma base de dados de requisições de serviço para VRP para realização do *benchmark* da literatura. Por fim, o Capítulo 5 apresenta as conclusões e propostas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico necessário sobre os assuntos de problemas de roteamento de veículos e suas abordagens (Seção 2.1). Também é apresentado ainda conceitos sobre metaheurísticas, metaheurística de solução única e ALNS (Seção 2.2). Na Seção 2.3 é feita uma apresentação dos operadores utilizados no ALNS. Por último, é apresentada uma revisão dos trabalhos relacionados na literatura, onde são utilizadas metaheurísticas no contexto do problema do roteamento de veículos (Seção 2.4).

2.1 PROBLEMAS DE ROTEAMENTO DE VEÍCULOS

A importância dos problemas de roteamento de veículos surge no contexto de logística. Empresas de logística ou quem precisa manter uma rede logística em funcionamento, e que dispõe de ferramentas para planejamento de boas rotas de distribuição tem um fator de sucesso e competitivo maior que suas concorrentes. A utilização de processamento computadorizado no processo de planejamento da distribuição diminui entre 5 % e 20 % dos gastos com transporte (TOTH; VIGO, 2002). Típicas aplicações que lidam com esse problema são coleta de lixo, limpeza de ruas, rotas de ônibus, sistemas entregas/coletas sob demanda (ex: ifood, uber, etc), criação de rotas para representantes comerciais, unidades de manutenção (ex: empresas elétricas, TV a cabo, Internet) e etc.

O problema é descrito da seguinte forma: existe um conjunto de veículos (frota) que precisam se locomover até um conjunto de clientes localizados geograficamente em locais diferentes. Os veículos precisam partir e retornar a locais pré definidos, chamados depósitos. O objetivo do problema é criar a rota para cada um dos veículos, onde busca-se atender todos os clientes ou o máximo possível deles, atendendo todas as restrições dos clientes e minimizando o custo total da frota.

A malha viária pode ser representada por um grafo, onde as arestas representam o caminho entre os clientes e os vértices representam o local onde estão os clientes e depósitos. Arestas podem ser dirigidos ou não dirigidos, dependendo da forma como a malha é modelada. Arestas dirigidas podem representar limitações da malha real, onde por exemplo, a existência de apenas um sentido na rua, etc. Arestas também possuem um custo que está associado ao tempo de viagem e tipo de veículo (TOTH; VIGO, 2002).

Clientes são representados por vértices no grafo e além de possuir a sua posição geográfica, podem ter características diversas, como as seguintes, por exemplo:

- Demanda: cada cliente possui uma demanda que representa o volume, peso ou quantidade da carga dos pacotes que vão ser entregues/coletados na visita. Cada veículo vai ter uma capacidade, representando suas dimensões físicas. Se todos os veículos possuírem a mesma capacidade, a frota é homogênea. Caso contrário, é heterogênea.

- Janelas de atendimento: No mundo real, nem todos os estabelecimentos atendem 24 horas por dia, ou seja, funcionam em apenas um período de tempo. A janela de atendimento serve justamente para representar essa limitação. Se um cliente fica aberto entre 09:00 e 18:00, sua janela de atendimento pode ser representada por [09:00-18:00], por exemplo. Uma janela de tempo pode representar também uma limitação de tráfego. Se determinado bairro possui muito trânsito em determinados horários, pode não valer a pena fazer a visita nesse horário. Por último, clientes podem ter mais de uma janela de tempo, como um estabelecimento que recebe suas cargas apenas pela manhã e à noite.

- Tempo de serviço: tempo necessário para realizar o carregamento ou descarregamento da carga. Um caminhão vai levar muito mais tempo para descarregar do que uma motocicleta leva para entregar envelopes. O tempo de serviço depende das características da frota. Em alguns casos, pode ser considerado que todos os veículos têm o mesmo tempo de serviço para todas as visitas.

Dependendo do cenário, não é possível atender a todos os clientes. Assim, é necessário realizar a seleção de quais clientes devem ser visitados. A forma mais simples para isso é atrelar uma recompensa ou penalidade para cada cliente. A recompensa é coletada apenas no caso da visita e a penalidade no caso da visita não acontecer. Assim, o problema ganha restrições que garante que uma recompensa mínima seja coleta ou o montante de penalidade não ultrapasse um limite máximo (TOTH; VIGO, 2002).

As rotas também podem conter restrições. Algumas não podem ter mais que determinado comprimento ou tempo de viagem, por questões legais, por exemplo. Um motorista não pode trabalhar mais do que oito horas por dia ou não pode trafegar mais do que uma quantidade quilômetros, pois a legislação considera como viagem intermunicipal e é necessário o pagamento de taxas extras, acarretando em mais custos para a empresa. Cada rota pode ter um custo fixo atrelado ao seu funcionamento. Por exemplo, considere o gasto para alocar um funcionário ou alugar/comprar veículo apenas para atender aquela rota. Esse custo serve para desincentivar a criação de novas rotas. Mas a partir do momento que o gasto para alocar visitas em outras rotas for maior do que o gasto de apenas criar uma nova, essa nova rota deve ser criada, pois representa diminuição dos custos.

No quesito restrições, é importante citar como exemplos o tamanho máximo do comprimento das rotas; a capacidade máxima da carga do veículo, o atendimento das janelas de tempo dos clientes e etc. Em algumas abordagens, pode-se ter a possibilidade da mesma rota conter coleta e entregas de encomendas. Nesse caso, aqui existe a restrição que precisa garantir que a visita ao local de coleta da encomenda seja feita antes da visita ao local de entrega, e também é preciso garantir que o veículo que realizou a coleta é o mesmo que realizá a entrega. Outro exemplo é que determinadas cargas podem possuir um tempo limite de entrega, como produtos perecíveis ou um sistema de distribuição de sangue doado (LESTARI, 2021).

O principal esforço a ser realizado para a solução do problema do roteamento de veículos é o objetivo da otimização, modelado como uma função objetivo. A função objetivo não impõe valores limites superiores ou inferiores como uma restrição para a solução do problema. No cenário de roteamento de veículos, é importante minimizar o custo, logo, a função objetivo expressa o somatório dos custos das rotas e o objetivo do algoritmo é encontrar variações das rotas que tenham o custo mais baixo possível. Outros objetivos podem ser a minimização de veículos utilizados e minimização de penalidades (TOTH; VIGO, 2002).

Em um cenário em que a recompensa adquirida ao se visitar os clientes é mais importante do que o custo para chegar até ele, a função objetivo é expressa pelo somatório das recompensas coletadas pelos veículos e, portanto, o foco se torna em maximizar a recompensa total coletada. Outros objetivos de maximização podem ser a quantidade de visitas (cenário onde se recebe um valor fixo por coleta/entrega) ou maximizar o peso transportado (em um cenário onde se ganha por quilo ou tonelada transportada).

2.1.1 Formulação Matemática do *Vehicle Routing Problem* (VRP)

Nesta seção será apresentada a formulação matemática do *Vehicle Routing Problem* (VRP), além da demonstração de algumas de suas variantes. O problema foi originalmente proposto em Dantzig e Ramser (1959) com o nome *The Truck Dispatching Problem*. Em Christofides (1976) foi atribuído seu nome clássico de Vehicle Routing Problem, pois antes dele, dependendo do artigo, o nome variava. A formulação matemática apresentada para este trabalho é uma adaptação do trabalho Laporte (2009).

Considere o grafo $G = (V, A)$, onde $V = \{1, ..., n\}$ é o conjunto de vértices e $A = \{(i, j) : i, j \in V, i \neq j\}$ é o conjunto de arestas. O vértice 0 representa o depósito de onde os veículos devem iniciar e finalizar sua rota. A frota F de tamanho m é composta com veículos idênticos e m é conhecido com antecedência. Um custo c_{ij} é associado a aresta entre os clientes i e j. O custo pode ser definido como custo

de viagem, distância ou tempo de viagem. O VRP consiste em designar m rotas, onde cada rota inicia e finaliza no depósito, cada cliente é visitado apenas uma vez por apenas um veículo, buscando minimizar o custo total.

Assim, o problema é formulado da seguinte forma:

$$\text{Minimize:} \sum_{f \in F} \sum_{(i,j) \in A} c_{ij} x_{fij} \tag{2.1}$$

$$\text{Sujeito à:} \sum_{j=1}^{n} x_{0j} = 2m \tag{2.2}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2; k \in V \backslash \{0\} \tag{2.3}$$

$$x_{0j} = 0, 1, 2; j \in V \backslash \{0\} \tag{2.4}$$

$$x_{i,j} = 0, 1; (i, j \in v \backslash \{0\}) \tag{2.5}$$

A Função Objetivo 2.1 busca minimizar o custo total. O custo total é dado pela soma do custo da rota de cada veículo. A variável x_{fij} recebe valor 1 quando o veículo f viaja entre os clientes i e j e 0 caso contrário. A Equação 2.2 garante que todos os veículos que saíram do depósito voltem até ele [1]. A Equação 2.3 garante que o veículo que visita um vértice qualquer saia por outro caminho. A Equação 2.4 garante que variável x_{0j} tem apenas os valores 0 (caso nenhum veículo tenha passado pela aresta do depósito), 1 (um veículo saiu ou entrou no deposito) ou 2 (caso o veículo tenha usado a mesma aresta para entrar e sair do depósito). A Equação 2.5 indica que a aresta que não está ligada ao depósito apenas possa ser usada uma vez por um veículo.

Repare que não existem restrições que tratam de capacidade, coleta de recompensa ou comprimento máximo de rotas. O problema originalmente apenas trata da minimização do custo e a adição de outras restrições acaba por gerar variantes do problema.

A variante *Capacitated Vehicle Routing Problem* (CVRP) adiciona uma demanda d_i para cada cliente i, esse cliente é representado por um vértice v_i. Além disso, cada veículo f tem associada a si uma capacidade q_f que indica quanta carga consegue carregar. A demanda e capacidade devem ser expressas na mesma unidade de medida, como peso, volume ou quantidade e a capacidade máxima da frota pode ser expressa por Q. Como é comum que a frota seja homogênea, a capacidade de cada veículo é dada por Q/m. A Equação 2.6 indica que o somatório das cargas de

[1] A notação $V \backslash \{0\}$ é equivalente ao conjunto V sem o vértice do depósito.

todos os veículos não pode ultrapassar a capacidade total da frota e é adicionada as Equações 2.1-2.5 para definir o CVRP. A variável y_i indica se o vértice i foi visitado, 1 caso verdade; 0 caso contrário (BEN-SAID; EL-HAJJ; MOUKRIM, 2019).

$$\sum_{i \in V} d_i y_i <= Q \tag{2.6}$$

No caso do *Vehicle Routing Problem with Time Windows* (VRPTW) são adicionadas janelas de tempo ao problema, que são restrições que evitam que visitas sejam feitas fora do horário correto para atendimento da demanda - descarga ou carga - por exemplo. Uma janela de tempo do cliente i pode ser representada por $[s_i, e_i]$, onde s_i é o início da janela de tempo e e_i é o término da janela de tempo. A variável v_{if} representa o momento que o veículo f iniciou a visita e t_{ij} o tempo que se leva para percorrer a aresta entre i e j. A Restrição 2.7 garante que a visita do veículo k inicie entre o início e final da janela de tempo do cliente i. A Restrição 2.8 estabelece a relação entre o horário de saída do veículo de um cliente (vértice) e seu sucessor imediato, onde a soma do início de atendimento anterior somado ao tempo de deslocamento até o próximo vértice deve ser menor ou igual do que o tempo de início do atendimento do próximo vértice. Em outras palavras, o veículo não pode iniciar um atendimento antes de chegar no cliente (KALLEHAUGE et al., 2005).

$$s_i \leq v_{if} \leq e_i; i \in V, f \in F \tag{2.7}$$

$$x_{fij}(v_{if} + t_{ij} - v_{jf}) \leq 0; (i,j) \in V, f \in F \tag{2.8}$$

Existem ainda várias outras variantes do VRP, cada uma com seu conjunto de restrições. Restrições de capacidade, janela de tempo, recompensa mínima coletada, etc., podem ser combinadas para formar outras variantes. As variantes CVRP e VRPTW servem como exemplos, onde suas características (representadas pelas restrições) são adicionadas ao VRP. Outra variante que será abordada será o VRPP na Seção 2.1.3. Esta variante ganha mais atenção por apresentar a adição de recompensa em sua função objetivo, característica importante para este trabalho. Outra abordagem para o problema de roteamento de veículos, que considera a recompensa como fator importante, é a do TOP, que será visto na Seção 2.1.2, cuja função objetivo busca maximizar a recompensa coletada pela frota.

2.1.2 *Team Orienteering Problem* (TOP)

O *Team Orienteering Problem* (TOP) foi proposto originalmente em (CHAO; GOLDEN; WASIL, 1996) como solução para o problema de alocação de membros de

uma equipe esportiva. O TOP busca maximizar a recompensa coletada ao contrário do VRP, onde busca-se minimizar o gasto total da frota. No TOP, o custo pode ser representado por uma restrição com valor máximo, que não pode ser ultrapassado. A função objetivo apenas considera a recompensa a coleta que quanto maior melhor.

Assim, dado um conjunto de $V = 1, ..., n$ de vértices e um conjunto A de arestas, que formam um grafo $G = (V, A)$, cada vértice i possui uma recompensa r_i positiva, que representa a vantagem ao visitá-lo. Cada aresta entre os vértices i e j possui um custo $c_{i,j}$. A frota F possui tamanho m e seu tamanho é conhecido desde o início. O vértice 1 representa o depósito que os veículos devem iniciar a rota e, dependendo da modelagem, o vértice final pode ser igual ao inicial ou diferente. Por conta disso, o vértice final é representado por n. Cada rota não pode ter tempo maior que T_{max}. Neste problema, custo e tempo são equivalentes. Cada recompensa só é coletada uma vez por um único veículo. Assim, não faz sentido visitar o mesmo vértice mais de uma vez. As Equações 2.9 até a 2.16 modelam o problema de otimização TOP.

$$\max: \sum_{i=2}^{n-1} \sum_{f=1}^{m} r_i y_{if} \tag{2.9}$$

$$\text{Sujeito a:} \sum_{j=2}^{n} \sum_{f=1}^{m} x_{1jf} = \sum_{i=1}^{n-1} \sum_{f=1}^{m} x_{ink} = m \tag{2.10}$$

$$\sum_{i<J} x_{ijf} + \sum_{i>j} x_{jif} = 2y_{jk}, (j = 2, ..., n-1; f = 1, ..., m) \tag{2.11}$$

$$\sum_{f=1}^{m} y_{if} \leqslant 1, (i = 2, ..., n-1) \tag{2.12}$$

$$\sum_{i=1}^{n-1} \sum_{j>i} c_{ij} x_{ijf} \leqslant T_{max}, (f = 1, ..., m) \tag{2.13}$$

$$\sum_{\substack{i,j \in U \\ i<j}} x_{ijf} \leqslant |U| - 1, (U \subset V - \{1, n\}; 2 \leqslant |U| \leqslant n-2; f = 1, ..., m) \tag{2.14}$$

$$x_{ijf} \in \{0, 1\}, (1 \leqslant i < j \leqslant n; f = 1, ..., m) \tag{2.15}$$

$$y_{1f} = y_{nf} = 1, y_{if} \in 0, 1 (i = 2, ..., n-1; f = 1, ..., m) \tag{2.16}$$

A função objetivo é representada pela Equação 2.9, onde busca-se maximizar a coleta de recompensa pela frota. A variável y_{if} representa se o vértice i foi visitado pelo veículo f. Se a visita ocorreu, a variável tem valor 1 e caso contrário, tem valor 0. A Equação 2.10 indica que cada rota deve iniciar no vértice inicial e finalizar no

vértice final, e a quantidade de vezes que os vértices 1 e n são visitados deve ser igual ao número de veículos. A Equação 2.11 garante a conectividade das rotas. Já a Equação 2.12 certifica que cada vértice (exceto o inicial e o final) deve ser visitado no máximo uma vez. A Equação 2.13 descreve o tempo máximo em que cada rota deve ser concluída. A Equação 2.14 certifica que a existência de sub-caminhos seja proibida. As Equações 2.15 e 2.16 garantem a integridade das variáveis binárias.

Assim, como no caso do VRP, a adição de novas restrições acaba por definir novas variantes. A adição de uma restrição de capacidade máxima, como a Equação 2.6 define a variante *Capacitated Team Orienteering Problem* (CTOP) (ARCHETTI et al., 2009). A adição de restrições para janela de tempo, como as Equações 2.7 e 2.8 define a variante *Team Orienteering Problem with Time Windows* (TOPTW) (LABADIE et al., 2012).

2.1.3 *Vehicle Routing Problem with Profit* (VRPP)

A principal característica da classe do *Vehicle Routing Problem with Profit* (VRPP) é a de que à cada cliente está associada uma recompensa que o torna mais ou menos atrativo para ser visitado. Assim, qualquer rota, ou conjunto de rotas, começando e terminando em um determinado depósito, pode ser medida tanto em termos de custo quanto em termos de lucro. A diferença entre o lucro e o custo da rota pode ser maximizada, ou apenas um dos dois pode ser utilizado na função obejtivo, enquanto o outro pode ser representado em uma restrição (ARCHETTI; SPERANZA; VIGO, 2014).

Dado o grafo $G = (V, A)$, onde $V = \{1, ..., n\}$ é o conjunto de vértices e $A = \{(i, j) : i, j \in V, i \neq j\}$ é o conjunto de arcos. O vértice 0 representa o depósito de onde os veículos devem iniciar e finalizar sua rota. A frota F de tamanho m é composta com veículos idênticos e m é conhecido com antecedência. Os custos c_{ij} e t_{ij}, que representam respectivamente a distância e o tempo, são associados ao arco entre os clientes i e j. A variável r_i indica a recompensa coletada ao visitar o cliente i. A variável T_{max} representa o tempo máximo que cada rota não pode ultrapassar e R_{min} representa o valor mínimo de recompensa coletado que uma rota deve ter. As variáveis binárias x_{ij} e y_i indicam se a aresta entre os clientes i e j foi utilizada e se o cliente i foi visitado, respectivamente. O VRPP consiste em designar m rotas, onde cada rota inicia e finaliza no depósito e cada cliente é visitado apenas uma vez por apenas um veículo, buscando maximizar a diferença entre a recompensa total coletada e o custo total. A modelagem apresentada a seguir é uma adaptação de Archetti, Speranza e Vigo (2014), Aksen e Aras (2006).

$$\text{Maximize:} \sum_{i \in V} r_i y_i - \sum_{i \in E} \sum_{j \in E} c_{ij} x_{ij} \tag{2.17}$$

$$\text{Sujeito:} \sum_{i \in A; i \neq j} x_{ij} = \sum_{i \in A; i \neq j} x_{ji}; i \in A \tag{2.18}$$

$$\sum_{i \in A; i \neq i} x_{ij} = y_i; i \in A \backslash \{0\} \tag{2.19}$$

$$\sum_{i \in A} \sum_{j \in A} x_{ij} \geq y_i \tag{2.20}$$

$$\sum_{i \in A} \sum_{j \in A} t_{ij} x_{ij} \leq T_{max} \tag{2.21}$$

$$\sum_{i \in E} r_i y_i \geq R_{min} \tag{2.22}$$

$$y_i \in 0, 1; i \in V \tag{2.23}$$

$$x_{ij} \in 0, 1; (i, j) \in A \tag{2.24}$$

A função objetivo é representada em 2.17. Nela busca-se maximizar a diferença entre recompensa total coletada e custo total. Repare que como a função objetivo é uma diferença entre somatórios, podem existir valores negativos. A diferença pode representar o lucro obtido, onde o primeiro termo é a receita (pagamento pelos serviços prestados) e o segundo termo representa os gastos operacionais. A Equação 2.18 garante que um veículo que tenha feito a visita, saia do cliente. A Equação 2.19 garante que um cliente visitado tenha uma arco de saída do mesmo. A eliminação de sub-caminhos é feita pela Equação 2.20. A Equação 2.21 garante que a rota não ultrapasse um limite máximo de tempo. A Equação 2.22 garante que um valor mínimo de recompensa seja coletada. Essas duas últimas restrições são opcionais e dependem do cenário em que se trabalha. Por último, as Equações 2.23 e 2.24 garantem integridade das variáveis binárias.

2.2 METAHEURÍSTICAS

Esta seção apresenta conceitos sobre metaheurísticas. A Seção 2.2.1 apresenta conceitos básicos sobre o funcionamento de metaheuristicas. Na Seção 2.2.2 é apresentado o funcionamento básico de uma metaheurística baseada em solução única. Na Seção 2.2.4 será apresentado a fundamentação sobre o algoritmo ALNS.

2.2.1 Conceitos básicos de metaheurísticas

As metaheurísticas são uma classe de algoritmos genéricos de aproximação muito utilizados em problemas de otimização. Um problema de otimização é aquele que busca maximizar ou minimizar determinadas características relacionadas à solução do problema. Por exemplo, em problemas de roteamento de veículos, busca otimizar o custo total da rota, minimizando o seu valor. Ao contrário de métodos exatos, metaheurísticas conseguem resolver problemas grandes podendo entregar soluções satisfatórias em tempo razoável. Porém, não apresentam garantia de encontrar a melhor solução para o problema. As metaheurísticas possuem como características o comportamento estocástico que não garante que todo o espaço de busca seja percorrido. Os métodos exatos, por sua vez, exploram exaustivamente todo espaço de busca. Metaheurísticas ganharam muita popularidade nas últimas décadas por conseguirem apresentar soluções efetivas para variados tipos de problemas. Entre suas aplicações pode-se citar, como exemplo, otimização de topologia, otimização estrutural em eletrônica, aerodinâmica, bioinformática, planejamento em problemas de roteamento, problemas de alocação, etc. (TALBI, 2009; GENDREAU; POTVIN et al., 2010).

A forma de representação da solução desempenha um papel fundamental na sua eficiência e efetividade. A escolha deve levar em consideração características pertinentes para o cenário tratado. A escolha de uma má representação da solução dificulta a sua implementação e impacta a qualidade da solução gerada pelo algoritmo (TALBI, 2009; GENDREAU; POTVIN et al., 2010). Para o problema de roteamento de veículos, por exemplo, as características importantes que devem constar na representação da solução são a ordem dos locais de visita e em quais rotas essas visitas devem aparecer. É comum representar as rotas como uma matriz onde cada linha seria a rota e cada posição representa o local de visita. Na Figura 1 é possível ver um exemplo da representação de solução do problema contendo várias rotas. A rota 1 contém os vértices 1, 10, 4 e 5. Cada um dos vértices é uma representação do local onde o veículo deve realizar a visita. Ali podem ser representados o valor de demanda, recompensa, coordenadas do local, entre outros.

Outro ponto que gera impacto na metaheurística é a definição da função objetivo. Ela formula o objetivo a ser alcançado e associa cada solução com um valor real que descreve a sua qualidade. A função objetivo representa um importante elemento na modelagem do problema, pois guia a busca em direção a boas soluções (TALBI, 2009; GENDREAU; POTVIN et al., 2010). Por exemplo, se a solução for representada como uma matriz, onde cada linha representa uma rota e as colunas as visitas, as células dessa matriz terão o custo da distância para se visitar cada cliente, que se não for visitado pela rota terá custo 0, a função objetivo soma todas as distâncias e apresenta um valor que indica a qualidade da solução (um exemplo pode ser visto na

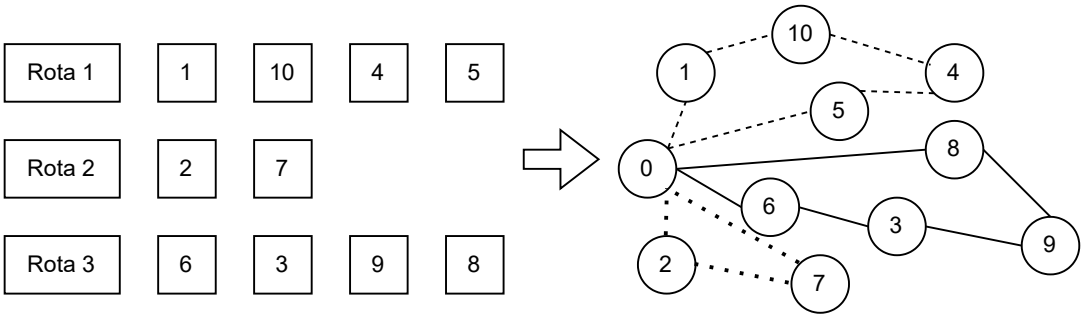

Figura 1 – Exemplo de representação de solução

Fonte: Autor (2022).

Figura 2). Nesse exemplo, valores altos representam soluções ruins e valores baixos representam soluções boas.

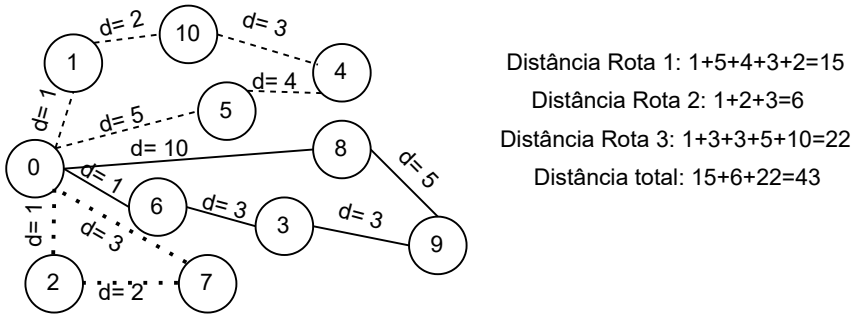

Figura 2 – Exemplo de cálculo da função objetivo

Fonte: Autor (2022).

Como é possível definir um valor numérico para cada solução gerada, é possível criar um ranking das soluções. A melhor solução encontrada durante toda a execução da metaheurística é chamada de ótimo global. Esta solução é apresentada ao final da execução e todas as outras são descartadas, pois são piores que a ótimo global.

No desenvolvimento de metaheurísticas, dois fatores contraditórios devem ser analisados: a diversificação e a intensificação. A diversificação é a exploração do espaço de busca. O espaço de busca são todas as possíveis soluções que se pode gerar relativas ao problema em questão. Ao diversificar, buscamos criar soluções ainda não criadas ou alterar adequadamente as características dela. Na intensificação, regiões promissoras são exploradas com mais detalhamento tentando melhorar uma solução o máximo possível, criando pequenas perturbações, mas que não alteram muito as características. Quando uma metaheurística apenas se preocupa com a intensificação, é muito provável que se caia no problema de encontrar apenas as mesmas soluções que podem não ser os ótimos globais. Se o foco for a diversificação, várias partes do espaço de busca podem ser exploradas sem muito aprofundamento, o que pode levar ao descarte de soluções promissoras. Um equilíbrio entre os dois fatores é o ideal.

Assim, aplicar um processo de intensificação e um processo de diversificação, intercaladamente, pode trazer bons resultados (TALBI, 2009; GENDREAU; POTVIN et al., 2010).

2.2.2 *Metaheurística Baseada em Solução Única* (MBSU)

Para solucionar problemas de otimização, as Metaherísticas Baseadas em Solução Única (MBSUs) trabalham apenas com uma solução por vez (iteração). Em cada iteração da MBSU são geradas pequenas perturbações na solução que resultam em outras soluções que compartilham muitas características. A melhor destas é selecionada e é utilizada como base para as perturbações da próxima iteração (TALBI, 2009; GENDREAU; POTVIN et al., 2010). Antes de definirmos uma MBSU propriamente, é importante comentar seus conceitos básicos.

O primeiro deles é o conceito de operador. Um operador gera perturbações em uma solução. A perturbação pode ser a adição, alteração ou remoção de características da solução. Cada operador gera perturbações diferentes e cada perturbação gera uma nova solução. A perturbação pode ser aleatória ou seguir um critério ou heurística. Um exemplo de operador é a troca de posição entre localizações em uma mesma rota. A troca pode ser feita utilizando posições aleatórias ou ainda calculando qual troca vai ter a maior diminuição no custo da rota. Outros exemplos são remoção aleatória, remoção da pior localização (mais distante, por exemplo), alteração de ordem de localizações entre duas rotas, adição aleatória, adição do melhor, etc.

O conjunto de todas as soluções geradas pela aplicação do operador é chamada de vizinhança. A vizinhança é a gerada pela aplicação de um operador em uma solução. Levando em consideração o operador de troca de posições das visitas (ordem) na rota citada anteriormente, a vizinhança é o conjunto de todas as soluções com a troca de um par de localizações (Figura 3). Repare que a perturbação na solução executada pelo operador é pequena, gerando uma solução parecida com a solução original. Por isso, o conjunto se chama vizinhança. Importante destacar que operadores diferentes geram vizinhanças diferentes (TALBI, 2009; GENDREAU; POTVIN et al., 2010).

Na Seção 2.2.1 apresentou-se o conceito de ótimo global, mas é importante definir o ótimo local. Ótimo local é o caso de uma solução, que ao aplicar um operador e gerar sua vizinhança, não resulta em um vizinho melhor, ou seja, é a melhor solução para aquela região do espaço de busca. A Figura 4 ilustra os conceitos de ótimo local e global. Encontrar um ótimo local não é necessariamente bom, pois busca-se o ótimo global. Aqui entra o comportamento de intensificação e diversificação. A intensificação busca alcançar o ótimo local da região no espaço de busca e a diversificação busca perturbar a solução para que ela vá para outra região, que pode ser a região do ótimo

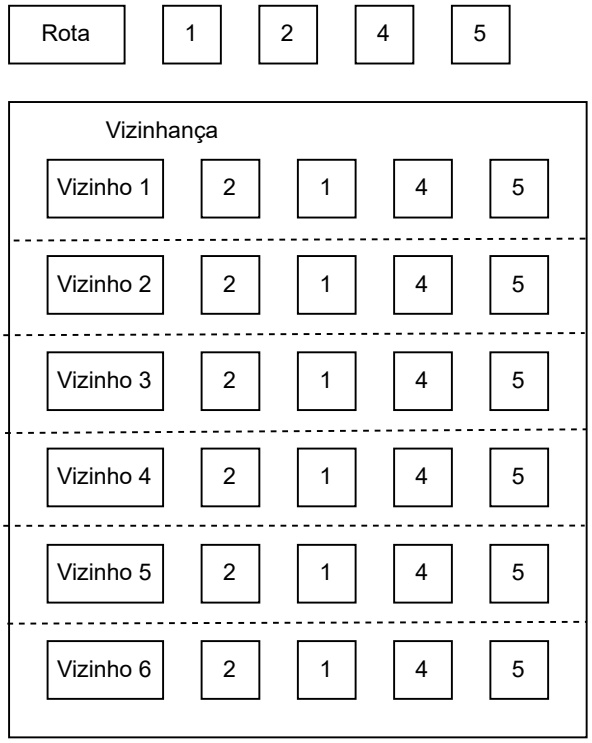

Figura 3 – Exemplo de geração de vizinhança com operador que troca a ordem das visitas

Fonte: Autor (2022).

global.

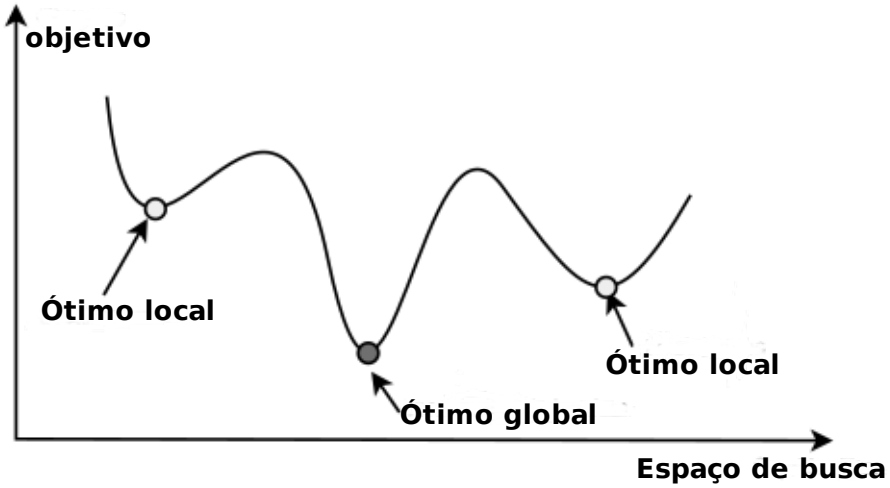

Figura 4 – Ótimo local e ótimo global em um espaço de busca contínuo. Um problema pode ter muitas soluções ótimas globais. O objetivo é minimizar.

Fonte: (TALBI, 2009).

Outro ponto que merece atenção em MBSU é a estratégia de criação de solução. Como os operadores necessitam de soluções de entrada para gerar a vizinhança, é necessário criar uma solução inicial. Duas principais estratégias são utilizadas: a abordagem aleatória e a abordagem gulosa. Existe um equilíbrio para uso de cada

uma que impacta a qualidade da solução e custo computacional. Gerar uma solução inicial aleatoriamente é rápido, mas pode fazer a metaheurística demorar muito tempo para convergir para uma boa solução. Criar uma solução gulosa, pode levar a metaheurística necessitar de menos iterações para chegar no ótimo local. No entanto, isso não significa que usar soluções melhores como soluções iniciais sempre levará a melhores ótimos locais. A utilização de uma estratégia híbrida leva a melhores resultados (TALBI, 2009; GENDREAU; POTVIN et al., 2010).

Para exemplificar os passos básicos de uma MBSU apresenta-se a *Local Search* (LS) (AARTS; AARTS; LENSTRA, 2003). Essa metaheurística inicia com uma solução inicial e a cada iteração ocorre a troca por uma solução vizinha que melhore a função objetivo. A LS finaliza quando todos os vizinhos são piores do que a solução atual, o que indica que chegou ao ótimo local. Usar apenas a vizinhança para a troca de solução busca tornar o processo mais rápido. Variações de LS se diferenciam no quesito de seleção de vizinhos e na estratégia de fuga do ótimo local.

No quesito de seleção de vizinhos existem três estratégias principais: i) Melhor mudança. Neste caso, o vizinho gerado pelo operador que melhor incrementa a qualidade da solução é selecionado, mas é necessário gerar toda vizinhança. ii) Primeira mudança. Nesta estratégia, o primeiro vizinho gerado que melhorar a solução vai ser selecionado, sem necessidade de continuar gerando a vizinhança. iii) Mudança aleatória. Nesta estratégia, uma seleção aleatória é aplicada aos vizinhos que melhoram a solução atual (TALBI, 2009; GENDREAU; POTVIN et al., 2010).

No quesito estratégia de fuga existem quatro abordagens. As quatro abordagens são: i) Reiniciar o processo com soluções iniciais diferentes. Nesse caso, cria-se a solução inicial, aplica-se o operador até chegar no ótimo local. Depois cria-se nova solução e reinicia-se o processo. ii) Aceitar vizinhos piores. Aqui busca-se aceitar vizinhos piores que a solução atual no caso em que não for possível encontrar uma solução melhor. iii) Mudar a vizinhança. Esta estratégia consiste em mudar a estrutura da vizinhança durante o processo para explorar melhor o espaço de busca. Essa alteração da vizinhança envolve operadores que alteram bastante a solução atual, gerando soluções com características bem diferentes iv) Mudar a função objetivo ou os dados de entrada do problema. Nesta estratégia, o problema é transformado, perturbando os dados de entrada do problema, a função objetivo ou as restrições, na esperança de resolver de forma mais eficiente o problema original (TALBI, 2009; GENDREAU; POTVIN et al., 2010).

No Algoritmo 1 está o *template* para uma LS genérica. O algoritmo inicia recebendo na linha 1 a solução inicial como entrada e inicia o contador de iterações na linha 2. Entre as linhas 3 e 9 ocorre um laço de repetição que busca gerar a vizinhança aplicando um operador (linha 5) e seleciona uma solução para substituir a

solução atual (linha 7). O critério de parada é diferente entre implementações, mas é comum utilizar número de iterações. Na linha 10, a melhor solução encontrada é devolvida pelo algoritmo.

Algoritmo 1 Template de *Metaheurística Baseada em Solução Única* (MBSU)

1: **Input:**Solução inicial s_0
2: $t = 0$
3: **repeat**
4: /* Gerar vizinhança (parcial ou completa) a partir de s_t*/
5: $v_t = aplicarOperador(s_t)$
6: /*Selecionar uma solução para substituir atual*/
7: $s_{t+1} = seleciona(v_t)$
8: $t = t + 1$
9: **until** Critério de parada não for satisfeito
10: **Output:** Melhor solução encontrada

Fonte: Adaptado de Talbi (2009).

2.2.3 *Simulated Anneling* (SA)

Simulated Anneling (SA) surgiu nos trabalhos de (KIRKPATRICK; JR; VECCHI, 1983; ČERNỲ, 1985). O SA baseia-se nos princípios da mecânica, onde o sistema inicia com uma temperatura alta e, em seguida, com o resfriamento lento o material passa a obter uma estrutura cristalina forte. Se a temperatura inicial não for suficientemente alta ou for aplicado um resfriamento rápido, são obtidas imperfeições. O algoritmo SA simula as mudanças de energia em um sistema submetido a um processo de resfriamento até convergir para um estado de equilíbrio (TALBI, 2009; GENDREAU; POTVIN et al., 2010). Aplicando esse conceito para um algoritmo, quando o procedimento é iniciado, ele está “quente”, ou seja, permite que ocorram trocas que possam piorar a solução atual. Geralmente é representado pela possibilidade de substituir a solução atual por uma pior, como uma estratégia de diversificação. Com o passar das iterações o sistema resfria, diminuindo a possibilidade de aceitar a piora da solução atual.

A partir de uma solução inicial, o SA realiza várias iterações. Em cada iteração, uma vizinhança é gerada. Vizinhos melhores (que têm valor da função objetivo melhor) sempre são selecionados. No caso em que o vizinho é pior, este pode ser selecionado dada uma probabilidade. A probabilidade varia dependendo da temperatura e da variação da função objetivo. A probabilidade de aceitação geralmente é dada pela distribuição de Boltzmann (TALBI, 2009) e está representada na Equação 2.25.

$$P(sol, novaSol, T) = e^{-\frac{f(novaSol) - f(sol)}{T}} \tag{2.25}$$

A variável T indica a temperatura do sistema e determina a chance de aceitar soluções que não melhoram. A temperatura inicia com valor relativamente alto para permitir que no início várias soluções piores sejam aceitas. Mas com o passar das iterações, o seu valor vai diminuindo e por consequência, diminuindo a probabilidade de aceitar soluções piores. A variável $f(novaSol)$ é o valor da função objetivo da nova solução gerada e $f(sol)$ é o valor da função objetivo da solução atual.

Se a temperatura inicial for muito alta, a busca será uma busca local aleatória. Caso contrário, se a temperatura inicial for muito baixa, a busca será mais ou menos um algoritmo de busca local aprimorado. Portanto, para sua utilização prática, a temperatura inicial não deve ser muito alta nem muito baixa. Geralmente esse valor é definido por uma experimentação preliminar para encontrar um valor satisfatório (KIRKPATRICK; JR; VECCHI, 1983; ČERNỲ, 1985).

Com o passar das iterações, ocorre o resfriamento que determina a forma como a temperatura vai variar com o tempo. É importante notar que a temperatura inicia com um valor acima de zero e deverá, com o número suficiente de iterações, convergir para zero. As três principais formas de se realizar o resfriamento são:

- **Linearmente**: O valor da temperatura é decrementado linearmente como na Equação 2.26, onde i é o valor da iteração atual e β é uma constante pré-definida.
- **Geometricamente**: O valor da temperatura é atualizado usando a Equação 2.27, onde α está entre 0 e 1. Geralmente entre 0.5 e 0.99.
- **Logarítmica**: A temperatura diminui de acordo com a Equação 2.28, onde i é o valor da iteração atual.

$$T_i = T_0 - i \times \beta \tag{2.26}$$

$$T_i = \alpha \times T_0 \tag{2.27}$$

$$T_i = \frac{T_0}{log(i)} \tag{2.28}$$

O Algoritmo 2 apresenta o *template* do *Simulated Anneling* (SA). Na linha 1, são passados como parâmetros de entrada a solução inicial e temperatura inicial. Entre as linhas 5 e 15 ocorre um laço de repetição que busca o estado de equilíbrio do sistema (geralmente expresso em número de iterações). Na linha 7 ocorre a geração

da vizinhança e seleção da solução candidata. Na linha 9 ocorre a avaliação para verificar a diferença entre a função objetivo da solução atual e da solução candidata. Se a diferença for positiva, ocorre substituição (linha 11); caso contrário, escolhe-se um número entre 0 e 1 (linha 13) e calcula-se a probabilidade de aceitação da solução como na Equação 2.25 (linha 14). Se p for maior que r, ocorre a troca da solução atual pela solução aceita. Na linha 20 ocorre a atualização da temperatura. Entre as linhas 4 e 21 está o laço de repetição que continua até um critério de parada, geralmente quando a temperatura chega no valor mínimo (ou zero). Na linha 22, ocorre o retorno da melhor solução encontrada.

Algoritmo 2 Template de *Simulated Anneling* (SA)

1: **input:** s_0 e t_{max}
2: $s = s_0$/*Solução inicial*/
3: $T = T_{max}$/*Temperatura inicial*/
4: **repeat**
5: **repeat**
6: /*Aplicar operadores e gerar vizinhança*/
7: $s' = gerarVizinhancaESelecionar(s')$
8: /*Calcula a diferença entre função objetivo da solução atual e candidata*/
9: $\Delta E = f(s') - f(s)$
10: **if** $\Delta E \leq 0$ **then**
11: $s = s'$
12: **else**
13: $r = sortearNumeroEntre(0, 1)$
14: $p = e^{\frac{-\Delta E}{T}}$
15: **if** $r < p$ **then**
16: $s = s'$
17: **end if**
18: **end if**
19: **until** Condição de equilíbrio /*Geralmente número de iterações*/
20: $T = atualizaTemperatura(T)$
21: **until** Critério de parada não for satisfeito/*$T < T_{min}$*/
22: **Output:** Melhor solução encontrada

Fonte: Adaptado de Talbi (2009).

2.2.4 *Adaptive Large Neighborhood Search* (ALNS)

O *Adaptive Large Neighborhood Search* (ALNS) é uma *Metaheurística Baseada em Solução Única* (MBSU) e foi proposto em (ROPKE; PISINGER, 2006) e estende o *Large Neighborhood Search* (LNS) (SHAW, 1998). Assim, é importante falar sobre o LNS antes de definir ALNS.

No LNS a vizinhança é criada por operadores de destruição e reparo. Um operador de destruição, destrói parte da solução atual, por exemplo, removendo visi-

tas. Enquanto um operador de reparo reconstrói a solução destruída inserindo visitas ainda não feitas. O operador de destruição normalmente contém um elemento de estocasticidade para garantir que diferentes partes da solução sejam destruídas em cada iteração. A vizinhança de uma solução é então definida como o conjunto de soluções que podem ser alcançadas aplicando primeiro o operador de destruição e depois o operador de reparo (GENDREAU; POTVIN et al., 2010). Como o operador de destruição pode destruir a solução de várias formas diferentes, a vizinhança contém uma grande quantidade de soluções, o que explica o nome da heurística.

A ideia principal por trás do LNS é que uma grande vizinhança permite que a heurística navegue facilmente no espaço da busca. Isso se opõe a uma pequena vizinhança que pode dificultar muito a navegação no espaço de busca. O método de destruição é uma parte importante do LNS. A escolha mais importante ao implementar o método é o grau de destruição: se apenas uma pequena parte da solução for destruída, a heurística pode ter problemas para explorar o espaço de busca, pois o efeito de uma grande vizinhança é perdido. Se uma parte muito grande da solução for destruída, o LNS passará muito tempo reparando soluções sem necessariamente conseguir encontrar novas soluções melhores. Isso pode consumir muito tempo ou resultar em soluções de baixa qualidade, dependendo de como a solução parcial é reparada (GENDREAU; POTVIN et al., 2010).

É importante escolher o operador de reparo pelo seu impacto na qualidade da solução e tempo de execução. Uma operação de reparo ideal será mais lenta com um operador mais complexo, mas pode levar a soluções de alta qualidade em poucas iterações. No entanto, do ponto de vista da diversificação, um operador de reparo ótimo pode não ser atraente: apenas um conjunto restrito de soluções podem ser produzidas e pode ser difícil deixar ótimos locais a menos que uma grande parte da solução seja destruída em cada iteração (GENDREAU; POTVIN et al., 2010).

Por sua vez, o ALNS estende a heurística do LNS, permitindo que vários operadores de destruição e reparo sejam usados ao mesmo tempo. A cada método de destruição/reparo é atribuído a um peso, que controla a frequência com que o método específico é utilizado durante a busca. Os pesos são ajustados dinamicamente à medida que a busca progride para que a heurística se adapte à instância em questão e ao estado da busca. Operadores que têm valores altos em seus pesos indicam que obtiveram mais sucesso em melhorar a solução (GENDREAU; POTVIN et al., 2010).

O Algoritmo 3 apresenta o *template* do ALNS. Nas linhas 1, 2 e 3, são passados ao algoritmo a solução inicial, operadores de destruição e operadores de reparo, respectivamente. Nas linhas 4 e 5, os valores dos pesos dos operadores são iniciados. No ALNS, todos os pesos iniciam com valores idênticos (com valor 1, por exemplo). Nas linhas 9 e 10, ocorre a seleção dos operadores. A seleção ocorre por um método

Algoritmo 3 Template de *Adaptive Large Neighborhood Search* (ALNS)

1: **input:** s_0 /*Solução inicial*/
2: **input:** O^- /*Operadores de destruição*/
3: **input:** O^+ /*Operadores de reparo*/
4: $p^- = \{1, ..., 1\}$/*Peso dos operadores de destruição*/
5: $p^+ = \{1, ..., 1\}$/*Peso dos operadores de reparo*/
6: $s_{melhor} = s_{atual} = s_0$
7: **repeat**
8: /*selecionar operador de destruição e reparo*/
9: $d = seleciona(O^-)$
10: $r = seleciona(O^+)$
11: /*Aplica operadores*/
12: $s_{novo} = r(d(s_{atual}))$
13: **if** $accept(s_{novo}, s_{atual})$ **then**
14: $s_{novo} = s_{atual}$
15: **end if**
16: **if** $best(s_{melhor}, s_{novo})$ **then**
17: $s_{melhor} = s_{novo}$
18: **end if**
19: $atualize(p^-, p^+)$
20: **until** Critério de parada não for satisfeito
21: **Output:** s_{melhor}

Fonte: Adaptado de Gendreau, Potvin et al. (2010).

que sorteia um operador aleatoriamente, porém favorece a escolha dos operadores com maiores pesos, mas não impede que operadores com pesos menores sejam selecionados. A probabilidade P_i de um operador i ser selecionado é dada pela Equação 2.29, onde n é a quantidade de operadores e s_i é sua pontuação. Em seguida, na linha 13 apresenta a função que trata da aceitação ou não da solução gerada. A implementação mais simples do ALNS diz que sempre que houver melhora, a solução deve ser aceita. Na linha 16 ocorre a atualização da melhor solução, se for o caso (ROPKE; PISINGER, 2006).

$$P_i = \frac{s_i}{\sum_{j=1}^{n} s_j} \tag{2.29}$$

Os pesos para os operadores são ajustados dinamicamente, com base no desempenho registrado de cada método de destruição e reparo. Isso ocorre na linha 19: quando uma iteração da heurística ALNS é concluída, uma pontuação para o método de destruição e reparo usado na iteração é atribuída.

O novo peso do operador é dado pela Equação 2.30. O parâmetro λ é o parâmetro de decaimento que controla a sensibilidade dos pesos às mudanças no desem-

penho dos métodos de destruição e reparo. A variável p_i é o peso atual do operador, o qual será atualizado. O parâmetro Ψ indica qual dos pesos é utilizado dependendo da situação do algoritmo. Observe que os pesos dos operadores que não são usados na iteração atual permanecem inalterados. O objetivo do ajuste de peso adaptativo é selecionar operadores que funcionem bem para a instância que está sendo resolvida. É incentivado o uso de heurísticas que levam a busca adiante. Por outro lado, não ocorre o encorajamento de heurísticas que levam a muitas soluções rejeitadas, pois uma iteração que resulta em uma solução rejeitada é uma iteração desperdiçada (ROPKE; PISINGER, 2006).

$$p_i = \lambda p_i + (1 - \lambda)\Psi; \lambda \in [0, 1] \tag{2.30}$$

2.3 OPERADORES

Esta seção apresenta os operadores utilizados pelo algoritmo ALNS. O ALNS utiliza dois tipos principais de operadores: destruição e reparo e são apresentados nas Seções 2.3.1 e 2.3.2, respectivamente.

2.3.1 Operadores de destruição

Os operadores de destruição são usados pelo ALNS para remover visitas das rotas de uma solução. O princípio básico é buscar remoções que possam abrir espaço para inserções mais eficientes no futuro. A primeira questão que deve ser debatida é qual o número de remoções que deve ser feita a cada iteração. Pode-se utilizar um número fixo como 15% ou 50% do tamanho da solução, por exemplo. Mas uma estratégia interessante é criar uma forma de gerar quantidade diferente de remoções durante a execução do algoritmo. Uma forma de se alcançar isso é demonstrada em Ropke e Pisinger (2006), por meio da Expressão 2.31, onde y representa o número de remoções, R representa o número de requisições (que é representado por uma visita), N é o número de remoções máximo do problema (tamanho da solução) e ξ é um parâmetro de controle que mantém o número de remoções dentro de um intervalo que faça sentido para o cenário, este trabalho foi utilizado o valor de 40 %.

$$4 \leq y \leq min(N, \xi * R) \tag{2.31}$$

Os principais operadores encontrados da literatura e que são apresentados nas próximas subseções são: remoção aleatória, remoção de pior custo e remoção relacional.

2.3.1.1 Remoção aleatória

A remoção aleatória computa um número y, que representa quantas requisições devem ser removidas da solução S. Para cada uma das y remoções, será feita a seleção aleatória de uma requisição e sua remoção da solução. A ideia dessa heurística é aumentar a exploração do algoritmo facilitando a fuga de ótimos locais, já que a mesma não considera nenhum tipo de critério para selecionar qual requisição deve ser removida (PISINGER; ROPKE, 2007; NACCACHE; CÔTÉ; COELHO, 2018).

No Algoritmo 4 é possível observar o *template* do algoritmo de remoção aleatório. O algoritmo recebe uma solução S e número de remoções y como argumentos de entrada. Ele consiste em um laço de repetição que vai remover as requisições. Na linha 4, ocorre a seleção aleatória de uma requisição e na linha 5, ela é removida. Ao final o algoritmo retorna como saída a solução S sem y requisições.

Algoritmo 4 Template do algoritmo de remoção aleatória

```
1: Input:Solução S
2: Input:Número de remoções y
3: while y > 0 do
4:     r = selecionaVisita( S )
5:     S = S\{r}
6:     y = y-1
7: end while
8: Output: S
```

2.3.1.2 Remoção de pior custo

O operador de pior custo [2] é uma heurística que busca remover requisições que geram mais diferença na função objetivo da solução. A diferença da função objetivo entre a solução atual e sua versão com a remoção da visita é chamada de ganho. A cada iteração, a seleção da requisição que deve ser removida leva em conta o ganho gerado e busca-se escolher a requisição que gera maior ganho. A ideia desse operador é remover das rotas as requisições que estejam desviando muito o veículo do trajeto (ou seja, requisições que estão alocadas no lugar errado), e assim realocar em outras rotas (ou posições) para reduzir o custo total do roteamento (PISINGER; ROPKE, 2007; NACCACHE; CÔTÉ; COELHO, 2018).

O *template* da remoção de pior custo é apresentado no Algoritmo 5. O algoritmo recebe como entrada uma solução S, número de requisições a serem retiradas y e um fator aleatório p, que é um número aleatório real positivo, que indica a probabilidade de selecionar uma visita aleatoriamente, assim adicionando ruído. O laço de

[2] O custo do nome do operador não tem relação direta com o custo da função objetivo.

repetição entre as linhas 4 e 12 executa enquanto ainda houver requisições para serem removidas. Na linha 5, a lista G é iniciada sem nenhum elemento. Entre as linhas 6 a 8, executa um laço de repetição que calcula o ganho que a visita r gera ao ser removida da solução s. O ganho é a diferença entre a função objetivo da solução e da solução com a remoção feita. Na linha 9, a lista G é ordenada de forma que o primeiro elemento é aquele que gerou mais ganho. Na linha 10, ocorre a seleção da visita, se p tem valor igual a zero, significa que o primeiro elemento da lista vai ser selecionado, caso contrário, é adicionado ruído. A adição de ruído faz com que a requisições tem tenha alterações nos valores de ganho, assim permitindo que outras requisições sejam selecionadas para remoção. Na linha 11 ocorre a remoção da requisição.

Algoritmo 5 Template do algoritmo de remoção de pior custo

```
1:  Input:Solução S
2:  Input:Número de remoções y
3:  Input:Fator aleatório p ∈ ℝ+
4:  while y > 0 do
5:      G = {}
6:      for r ∈ S do
7:          G = G ∪ calculeGanho(r)
8:      end for
9:      G = ordenar( G )
10:     r = selecionaVisita( G, S, p )
11:     S = S\{r}
12:     y = y-1
13: end while
14: Output: S
```

2.3.1.3 Remoção relacional

O operador de remoção relacional é uma heurística que foca em remover visitas com o maior grau de similaridade entre si. A ideia é que removendo tais visitas seria possível fazer trocas entre as alocações e encontrar melhores soluções. Ou seja, essa heurística tenta resolver o problema onde a remoção de visitas muito diferentes não resulta em nenhuma troca da ordem das visitas, visto que elas só poderiam ser reinseridas nas mesmas posições. A verificação de similaridade pode ser feita combinando diferentes métricas do problema e é feita comparando duas visitas. Na Equação 2.32 é apresentada uma representação genérica do cálculo do valor de similaridade, onde p', p'' e p''' são pesos usados em cada termo e estes valores podem variar de problema para problema (PISINGER; ROPKE, 2007). A $\Delta distancia$ indica a diferença de distância entre as duas visitas, $\Delta demanda$ indica a diferença de demanda e $\Delta recompensa$ indica a diferença de recompensa.

$$similaridade(i,j) = p'(\Delta distancia) + p''(\Delta demanda) + p'''(\Delta recompensa) \quad (2.32)$$

O Algoritmo 6 apresenta o *template* do algoritmo de remoção relacional. O algoritmo recebe como entrada uma solução S, número de remoções y e fator aleatório p responsável pela adição de ruído. Na linha 4, ocorre a remoção de uma requisição da solução de forma aleatória. Na linha 5, a lista D é criada e recebe a requisição removida. Essa lista vai conter todas as requisições removidas da solução. Entre as linhas 7 e 18, é executado o laço de repetição que vai realizar todas as remoções. Na linha 8, é selecionada uma nova requisição e na linha 9 vai ser iniciada a lista H que vai abrigar todos os valores de similaridade. Na linha 11, ocorre o cálculo de similaridade entre a requisição r e a primeira requisição removida r_0. Na linha 13 é feita a ordenação da lista H, onde o primeiro elemento vai ser a requisição com mais similaridade em relação a requisição r_0. Na linha 14 ocorre a seleção da requisição, se p tem valor igual a zero, significa que o primeiro elemento da lista vai ser selecionado, caso contrário, é adicionado ruído na seleção. A adição de ruído é sortear um número aleatório para selecionar o elemento da lista, assim permitindo que outras requisições sejam escolhidas. Na linha 15 ocorre a remoção da requisição e na linha 16, a requisição selecionada é adicionada à lista D.

Algoritmo 6 Template do algoritmo de remoção relacional

```
1:  Input:Solução S
2:  Input:Número de remoções y
3:  Input:Fator aleatório p ∈ ℝ+
4:  r = removaVisitaAleatoriamente( S )
5:  D = {r}
6:  y = y-1
7:  while y > 0 do
8:      r = selecioneAleatoriamente( D )
9:      H = {}
10:     for r ∈ S do
11:         H = H ∪ calculeSimilaridade( r, r0 )
12:     end for
13:     H = ordenar( H )
14:     r = selecionaVisita( H, S, p )
15:     S = S\{r}
16:     D = D ∪ {r}
17:     y = y-1
18: end while
19: Output: S
```

2.3.1.4 Remoção de rota

A heurística de remoção de rotas tem como objetivo a exclusão completa de uma rota da solução. Com isso, busca-se a possibilidade de redistribuir seus nós de maneira mais eficiente em outras rotas já existentes ou reconstruí-la de modo a maximizar a redução do custo da rota. O Algoritmo 7 descreve o seu funcionamento. O algoritmo recebe uma solução S e uma proporção de remoções p. Por exemplo, se p for 0.4 e S tiver 10 rotas, o operador vai remover 4 rotas. O cálculo é feito na linha 3. Na linha 4, o operador vai escolher aleatoriamente rotas para serem removidas. No laço de repetição da linha 5, o operador vai remover as rotas e no final o algoritmo vai retornar a solução.

Algoritmo 7 Template do algoritmo de remoção de rotas

```
1: Input:Solução S
2: Input: Proporção de remoções p
3: numeroRemocoes = tamanho( S ) * p
4: R = selecioneRotasAleatoriamente( numeroRemocoes )
5: for r ∈ R do
6:     S = removeRota( r, S )
7: end for
8: Output: S
```

2.3.2 Operadores de reparo

Os operadores de reparo são executados pelo ALNS logo após os operadores de destruição. Assim, o próximo passo é tentar inserir as requisições anteriormente removidas. Essa etapa é realizada pelos operadores de reparo, que são heurísticas que buscam melhorar a solução fazendo inserções de visitas mais apropriadas. Os dois operadores citados neste trabalho são a inserção gulosa e inserção com critério de arrependimento.

2.3.2.1 Inserção gulosa

O operador de inserção gulosa é uma heurística de inserção simples. Ela utiliza o critério de inserir requisições com menor piora da função objetivo na solução. Para definir qual visita deve ser inserida e em qual posição, é executado um algoritmo que busca a posição (rota e índice) que gera a menor piora para cada visita, sendo que a visita que gerar a menor piora é selecionada. Ao término do processo, a visita escolhida é adicionada na solução. Esse operador itera até que todas as visitas sejam inseridas, uma a uma, ou não seja possível mais inserir visitas devido às restrições (PISINGER; ROPKE, 2007).

O Algoritmo 8 apresenta o *template* da inserção gulosa. O algoritmo recebe como entrada a solução S e as visitas R que foram removidas pelos operadores de destruição. Entre as linhas 3 e 12, existe o laço que insere as visitas enquanto houver visitas em R e for possível inserir sem ofender as restrições. Na linha 4 é criada a lista dos custos, sob a qual é realizado o cálculo do custo para cada visita em R. Na linha 8 ocorre a ordenação dos custos e na linha 9 seleciona-se a primeira posição da lista de custos. A primeira posição é o elemento que gera o menor acréscimo ao ser inserida. Na linha 10 realiza-se a inserção na solução e na linha 11 é feita a remoção da lista R, ou seja, remove da lista de visitas não utilizadas.

Algoritmo 8 Template do algoritmo de inserção gulosa

```
1:  Input:Solução S
2:  Input:Visitas não utilizadas R
3:  while |R| > 0 e for possível inserir visita do
4:      C = {}
5:      for r ∈ R  do
6:          C = C ∪ calculePiora( r, S )
7:      end for
8:      C = ordenar( C )
9:      r = c[0] /*Melhor visita*/
10:     S = S ∪ {r}
11:     R = R \ {r}
12: end while
13: Output: S
```

2.3.2.2 Inserção com critério de arrependimento

Esse operador visa complementar um problema relacionado ao operador de inserção guloso. O fato de o operador guloso sempre selecionar a melhor inserção do momento faz com que ele ignore o impacto futuro daquela inserção nas demais inserções. O operador de inserção com critério de arrependimento resolve esse problema. Basicamente, ele considera a informação futura para cada visita, onde é analisado o custo da melhor inserção e considerado o quão pior as demais opções de inserção serão se a melhor não for feita. Formalmente pode-se definir $\Delta_{i,n}$ como sendo o n-ésimo menor custo de inserir a requisição i. O valor de arrependimento c_i é igual a $\Delta_{i,2} - \Delta_{i,1}$, ou seja, o valor de arrependimento é representado pela diferença de custo entre a segunda melhor e a melhor inserção de visitas; e em cada iteração é selecionada para inserção a visita com maior valor de arrependimento. O operador de arrependimento foca em selecionar a visita que se não for selecionada agora, pode gerar um maior arrependimento no futuro(PISINGER; ROPKE, 2007; PSARAFTIS, 1995).

O operador foi definido em termos da diferença de custo entre a primeira e a segunda inserção de uma requisição. Porém, ele pode ser estendido para suportar um intervalo maior de comparações. Nesse caso, **arrependimento-k** é o operador onde k requisições são consideradas no cálculo do valor de arrependimento. A forma de calcular o valor de arrependimento é dado pela Equação 2.33.

$$c_i = max\{\sum_{j=1}^{k}(\Delta_{i,j} - \Delta_{i,1}\} \tag{2.33}$$

Algoritmo 9 apresenta o *template* da inserção com critério de arrependimento. Ele recebe como entradas uma solução S e as visitas não utilizadas R removidas pelos operadores de destruição. Entre as linhas 3 e 16, ocorre a inserção de visitas enquanto ainda houver requisições não atendidas e não ofender as restrições. Na linha 4 é criada a lista que armazena os custos, que é calculado para cada visita (linha 5) em cada rota (linha 6) da solução. O cálculo é executado na linha 7. Na linha 10 é feita a ordenação e na linha 13 é selecionada a visita que gera o maior valor de arrependimento para na linha 14 ser adicionada à solução. A linha 15 atualiza a lista de visitas a serem testadas R, removendo a requisição recentemente inserida na solução S

Algoritmo 9 Template do algoritmo de inserção com critério de arrependimento

```
1:  Input:Solução S
2:  Input:Visitas não utilizadas R
3:  while |R| > 0 e for possível inserir visitas do
4:     C = {}
5:     for r ∈ R  do
6:        for k ∈ S /*Para cada rota da solução*/ do
7:           C = C ∪ calculeMelhorCusto( r, k, S )
8:        end for
9:     end for
10:    C = ordenar( C )
11:    /*Seleciona a visita com maior critério de arrependimento*/
12:    /*c_i = Δ_{i,2} − Δ_{i,1} ou c_i = max{Σ_{j=1}^{k}(Δ_{i,j} − Δ_{i,1}} * /
13:    r = selecioneVisita(C)
14:    S = S ∪ {r}
15:    R = R \ {r}
16: end while
17: Output: S
```

2.4 REVISÃO DE TRABALHOS RELACIONADOS

Nesta seção são apresentados os trabalhos relacionados ao tema de pesquisa desta dissertação. Cada trabalho apresentado aqui possui algum aspecto relacionado

ao VRP, TOP ou VRPP. A sua revisão visa apresentar o estado da arte na solução de problemas correlatos ao problema abordado neste trabalho, a fim de destacar suas contribuições, que poderão ser incorporadas no presente trabalho, ou mesmo apontar oportunidades de melhorias que serão abordadas.

Para se selecionar os trabalhos relacionados, a ferramenta de busca utilizada foi o Google Acadêmico (https://scholar.google.com/). A busca teve como objetivo encontrar os trabalhos que tinham recompensa em sua função objetivo ou tinham alguma restrição com recompensa, porém não se restringindo a eles. Nesse sentido, buscou-se por trabalhos escritos em inglês, utilizando-se os termos:

- *traveling salesman problem*.
- *traveling salesman problem with profits*.
- *traveling salesman problem with time windows*.
- *vehicle routing problem*.
- *vehicle routing problem with profits*.
- *vehicle routing problem with profits and time windows*.
- *team orienteering problem*.
- *team orienteerign problem with time windows*.

No trabalho de Lysgaard, Letchford e Eglese (2004) é apresentada a variante CVRP. Aqui é feita a adição da restrição da capacidade da frota ao problema original VRP. Nesse caso, a função objetivo minimiza o custo. O trabalho implementou um algoritmo baseado no método *Branch-and-cut*.

No trabalho de Aksen e Aras (2006) é apresentada a variante *VRP with Profits and Time Deadlines* (VRPP-TD). Aqui é feita a alteração da função objetivo para maximização da diferença da recompensa coletada e distância percorrida. As restrições incluem a capacidade e tempo máximo para visita. O trabalho implementou a metaheurística *Simulated Anneling* (SA) e um *Iterative Marginal Profit Analysis Method* (iMPA).

O trabalho de El-Hajj et al. (2020) é apresentada a variante VRP com recompensa e multiperíodos (mVRPP). O multiperiodos são janelas de serviço dentro da mesma rota, por exemplo, dias de semana em um planejamento semanal. A função objetivo maximiza a recompensa [3]. As restrições envolvem a modelagem de multi-

[3] Isso mesmo, é uma variante do VRP, mas que maximiza a recompensa como o TOP

períodos e tempo máximo de rotas. O trabalho implementou um algoritmo baseado no *Particle Swarm Optimization* (PSO).

O trabalho de Stavropoulou, Repoussis e Tarantilis (2019) apresenta a variante do VRP com recompensa e algumas restrições (conVRPP). As restrições envolvem capacidade, tempo máximo de rotas e número de visitas diferentes para clientes. A função objetivo busca maximizar a diferença entre a recompensa e a soma do custo e tempo de serviço. O trabalho utilizou o método baseado em *Adaptive Tabu Search* (ATS).

No trabalho de Orlis et al. (2020) é apresentado um problema que surge na logística de distribuição de dinheiro, o *Capacitated Routing Problem with Profits and Service Level Requirements* (CRPPSLR). O CRPPSLR estende a classe de Problemas de Roteamento com Lucros considerando clientes solicitando entregas em seus (possivelmente múltiplos) pontos de atendimento. Além disso, cada cliente impõe um requisito de nível de serviço especificando um limite mínimo aceitável na fração de seus pontos de serviço sendo entregues. Uma penalidade financeira específica do cliente é incorrida pelo provedor de serviços de logística quando esse requisito não é atendido. O CRPPSLR consiste em encontrar rotas de veículos maximizando a diferença entre as receitas arrecadadas e os custos de transporte e multa incorridos de forma que as restrições de capacidade dos veículos e duração da rota sejam atendidas. Uma frota de veículos homogênea está disponível para atender os clientes. Um algoritmo *branch-and-cut* é utilizado. A função objetivo é dada pela maximização da recompensa coletada subtraída pelo custo de deslocamento e da penalidade dos atendimentos não ocorridos.

No trabalho de Pedro, Saldanha e Camargo (2013) é apresentado o Traveling Salesman Problem (TSP) com coleta de recompensa (PCTSP). O TSP busca criar uma rota apenas, ao contrário do VRP, que busca criar múltiplas rotas para atender as demandas. A função objetivo busca maximizar a recompensa. As restrições adicionam uma recompensa mínima. O trabalho implementou um algoritmo baseado no *Particle Swarm Optimization* (PSO).

No trabalho de Beraldi et al. (2019) é apresentada uma variante do Traveling Salesman Problem (TSP) clássico onde um reparador deve visitar todos os nós de um grafo exatamente uma vez, minimizando os tempos de chegada aos clientes. A variante é a *Traveling Repairman Problem with Profits* (TRPP). O trabalho propôs um algoritmo baseado na heurística *Beam Search* (BS). A função objetivo é definida pela maximização entre o lucro total e a soma dos tempos.

O trabalho de Dang, Guibadj e Moukrim (2013) trabalhou com o TOP. A função objetivo busca maximizar a recompensa e existe a restrição de tempo máximo da rota.

O trabalho implementou um algoritmo baseado no *Particle Swarm Optimization* (PSO).

O trabalho de Ben-Said, El-Hajj e Moukrim (2019) trabalha com a variante CTOP, que adiciona a restrição da capacidade do veículo ao TOP. A função objetivo busca maximizar a recompensa. As restrições incluem o tempo máximo da rota e capacidade. O trabalho utilizou a abordagem baseada na heurística *Variable Space Search*.

De forma geral, os trabalhos citados demonstram que diversas abordagens já foram propostas como solução, contemplando variadas características do VRP, tais como as como janela de tempo, capacidade, janela de atendimento, etc. Além disso, várias funções objetivo foram propostas, como minimização do custo, maximização de recompensa, combinação entre custo e recompensa ou ainda combinação desses dois fatores e outros. Para resumir a comparação em relação a literatura, a Tabela 1 apresenta uma comparação entre os trabalhos analisados nesta seção e este trabalho.

O primeiro ponto a ser levantado é sobre a diversidade de métodos utilizados para a resolução do problema. Como pode ser observado, todos utilizam metaheurísticas, heurísticas ou métodos híbridos. Isso se justifica pela dificuldade que abordagens de métodos exatos enfrentam na solução desse tipo de problema. Dentro das metaheurísticas e heurísticas utilizadas, as variações do método MBSU, tais como: VSS, SA e BS foram as mais presentes, seguido pelo PSO e pelo *branch-and-cut*. Portanto, pode-se notar que métodos baseados em solução única são bastante utilizados. Entretanto, nenhum utilizou a metaheurística ALNS. O ALNS apresenta um ponto que pode ser um diferencial, que é os valores do peso que se alteram durante o algoritmo, permitindo a adaptação para cada instância do problema.

Ainda, pode-se observar que nenhum trabalho relacionado abordou a solução da variante do VRPPTW com as restrições de Capacidade, Janela de Tempo e Tempo Máximo. Portanto, este trabalho apresenta como contribuição o avanço no desenvolvimento de solução para uma variante do VRP ainda não explorada. Por fim, conforme revisado, nenhum trabalho apresenta um *dataset* de *benchmark* para o VRPPTW público e baseado em *datasets* padrão. Portanto, esse trabalho também apresenta como contribuição a proposição de um *dataset* a partir de extensão do *dataset* de Solomon (1987), com a adição da recompensa associada a cada requisição, para servir de *benchmark* para a realização de experimentos com soluções para VRPPTW.

Tabela 1 – Trabalhos relacionados

Artigo	Tipo	Restrições	Método de solução	Função Objetivo	Instâncias
(LYSGAARD; LETCHFORD; EGLESE, 2004)	CVRP	Capacidade	Algoritmo Branch-and-cut	Minimiza o custo	Instâncias do CVRP.
(AKSEN; ARAS, 2006)	VRPP-TD	Capacidade, Deadline para vista	SA e iterative marginal profit analysis method (iMPA)	minimiza a diferença entre recompensa e distância	Geradas, não disponibilizadas
(EL-HAJJ et al., 2020)	mVRPP	Multi períodos e Tempo máximo	Algoritmo Particle Swarm Optimization	Maximiza Recompensa	Instâncias do TOP e CVRP
(STAVROPOULOU; REPOUSSIS; TARANTILIS, 2019)	ConVRPP	Capacidade, Tempo máximo, Número de visitas diferentes para clientes	Adaptive Tabu Search (ATS)	Maximize a diferença entre a recompensa e a soma do custo e tempo de serviço	Criou as próprias instâncias, não disponibilizou.
(ORLIS et al., 2020)	CRPPSLR	Capacidade e duração máxima da rota	Algoritmo *branch-and-cut*	Maximiza a recompensa coleta subtraída pelo custo de deslocamento e da penalidade dos atendimentos não ocorridos	5 Instâncias da literatura CVRP disponíveis.
(ORLIS et al., 2020)	CRPPSLR	Capacidade e duração máxima da rota	Algoritmo *branch-and-cut*	Maximiza a recompensa coleta subtraída pelo custo de deslocamento e da penalidade dos atendimentos não ocorridos	5 Instâncias da literatura CVRP disponíveis.
(PEDRO; SALDANHA; CAMARGO, 2013)	PCTSP	Recompensa minima	Abordagem com Tabu Search	Minimize a soma do custo e recompensa não coletada	Instâncias próprias, porém partes delas não estão mais disponíveis
(BERALDI et al., 2019)	TRP	Não tem	Algoritmo baseado na heurística beam search (BS)	Maximiza a diferença entre o lucro total e a soma dos tempos	Instâncias do Traveling Repairman Problem, não estão disponíveis.
(DANG; GUIBADJ; MOUKRIM, 2013)	TOP	Tempo máximo	Algoritmo Particle Swarm Optimization	Maximiza recompensa	Instâncias do TOP
(BEN-SAID; EL-HAJJ; MOUKRIM, 2019)	CTOP	Tempo máximo da rota, Capacidade	Heurística variable space search	Maximiza recompensa	Instâncias do CTOP
Este trabalho	VRPPTW	Capacidade, Janela de tempo, tempo máximo	ALNS-SA	Minimize a soma do custo e recompensa não coletada	Instâncias adaptadas do VRPTW

3 SOLUÇÃO PROPOSTA

Neste capitulo é apresentado a proposta de solução. Assim, é proposto o desenvolvimento de um algoritmo baseado na metaheurística ALNS. Antes de entrar nos detalhes da solução, inicialmente será dada ênfase aos aspectos que caracterizam o problema e que devem ser considerados para o desenvolvimento.

3.1 DEFINIÇÃO DO PROBLEMA

O primeiro ponto que merece atenção na modelagem do problema é sua função objetivo. Como visto na Seção 2.1.3, o VRPP possui como função objetivo a maximização da diferença entre a recompensa coletada e o custo das rotas (Equação 2.17). Importante notar que a estratégia de modelagem do problema como sendo a maximização da diferença entre dois somatórios pode levar a valores negativos para a função objetivo. Este é um ponto que, dependendo da metaheurística utilizada para solucionar o problema, pode transformar-se em uma característica a ser tratada no projeto do algoritmo. Para evitar que o valor da função objetivo fique negativo, a abordagem proposta neste trabalho faz uma adaptação na forma como a função é modelada. A função objetivo poderia ser maximizar a diferença entre recompensa coletada e o custo total das rotas. Porém, maximizar a recompensa coletada é equivalente a minimizar a recompensa não coletada. Assim, de forma a compatibilizar a função objetivo como uma minimização, além de evitar uma diferença de fatores na função objetivo, ela pode ser a minimização da soma do custo total das rotas e a recompensa total não coletada. Assim, a função objetivo proposta neste trabalho pode ser definida conforme apresentado na Equação 3.1, que foi adaptada de (ARCHETTI; SPERANZA; VIGO, 2014). Com essa modelagem, a função objetivo não assume valores negativos. Com relação às restrições, o problema deste trabalho contempla as seguintes: i) capacidade de cada veículo; ii) janela de atendimento do cliente; e iii) tempo máximo da rota.

Assim, dado o grafo $G = (V, A)$, onde $V = \{1, ..., n\}$ é o conjunto de vértices e $A = \{(i, j) : i, j \in V, i \neq j\}$ é o conjunto de arestas, o vértice 1 representa o depósito de onde os veículos devem iniciar e finalizar sua rota. A frota F de tamanho m é composta com veículos idênticos e m é conhecido com antecedência. Um custo c_{ij} é associado a aresta entre os clientes i e j. O custo pode ser definido como custo de viagem ou distância. A janela de tempo do cliente i representa o intervalo de tempo que o veículo dispõe para realizar a entrega e é representada por $[s_i, e_i]$, onde s_i é o inicio da janela de tempo e e_i é o término da janela de tempo. A variável v_{if} representa o momento

que o veículo f iniciou a visita e t_{ij} o tempo que se leva para percorrer a aresta entre i e j. A variável r_i indica a recompensa coletada ao visitar o cliente i e d_i representa a demanda do cliente i (volume/peso do pacote). A variável T_{max} representa o tempo máximo que cada rota não pode ultrapassar e Q a demanda máxima da frota. As variáveis binárias x_{ij} e y_i indicam se a aresta entre os clientes i e j foi utilizada e se o cliente i foi visitado, respectivamente.

A Expressão 3.1 representa a função objetivo, onde busca-se minimizar a soma do custo das rotas e a soma das recompensas não coletadas. A Restrição 3.2 garante que a soma de todas as demandas carregadas pelos veículos não ultrapasse o limite máximo da frota. A Restrição 3.3 garante que a visita do veículo k inicie entre o início e final da janela de tempo do cliente i. A Restrição 3.4 estabelece a relação entre o horário de saída do veículo de um cliente e seu sucessor imediato, onde a soma do início do atendimento anterior e o tempo de deslocamento até o próximo vértice deve ser menor ou igual ao tempo de início do atendimento do próximo vértice. A Restrição 3.5 garante que o tempo da rota não ultrapasse um limite máximo. A Restrição 3.6 garante que todos os veículos voltem ao depósito. A Restrição 3.7 garante que os veículos que entram por uma aresta em um vértice, saem por outra aresta. As Restrições 3.8 e 3.9 indicam que as variáveis x e y são binárias.

$$\text{Minimize} \sum_{i \in E} \sum_{j \in E} c_{ij} x_{ij} + \sum_{i \in V} r_i (1 - y_i) \tag{3.1}$$

$$\text{Sujeito:} \sum_{i \in V} d_i y_i < Q \tag{3.2}$$

$$s_i \leq v_{if} \leq e_i; i \in V, f \in F \tag{3.3}$$

$$x_{fij}(v_{if} + t_{ij} - v_{jf}) \leq 0; (i,j) \in V, f \in F \tag{3.4}$$

$$\sum_{i \in A} \sum_{j \in A} t_{ij} x_{ij} \leq T_{max} \tag{3.5}$$

$$\sum_{j=1}^{n} x_{1j} = 2m \tag{3.6}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2; k \in V \backslash \{1\} \tag{3.7}$$

$$x_{ij} \in 0, 1; (i,j) \in A \tag{3.8}$$

$$y_i \in 0, 1; i \in V \tag{3.9}$$

3.2 ALGORITMO PROPOSTO

Algoritmo 10 Template de *Adaptive Large Neighborhood Search* (ALNS) com caracteristicas do *Simulated Anneling* (SA)

Require: s_0 /*Solução inicial*/
Require: $iter$ /*Número de iterações*/
1: $f = 0.6$ /*Fator de reação*/
2: O^-={aleatorio, piorCusto, relacional} /*Operadores de destruição*/
3: O^+ = {guloso, arrependimento-2, arrependimento-3, arrependimento-4, arrependimento-n} /*Operadores de reparo*/
4: $p = 1$ /*Peso dos operadores*/
5: $s = 0$ /*Pontuação dos operadores*/
6: $c = 0$ /*Contagem dos operadores*/
7: $s_{melhor} = s_{atual} = s_0$
8: $temp = funObj(s_0)$ /*Inicia temperatura do SA*/
9: **for** $iter > 0$ **do**
10: /*seleciona operador de destruição e reparo*/
11: $d = seleciona(O^-)$
12: $r = seleciona(O^+)$
13: $s_{novo} = r(d(s_{atual}))$ /*Aplica operadores*/
14: $n = sortearNumeroEntre(0, 1)$
$p = e^{\frac{-(funObj(s_{atual}) - funObj(s_{novo}))}{T}}$
16: **if** $best(s_{melhor}, s_{novo})$ **then**
17: $s_{melhor} = s_{atual} = s_{novo}$
18: $sigma = 17$
19: **else if** $n < p$ **then**
20: **if** $best(s_{atual}, s_{novo})$ **then**
21: $s_{atual} = s_{novo}$
22: $sigma = 47$
23: **else**
24: $sigma = 3$
25: **end if**
26: Coloca a pontuação dos operadores d e r igual a $p_{anterior} + sigma$
27: Coloca a contagem dos operadores d e r igual a $c_{anterior} + 1$
28: **end if**
29: $temp = atualizaTemp()$ /* Equação 3.10 */
30: **if** $iter\%100 == 0$ **then**
31: **for** $o \in O^- \cup O^+$ **do**
32: Coloca o peso do operador o igual a $(1 - f) * p_o + (f * s_o)/max(c_o, 1)$
33: Coloca a pontuação do operador o igual a 0
34: Coloca a contagem do operador o igual a 0
35: **end for**
36: **end if**
37: $iter = iter - 1$
38: **end for**
39: **return** s_{best}

Esta seção visa apresentar em maiores detalhes o algoritmo utilizado neste trabalho. O Algoritmo 10 apresenta o pseudocódigo do ALNS implementado nesse trabalho. O algoritmo inicia recebendo uma solução inicial s_o e o número de iterações $iter$ que devem ser executadas. A forma de criação da solução inicial é discutida na Seção 3.3. Na linha 1, a variável f é inicializada com o valor 0.6. O valor de f foi encontrado a partir de pré experimentos. Essa variável é utilizada pelo SA para atualizar a temperatura do sistema. Na linha 2 é inicializada a lista com operadores de destruição, que são a remoção aleatória, remoção de pior custo e remoção relacional. Na linha 3 é inicializada a lista de operadores de reparo com inserção gulosa, arrependimento-2, arrependimento-3, arrependimento-4 e arrependimento-n. Todos esses operadores de construção têm uma variante com gerador de ruído e outro sem gerador de ruído. Todos os operadores foram apresentados na Seção 2.3 e mais detalhes da implementação são fornecidos na Seção 3.4.

Na linha 4 ocorre a inicialização das listas com os pesos de todos os operadores. Os pesos indicam quais operadores obtiveram mais sucesso ao tentar melhorar a solução. Quanto maior o peso de determinado operador, mais vezes ele conseguirá melhorar a solução e mais chances terá de ser selecionado novamente no futuro. Todos os operadores iniciam com valor igual a 1 para garantir que tenham a mesma probabilidade de serem selecionados pelo ALNS no início de sua execução. O valor dos pesos é atualizado após determinado número de iterações, que para este trabalho foi definido com o valor 100 (linha 30). O valor de 100 foi baseado em (SCHMITT, 2020).

Na linha 5, ocorre a inicialização das listas com a pontuação de cada operador. Similarmente aos pesos, as pontuações aumentam quando ocorre a melhora da solução. Mas, ao contrário do peso, a pontuação é atualizada a cada iteração. Nas linhas 6, ocorre a inicialização das listas com a contagem da quantidade de vezes que os operadores foram utilizados. Os valores de pontuação e contagem são importantes para o cálculo do peso (linha 32).

Na linha 8, a temperatura do SA é inicializada com o valor da função objetivo. Conforme mencionado na Seção 2.2.3, a temperatura do SA vai decaindo aos poucos até um estado final no qual a temperatura fica igual a zero. Inicializar a temperatura com o valor da função objetivo é interessante no contexto deste trabalho, pois o valor da função vem da soma do custo total e da recompensa não coletada, ou seja, nunca vai assumir valores negativos e o objetivo é minimizar a soma. Assim, ambos os valores serão minimizados durante a execução do algoritmo. Importante lembrar que quanto maior for a temperatura, maior a probabilidade de aceitar uma solução ruim. Assim, quanto mais demorar para ocorrer o resfriamento, mais soluções ruins podem ser aceitas no decorrer do algoritmo, permitindo maior diversidade no ALNS.

A partir da linha 9, inicia-se o laço principal do algoritmo que executa enquanto o número de iterações ainda não houver terminado. Na linha 11 e 12 acontece a seleção de um operador de destruição e reparo, respectivamente. Estes operadores serão aplicados sobre a solução atual na linha 13. Caso a nova solução seja melhor que a melhor solução conhecida até então (linha 16) a melhor solução é atualizada e o valor de $sigma$ é atualizado para 17. O $sigma$ é utilizado para atualizar o valor da pontuação do operador. Caso contrário (linha 19), ou seja, no caso em que os operadores não tenham obtido sucesso em encontrar uma nova melhor solução, utiliza-se a a probabilidade de aceitação, calculada de acordo com a fórmula na linha 15, que é similar a Equação 2.25, e um número gerado aleatoriamente na linha 14, para verificar se a nova solução, mesmo pior que a melhor solução até então, será aceita. Caso o número gerado aleatoriamente na linha 14 seja menor que a probabilidade calculada na linha 15 a solução é aceita. Assim, caso a nova solução seja melhor que a solução atual e pior que a melhor solução (verificado pela linha 20), a solução atual é substituída pela nova solução e o valor do $sigma$ é configurado para 47 (linhas 21 e 22). No caso em que a nova solução não é melhor que a solução atual, apenas ocorre a configuração do $sigma$ para 3 (linha 24). Na linha 26 ocorre a soma do valor de $sigma$ na pontuação dos operadores utilizados e na linha 27, ocorre o incremento da contagem dos operadores. Esses dois valores são importantes para o cálculo do peso, quanto maior os seus valores, mais eles foram utilizados e mais vão incrementar o valor do peso. Os três valores que são usados para o valor de $sigma$ foram encontrados utilizando a ferramenta de calibração de parâmetros *irace* (LóPEZ-IBáñEZ et al., 2016).

Na linha 29, ocorre o decréscimo da temperatura. A forma escolhida para o resfriamento foi a apresentada na Equação 3.10. A função de aceitação foi escolhida dentre as funções presentes em (Brian Luke, 2017). A escolha foi feita por meio de testes empíricos avaliando todas as funções com algumas instâncias do *benchmark* do VRPPTW. A atualização dos pesos dos operadores ocorre a cada 100 iterações (linha 30). Na atualização das pontuações dos operadores e da contagem, apenas os operadores que foram utilizados são atualizados. Na linha 32, ocorre o cálculo da atualização dos operadores, onde se leva em consideração o fator de reação f, o peso antigo p_o deste operador, a pontuação atual s_o do operador e a contagem atual c_o de vezes que o operador foi utilizado. Nas linhas 33 e 34 a pontuação e contagem de cada operador é inserido como 0. Remover a pontuação e a contagem ao atualizar os pesos garante que durante as próximas 100 iterações será refeita a contagem de usos e de pontuação para no futuro ocorrer o novo cálculo dos pesos.

$$T_i = \frac{1}{2}(T_0 - T_N) \times \left(1 + cos\left(\frac{i\pi}{N}\right)\right) + T_N \tag{3.10}$$

3.3 GERADOR DE SOLUÇÃO INICIAL

A representação da solução se dá por meio de uma matriz de requisições, onde cada linha da matriz é uma rota em especifico. A ordem de requisições na rota indica a ordem de visitas que os veículos devem realizar. Na implementação, cada requisição é representado por um objeto que contém as coordenadas do local, valor da recompensa, demanda (tem relação com a capacidade da frota, cada requisição tem uma demanda que consome a capacidade do veículo) e janela de atendimento.

Como o ALNS acaba por destruir e reparar a solução a cada iteração, existe a motivação para se criar um gerador de solução inicial simples, para evitar reprocessamento e desperdício de recursos computacionais. O Algoritmo 11 apresenta o funcionamento básico da geração da solução inicial.

O algoritmo inicia recebendo todas as requisições. Na linha 1, é inicializada uma solução vazia. Na linha 2, é inicializada a lista com as requisições que não foram inseridas na solução. Em seguida, para cada requisição v_i (linha 3) presente na lista de requisições que foi passada como parâmetro, é testada a inserção no final da rota e se essa inserção é pior que as inserções anteriores (linha 5). O critério para indicar a qualidade da inserção deriva da função objetivo. Na linha 10, se r_{melhor} não é igual a nulo, significa que foi possível encontrar uma posição na solução e a requisição vai ser inserida na rota que gere a pior rota, caso contrário, ocorre inserção na lista de requisições não utilizadas. Ao final, o algoritmo retorna a solução criada e a lista com as requisições não utilizadas.

O motivo de o gerador de soluções iniciais escolher a pior inserção serve para criar uma solução ruim e assim permitir que o ALNS consiga trabalhar em um espaço de busca maior. A função de aceitação do ALNS usa como temperatura inicial o valor da função objetivo e gera um valor que permite que soluções piores sejam aceitas com maior facilidade no inicio do algoritmo.

3.4 IMPLEMENTAÇÃO DOS OPERADORES DO ALNS

No funcionamento do algoritmo ALNS é necessário utilizar operadores de destruição e reparo. A Tabela 2 apresenta a lista de operadores que foram utilizados. É importante frisar que o número de remoções a cada iteração do algoritmo ALNS muda conforme a Equação 2.31.

São quatro operadores de destruição. O primeiro é de remoção aleatória que foi apresentado na Seção 2.3.1.1 e que realiza remoções escolhendo aleatoriamente requisições. O segundo operador é a remoção de pior custo (Seção 2.3.1.2). Esse operador procura a requisição que se for removida vai gerar o maior ganho. O ganho

Algoritmo 11 Template do gerador de solução inicial

Require: v /*Lista de requisições*/

1: $s = \{\}$
2: $us = \{\}$ /* que não foi possível serem inseridos pelo algoritmo de inserção.*/
3: **for** $v_i \in v$ **do**
4: $r_{pior} = null$
5: **for** $r \in s$ /*Para todas as rotas*/ **do**
6: **if** $insercao(v_i, r)$ for possivel **and** $insercao(v_i, r)$ for pior que $insercao(v_i, r_{melhor})$ **then**
7: $r_{pior} = r$
8: **end if**
9: **end for**
10: **if** $r_{pior} != null$ **then**
11: $s = insercao(v, r_{pior})$
12: **else**
13: $us = us \cup v_i$
14: **end if**
15: **end for**
16: **return** s e us

é modelado como sendo a diferença entre a distância total da rota da solução com requisição em relação a melhor rota possível (sem a requisição). Assim, ele busca remover a requisição que possui o maior gasto em deslocamento. O terceiro operador é de remoção relacional (Seção 2.3.1.3). Ele calcula um valor de similaridade e tende a remover requisições que possuam maior nível de similaridade entre si. Neste trabalho a forma do cálculo da similaridade é dado pela Equação 3.11, onde apenas a diferença da distância e diferença da demanda das requisições são consideradas e seus respectivos pesos são 9 e 7. Os pesos foram baseados no trabalho de Ropke e Pisinger (2006). O quarto operador é o de remoção de rotas (Seção 2.3.1.4). Para tanto, um número de rotas (que contêm as requisições) é escolhida aleatoriamente e destruída. A destruição consiste em remover todas as requisições da rota, deixando ela vazia. A proporção de remoções é de 40%, como em (SCHMITT, 2020).

$$similaridade(i, j) = 9 * (\Delta distancia) + 7 * (\Delta demanda) \tag{3.11}$$

Sobre os operadores de reparo, são cinco ao todo. O primeiro deles é a inserção gulosa (Seção 2.3.2.1). Ele busca inserir a requisição que gere o menor custo ao ser adicionado. O custo é modelado pela diferença entre a distância total das rotas da solução sem a requisição e a solução com a visita. Assim, o operador busca inserir sempre a requisição que gerar o menor aumento do custo. O outro operador de reparo foi a inserção com critério de arrependimento (Seção 2.3.2.2). Nele a inserção é feita e são feitos cálculos para verificar o custo de ter feito a inserção naquele mo-

mento. O custo das inserções futuras é o valor de arrependimento. O operador busca escolher as inserções com maior valor de arrependimento. Neste trabalho foram utilizados quatro operadores de arrependimento: arrependimento-2, arrependimento-3, arrependimento-4 e arrependimento-n, onde são avaliadas duas, três, quatro e *n* (onde *n* é o número máximo de rotas) visitas futuras antes da escolha. É importante frisar que cada operador de arrependimento tem uma versão onde há geração de ruído e outra sem geração de ruído. A geração de ruído serve para tentar evitar que o operador sempre busque inserir as mesmas requisições nos mesmos locais.

Tabela 2 – Operadores de destruição e reparo utilizado pelo algoritmo ALNS

Nome	Descrição
Remoção aleatória	Realiza remoções aleatórias
Remoção de pior custo	Remove as visitas que mais aumentem o custo da solução
Remoção relacional	Remove visitas baseado no critério de similaridade
Remoção de rota	Remove uma rota inteira
Inserção gulosa	Insere visitas de forma gulosa
arrependimento-2	Insere visitas pelo critério de arrependimento, verificando até duas visitas a frente
arrependimento-3	Insere visitas pelo critério de arrependimento, verificando até três visitas a frente
arrependimento-4	Insere visitas pelo critério de arrependimento, verificando até quatro visitas a frente
arrependimento-n	Insere visitas pelo critério de arrependimento, verificando até n visitas a frente, onde n é o número máximo de rotas

3.5 CONSIDERAÇÕES

Na Seção 3.1 foi apresentado a formulação matemática do problema VRPPTW. Foi apresentada a função objetivo, que é a maximização da soma do custo de deslocamento e recompensa não coletada. Também foram apresentadas as suas restrições, a saber, capacidade de cada veículos, janela de atendimento e tempo máximo da rota. O ponto que merece atenção nesta seção é a adaptação da função objetivo. Originalmente o VRPP maximiza o somatório da a diferença entre recompensa coletada e custo de deslocamento. Nesse caso, a função objetivo foi alterada para minimizar o somatório do custo de deslocamento e recompensa não coletada.

Na Seção 3.2 foi detalhado o algoritmo ALNS-SA. O funcionamento básico do algoritmo consiste em criar uma solução candidata e sobre ela aplicar dois tipos de operadores: destruição e reparo. Cada tipo de operador vai ter variações em seu funcionamento e possuir um peso. O valor do peso indica o quanto aquele operador obteve sucesso anteriormente em melhorar a solução. A cada nova iteração do algoritmo, os operadores serão novamente escolhidos levando em conta os pesos respectivos. Neste trabalho foi utilizada uma função resfriamento inspirada no *Simulated Anneling* (SA). O sistema inicia “quente”, permitindo maior aleatoriedade em sua execução. Durante a execução do algoritmo, sistema esfria, fazendo que tenha um comportamento menos aleatório. Importante dar ênfase no comportamento adaptativo do ALNS e a sua função de resfriamento. O algoritmo ALNS tem como característica adaptar a utilização dos operadores à instância sendo executada. A função de resfriamento busca explorar o espaço de busca tendo a possibilidade de escolher soluções piores do que a atual. A função de resfriamento escolhida tem como característica resfriar mais lentamente, mantendo o sistema mais aquecido e "esfriando"mais próximo do final da execução, tornando o algoritmo mais intensificador.

A Seção 3.3 apresenta o gerador de solução inicial, que busca gerar soluções ruins (valor alto para a função objetivo) para permitir, ao iniciar, que o algoritmo tenha um espaço de busca maior para explorar. O motivo de gerar soluções ruins é que o valor da função objetivo da solução inicial é o valor inicial da temperatura. Assim, a temperatura inicial e o valor da função objetivo ficam em escalas proporcionais. Iniciando a temperatura dessa forma, o sistema inicia "quente".

Na Seção 3.4 são apresentados os operadores de destruição e reparo utilizados neste trabalho. Assim, são utilizados quatro operadores de destruição e cinco operadores de reparo. Existe um conjunto de operadores para que a característica adaptativa do ALNS possa ser exercida. Para tanto, a Seção 4.4 disponibiliza uma análise mais detalhada.

4 EXPERIMENTOS E RESULTADOS

Neste capítulo são apresentados os experimentos executados e os resultados obtidos pelo método proposto. Na Seção 4.1, são apresentadas a metodologia de experimentação, a base de dados e outros detalhes da configuração dos experimentos. Na Seção 4.2 é feita uma comparação de desempenho com a literatura. Na Seção 4.3 são apresentados os resultados quantitativos de um experimento que compara o desempenho do algoritmo usando uma função objetivo que considera apenas o custo e outro que usa uma função objetivo que considera custo e recompensa. Finalmente, a Seção 4.4 apresenta uma análise qualitativa do funcionamento do algoritmo.

4.1 METODOLOGIA DE EXPERIMENTAÇÃO

Para testar a qualidade do algoritmo proposto foram utilizadas 168 instâncias do problema *Vehicle Routing Problem with Time Windows* (VRPTW), encontrados em Solomon (1987). Todas as instâncias representam um cenário diferente com requisições e número máximo de rotas diferentes. As instâncias podem ser encontradas em (Christelle Guéret et al., 2023). Ao todo existem 6 conjuntos de instâncias: c1, c2, r1, r2, rc1 e rc2. Dentro de cada conjunto de instâncias existem instâncias com 25, 50 e 100 requisições. A Tabela 3 apresenta um resumo das características das instâncias. Os conjuntos c1 e c2 são clusterizados, ou seja, as requisições ficam agrupadas, próximas entre si. Os conjuntos r1 e r2 são randomizados, ou seja, as requisição não necessariamente ficam próximas umas das outras. Os conjuntos rc1 e rc2 são randômico-clusterizados, portanto, algumas requisições estão agrupadas enquanto outras não.

Tabela 3 – Características das instâncias do *Vehicle Routing Problem with Time Windows* (VRPTW)

Grupo	Instâncias	Distribuição
c1	27	Clusterizado
c2	24	Clusterizado
r1	36	Randomizado
r2	33	Randomizado
rc1	24	Randomizado e Clusterizado
rc2	24	Randomizado e Clusterizado

As instâncias do VRPTW apresentam as coordenadas de cada requisição, número máximo de rotas, tempo máximo da rota, a janela de atendimento, quantidade e capacidade máxima dos veículos. Como são instâncias para o VRPTW e não para o VRPPTW, elas não apresentam valores de recompensas para cada local (visita)

atendida. Para isso, foi necessário utilizar um método para gerar os valores da recompensa. O método utilizado neste trabalho foi apresentado em Fischetti, Gonzalez e Toth (1998) e adaptado na Equação 4.1. A equação leva em consideração a distância para o depósito, e é proporcional à esta distância. O valor de θ é a soma do custo de deslocamento de todas as requisições e c_{0i} é o custo de deslocamento entre a requisição i e o depósito. Ainda existe um acréscimo de 50% para assegurar que a recompensa da requisição seja maior que o custo da distância. Esse acréscimo na recompensa serve para que o algoritmo tenha um motivo para inserir o máximo de requisições, ou seja, a recompensa coletada é maior que o custo de deslocamento.

$$r_i = (1 + \lfloor 99 * c_{0i}/\theta \rfloor) * 1.5 \tag{4.1}$$

Importante frisar que como não são conhecidos trabalhos na literatura onde a função objetivo considera o custo e a recompensa utilizando um *dataset* público como *benchmark*, não foi possível realizar uma comparação do VRPPTW com a literatura. Entretanto, foi realizado primeiramente um experimento onde a função objetivo do VRPPTW considerava apenas o custo de deslocamento, ou seja, o problema foi reduzido a solução de um VRPTW para comparar com os melhores resultados encontrados na literatura.

Posteriormente, um segundo experimento foi realizado com as instâncias do VRPPTW propostas neste trabalho para ter o valor da recompensa considerado na função objetivo. Este segundo experimento consistiu em realizar a execução das instâncias considerando apenas o custo de deslocamento na função objetivo e compará-los com a execução das mesmas instâncias considerando na função objetivo tanto o custo quanto a recompensa. Essa comparação foi realizada para identificar se com a consideração da recompensa na função objetivo a solução apresentava alguma vantagem em termos de minimização dos custo e/ou melhora na coleta de recompensas.

Outro ponto importante para o segundo experimento, é que as instâncias do VRPTW fornecem um número fixo de rotas para cada instância e as soluções ótimas usam apenas uma fração destas rotas. Para simular um cenário onde a coleta da recompensa seja relevante, nem todas as requisições devem ser atendidas, pois, caso contrário, a coleta seria total. Portanto, para que essa situação de escolha das coletas com maiores recompensas aconteça, buscou-se diminuir o número máximo de rotas para cada instância. Para isso, as instâncias foram executadas previamente para descobrir o número médio de rotas utilizadas na melhor solução e assim diminuir a quantidade de rotas disponíveis para forçar a situação onde não são inseridas todas as requisições. O número máximo de rotas para cada instância pode ser visto na Tabela 4.

Tabela 4 – Número máximo de rotas no segundo experimento

Instância	Rotas	Instância	Rotas	Instância	Rotas
C101-025	2	R102-100	16	R209-050	2
C101-050	4	R103-025	4	R209-100	3
C101-100	9	R103-050	7	R210-025	1
C102-025	2	R103-100	12	R210-050	2
C102-050	4	R104-025	3	R210-100	3
C102-100	9	R104-050	4	R211-025	1
C103-025	2	R104-100	9	R211-050	1
C103-050	4	R105-025	4	R211-100	2
C103-100	9	R105-050	7	RC101-025	3
C104-025	2	R105-100	13	RC101-050	6
C104-050	4	R106-025	4	RC101-100	14
C104-100	9	R106-050	6	RC102-025	2
C105-025	2	R106-100	11	RC102-050	6
C105-050	4	R107-025	3	RC102-100	13
C105-100	9	R107-050	5	RC103-025	2
C106-025	2	R107-100	9	RC103-050	5
C106-050	4	R108-025	3	RC103-100	10
C106-100	9	R108-050	4	RC104-025	2
C107-025	2	R108-100	9	RC104-050	3
C107-050	4	R109-025	4	RC104-100	9
C107-100	9	R109-050	6	RC105-025	3
C108-025	2	R109-100	11	RC105-050	6
C108-050	4	R110-025	4	RC105-100	13
C108-100	9	R110-050	5	RC106-025	2
C109-025	2	R110-100	9	RC106-050	6
C109-050	4	R111-025	3	RC106-100	11
C109-100	9	R111-050	5	RC107-025	2
C201-025	1	R111-100	9	RC107-050	5
C201-050	2	R112-025	3	RC107-100	10
C201-100	2	R112-050	4	RC108-025	2
C202-025	1	R112-100	9	RC108-050	3
C202-050	2	R201-025	3	RC108-100	9
C202-100	2	R201-050	4	RC201-025	2
C203-025	1	R201-100	4	RC201-050	4
C203-050	2	R202-025	2	RC201-100	5
C203-100	2	R202-050	3	RC202-025	2

C204-025	0	R202-100	4	RC202-050	4
C204-050	1	R203-025	2	RC202-100	4
C204-100	2	R203-050	2	RC203-025	2
C205-025	1	R203-100	3	RC203-050	3
C205-050	2	R204-025	1	RC203-100	3
C205-100	2	R204-050	1	RC204-025	1
C206-025	1	R204-100	3	RC204-050	2
C206-050	2	R205-025	2	RC204-100	2
C206-100	2	R205-050	2	RC205-025	2
C207-025	1	R205-100	3	RC205-050	4
C207-050	2	R206-025	1	RC205-100	4
C207-100	2	R206-050	1	RC206-025	2
C208-025	1	R206-100	2	RC206-050	3
C208-050	1	R207-025	1	RC206-100	4
C208-100	2	R207-050	1	RC207-025	1
R101-025	6	R207-100	2	RC207-050	3
R101-050	10	R208-025	0	RC207-100	4
R101-100	17	R208-050	1	RC208-025	1
R102-025	5	R208-100	2	RC208-050	1
R102-050	9	R209-025	1	RC208-100	3

O algoritmo foi implementado na linguagem Java versão openjdk 11.0.18 2023-01-17. O código foi executado na máquina com sistema operacional Ubuntu 20.04.6 LTS, processador Intel Core i7-11700 da 11ª geração de 16 núcleos e 32 GB de RAM. Cada instância foi executada 20 vezes para permitir a realização de testes estatísticos de significância devido ao comportamento estocástico do algoritmo ALNS-SA. Não houve paralelização na implementação do código em Java; apenas execução de *scripts* que tratavam as instâncias de forma concorrente.

4.2 COMPARAÇÃO COM *BENCHMARK* ORIGINAL DO VRPTW

Esta seção apresenta os resultados do algoritmo proposto. Na Tabela 5, é apresentado um resumo dos resultados obtidos pela comparação do algoritmo proposto com os dos trabalhos de Kohl et al. (1999) e Kallehauge et al. (2005). As instâncias, além de estarem separadas nos conjuntos apresentado na Tabela 3, também foram separadas em número de requisições. Resultados por instâncias podem ser vistos no Apêndice B. Os conjuntos são divididos em três, em proporção iguais. Assim, um conjunto com 27 instâncias tem 9 instâncias com 25 requisições, 9 com 50 requisi-

ções e 9 com 100 requisições. Para permitir a comparação, a função objetivo utilizada considerava apenas o valor do custo de deslocamento.

As colunas da Tabela 5 são as seguintes: *Instâncias* indica o conjunto de instâncias. A coluna *Estado da arte (EDA)* indica a média do melhor resultado para cada instância daquele conjunto obtido no estado da arte[1]. Os valores para comparação foram retirados dos trabalhos de Kohl et al. (1999) e Kallehauge et al. (2005). A coluna *Melhor* indica a média do melhor resultado encontrado para cada instância entre as vinte execuções do algoritmo proposto para aquele conjunto. A coluna *Média* indica a média do resultado médio alcançado nas vinte execuções do algoritmo. A coluna *Desvio Padrão* indica a média do desvio padrão apresentado durante as vinte execuções. A coluna *Diferença* indica a diferença entre as colunas *Estado da arte (EDA)* e *Melhor*. A coluna *RPD (%)* indica a métrica *Relative Percent Difference* (RPD) presente na Equação 4.2, onde F indica o valor da coluna *Melhor* e F_{EDA} indica o valor na coluna *Estado da arte (EDA)*. Essa métrica indica o quão distante o algoritmo ficou do resultado ótimo em porcentagem, se o valor for positivo (+), então significa que o algoritmo proposto foi prior, já se o valor for negativo (-), indica que o algoritmo proposto foi melhor. A coluna *Atingiu EDA* indica o número de instâncias daquele conjunto em que o algoritmo conseguiu atingir o estado arte.

$$RPD = \frac{F - F_{EDA}}{F_{EDA}} \tag{4.2}$$

Os resultados indicam que o algoritmo proposto teve um bom desempenho para instâncias com 25 requisições. Ao todo, em 42 instâncias o algoritmo chegou no estado da arte e o maior RPD foi de 1,87%. Assim, mesmo nas instâncias que o algoritmo não atingiu o estado da arte, o resultado ficou próximo dele. Observando os resultados para instâncias com 50 requisições, ao todo em 14 instâncias o algoritmo chegou ao estado da arte e a maior métrica RPD encontrada foi de 22,23%. Com relação ao conjunto de 100 requisições, foram 7 instâncias as que atingiram o estado da arte. O intervalo de variação do RPD, para esse conjunto, fica entre 0,07% e 9,22%. Em resumo, o algoritmo apresentou um bom desempenho em instâncias com 25 requisições, um resultado mais discreto nas instâncias com 50 requisições e um resultado intermediário com as instâncias com 100 requisições.

Ao observar os resultados por conjuntos completos, é possível notar que no conjunto de instâncias $c2$ foi alcançado o melhor resultado, onde apenas em uma instância não foi possível alcançar o estado da arte. No conjunto $r1$, em 7 instâncias foi atingido o estado da arte, embora o maior valor de RPD foi 0,72 %. O conjunto $r2$ apresentou 12 instâncias com resultado igual ao estado da arte. Nesse conjunto,

[1] Foi pego o melhor resultado para cada instância e depois calculado o resultado médio para aquele conjunto.

Tabela 5 – Comparação do desempenho do algoritmo ALNS-SA com benchmark

Instancias	Melhor - Estado da arte (EDA)	Melhor	Média	Desvio Padrão	Diferença	RPD (%)	Atingiu EDA
c1-025	190.59	190.97	190.97	0.00	0.38	0.20	8
c1-050	361.61	393.92	394.04	0.09	32.32	8.93	0
c1-100	826.52	893.71	895.18	1.01	67.20	8.13	0
c2-025	214.45	214.45	214.73	0.71	0.00	0.00	8
c2-050	357.50	357.50	357.50	0.00	0.00	0.00	8
c2-100	587.38	587.81	588.69	0.82	0.44	0.07	7
r1-025	460.70	463.37	463.41	0.06	2.67	0.63	5
r1-050	762.13	767.13	773.98	4.02	5.01	0.69	2
r1-100	1174.24	1182.68	1210.72	15.55	8.44	0.72	0
r2-025	382.15	382.15	383.47	1.67	0.00	0.00	11
r2-050	608.30	623.41	633.65	4.84	15.11	2.73	1
r2-100	856.65	893.61	920.29	12.49	36.95	4.39	0
rc1-025	342.77	350.24	350.29	0.08	7.47	1.87	2
rc1-050	663.02	804.26	809.12	2.35	141.24	22.23	0
rc1-100	1261.58	1361.94	1395.68	16.88	100.36	9.22	0
rc2-025	319.27	319.27	321.26	2.70	0.00	0.00	8
rc2-050	565.17	575.33	582.53	5.26	10.16	2.05	3
rc2-100	1001.01	1024.77	1059.87	16.30	23.76	2.44	0
Total							63

as 11 instâncias com 25 requisições atingiram o estado da arte. Apenas 1 instância com 50 requisições do conjunto $r2$ chegou ao estado da arte, embora o RPD geral desse conjunto foi de 2,73%. O RPD do conjunto $r2$ com 100 requisições ficou em 4,39%. O conjunto $rc2$ teve 11 instâncias com resultado igual ao estado da arte, com valor de RPD de 0,00%, 2,05% e 2,44%, para 25 requisições, 50 requisições e 100 requisições, respectivamente. O conjunto $c1$ apresentou 8 instâncias com resultado igual ao estado da arte e RPD de 0,20 %, 8,93% e 8,13% para 50 requisições e 100 requisições, respectivamente. O conjunto com resultados mais tímidos foi o $rc1$, onde apenas 2 instâncias obtiveram resultados igual ao estado da arte. Mas quando se olha os valores de RPD, tem-se 9,22% para 100 requisições e 22,23% para 50 requisições. Esses resultados parecem indicar que o algoritmo tem uma maior facilidade em resolver as instâncias com disposição de requisições aleatória, pois houve resultados consistentes nos grupos $r1$ e $r2$. Os conjuntos $c1$ e $c2$, que têm disposição de requisições *clusterizados*, o algoritmo obteve um resultado intermediário. O desempenho no conjunto $c2$ foi excepcionalmente bom, embora o resultado no conjunto $c1$ tenha sido mais discreto. Por último, os conjuntos $rc1$ e $rc2$ (com disposição de requisições misto entre aleatório e clusterizado) foram os que tiveram os resultados mais tímidos. O conjunto $rc2$ obteve um bom resultado, mas o conjunto $rc1$ obteve o pior resultado.

4.3 ANÁLISE QUANTITATIVA DO SEGUNDO EXPERIMENTO

Nesta seção é apresentada uma análise dos resultados obtidos pelo experimento que realiza a comparação entre uma execução do algoritmo com função objetivo que considera apenas o custo e outra execução do algoritmo que considera o custo e a recompensa em sua função objetivo. Na Tabela 6 são apresentados os resultados para cada grupo de instâncias do VRPTW. Resultados por instância podem ser visto no Apêndice C. A coluna *Instâncias* indica o conjunto de instâncias. O prefixo indica o conjunto e o sufixo indica a quantidade de requisições. A coluna C_1 indica o valor do custo quando a função objetivo apenas considera o custo. A coluna C_2 indica o valor do custo quando a função objetivo considera o custo e a recompensa não coletada. As colunas R_1 e R_2 indicam os valores da recompensa não coletada para quando a função objetivo considera apenas o custo e para quando a função objetivo considera o custo e a recompensa não coletada, respectivamente. Importante frisar que o valor de R_1 são os valores de recompensa das requisições não atendidas na solução do problema quando a função objetivo considera apenas o custo. A coluna *DC* indica a diferença do custo entre C_2 e C_1 e a coluna *DR* indica a diferença entre as colunas R_2 e R_1. Todas as colunas apresentam a média do valor da melhor solução para as vinte execuções da instância.

Como métrica de comparação entre as versões do algoritmo foi utilizado o *Relative Percent Difference* (RPD) entre o valor obtido pelo algoritmo com a função objetivo considerando apenas o custo e o algoritmo com a função objetivo considerando o custo e a recompensa não coletada. O valor na coluna *RPDC*, dado pela Equação 4.3, é obtido pelo calculo considerando o valor do custo para as duas versões da função objetivo. A Equação 4.4 considera o valor da recompensa não coletada para as duas versões da função objetivo para gerar o valor que está presente na coluna *RPDR*. As colunas *% ReqC* e *% ReqR* indicam a proporção de requisições utilizadas na solução pela função objetivo apenas considerando o custo e pela função objetivo considerando o custo e recompensa não coletada, respectivamente.

A coluna *pC* indica o valor de p para o teste de Wilcoxon comparando o custo entre as duas versões da função objetivo e a coluna *pR* indica o valor de p para recompensa não coletada entre as duas versões da função objetivo. O objetivo do teste dos sinais de Wilcoxon é comparar as performances de cada sujeito (ou pares de sujeitos) no sentido de verificar se existem diferenças significativas entre os seus resultados nas duas situações. Os resultados da Situação B são subtraídos dos da Situação A e à diferença resultante (d) é atribuído o sinal mais (+) ou, caso seja negativa, o sinal menos (-). Estas diferenças são ordenadas em função da sua grandeza (independentemente do sinal positivo ou negativo). A ideia é que se existirem apenas diferenças aleatórias, tal como é postulado pela hipótese nula, então haverá aproximadamente

o mesmo número de ordens elevadas e de ordens inferiores tanto para as diferenças positivas como negativas. Se verificar uma preponderância de baixos resultados para um dos lados, isso significa a existência de muitos resultados elevados para o outro lado, indicando uma diferença em favor de uma das situações, superior àquilo que seria de esperar se os resultados se devessem ao acaso (LARSON; FARBER, 2010).

Tabela 6 – Comparação dos resultados para ambas funções objetivos.

Instâncias	C_1	C_2	R_1	R_2	DC	DR	RPDC (%)	RPDR (%)	% ReqC	% ReqR	pC	pR
c1-025	125.48	161.28	96.66	32.02	35.80	-64.64	0.29	-0.67	0.55	0.81	0.0000	0.0000
c1-050	277.63	290.79	81.14	57.01	13.16	-24.13	0.05	-0.30	0.71	0.80	0.0000	0.0000
c1-100	788.68	793.88	58.26	47.19	5.21	-11.07	0.01	-0.19	0.85	0.88	0.2980	0.0000
c2-025	151.21	198.67	94.81	20.42	47.46	-74.39	0.31	-0.78	0.55	0.89	0.0000	0.0000
c2-050	331.22	345.65	56.11	35.35	14.43	-20.76	0.04	-0.37	0.78	0.86	0.0002	0.0000
c2-100	517.46	543.75	174.48	153.53	26.29	-20.94	0.05	-0.12	0.52	0.57	0.0001	0.0000
r1-025	302.49	309.61	75.72	63.39	7.12	-12.33	0.02	-0.16	0.67	0.72	0.0000	0.0000
r1-050	525.87	533.03	90.45	79.83	7.16	-10.62	0.01	-0.12	0.68	0.72	0.0002	0.0000
r1-100	954.36	958.52	78.39	71.46	4.16	-6.93	0.00	-0.09	0.81	0.83	0.2583	0.0000
r2-025	319.89	341.29	55.65	31.66	21.40	-23.99	0.07	-0.43	0.73	0.85	0.0000	0.0000
r2-050	502.19	523.32	89.88	66.84	21.13	-23.05	0.04	-0.26	0.66	0.75	0.0000	0.0000
r2-100	872.49	884.29	50.67	35.20	11.80	-15.47	0.01	-0.31	0.87	0.91	0.0012	0.0000
rc1-025	190.69	220.16	120.76	74.22	29.48	-46.54	0.15	-0.39	0.42	0.64	0.0000	0.0000
rc1-050	494.34	512.71	96.00	71.64	18.37	-24.36	0.04	-0.25	0.64	0.73	0.0000	0.0000
rc1-100	1095.86	1109.12	84.72	69.39	13.26	-15.33	0.01	-0.18	0.80	0.83	0.0016	0.0000
rc2-025	325.82	341.79	30.58	12.14	15.97	-18.44	0.05	-0.60	0.85	0.94	0.0000	0.0000
rc2-050	582.66	586.16	22.09	17.87	3.50	-4.22	0.01	-0.19	0.91	0.93	0.5241	0.0015
rc2-100	1052.21	1064.63	32.37	23.20	12.42	-9.17	0.01	-0.28	0.92	0.94	0.0182	0.0000

Importante notar que para os grupos de instâncias houve um aumento do custo e uma diminuição da recompensa não coletada. O fator que diminui a recompensa não coletada é o aumento das requisições atendidas pelas rotas montadas. Como foi feito o acréscimo de 50% no valor da recompensa da requisição para garantir que o algoritmo tenha motivo para realizar a inserção, o algoritmo buscou de fato utilizar o maior número possível de requisições. Como o valor da recompensa é maior do que o valor do custo de deslocamento, o algoritmo busca inserir a requisição. As colunas do valor de p também indicam que houve relevância estatística entre o custo e a recompensa não coletada. Os grupos de instâncias que não tiveram diferença significativa no valor do custo foram o c1-100, r1-100 e rc2-050. Os outros grupos apresentaram diferença. No quesito recompensa não coletada, todos os grupos apresentaram diferença significativa.

$$RPDC = \frac{C_2 - C_1}{C_1} \tag{4.3}$$

$$RPDR = \frac{R_2 - R_1}{R_1} \tag{4.4}$$

4.4 ANÁLISE QUALITATIVA

Esta seção apresenta uma análise do comportamento do algoritmo proposto durante sua execução. Duas análises são realizadas: i) análise da variação dos pesos dos operadores; ii) análise da variação da qualidade da solução e temperatura do sistema. Ao todo, três instâncias foram escolhidas para fornecer esses dados: *c208-100*, *r211-100* e *rc208-100*. Todas essas instâncias apresentam valores altos para a função objetivo e são computacionalmente custosas.

A primeira análise é apresentada nas Figuras 5, 6 e 7. Nela são apresentados três gráficos para as instâncias c208-100, r211-100 e rc208-100, respectivamente. Em cada gráfico são indicados os valores do custo da solução atual, custo da melhor solução e temperatura do *Simulated Anneling* (SA). Aqui é importante notar o decaimento da temperatura. A função de resfriamento faz com que a temperatura inicial seja igual ao custo da função objetivo da solução inicial e vá decaindo até atingir zero. Durante o decaimento, novas soluções piores geradas pelo algoritmo podem ser aceitas. Contudo, quando a temperatura se iguala a zero, o algoritmo não é mais capaz de aceitar soluções piores.

O primeiro ponto a se observar é a rápida convergência da melhor solução, ou seja, já nas primeiras iterações a função objetivo fica próxima da solução que o algoritmo consegue chegar. A rápida conversão do algoritmo pode ser um problema pois restringe o espaço de busca rapidamente e ele pode ficar preso em um ótimo

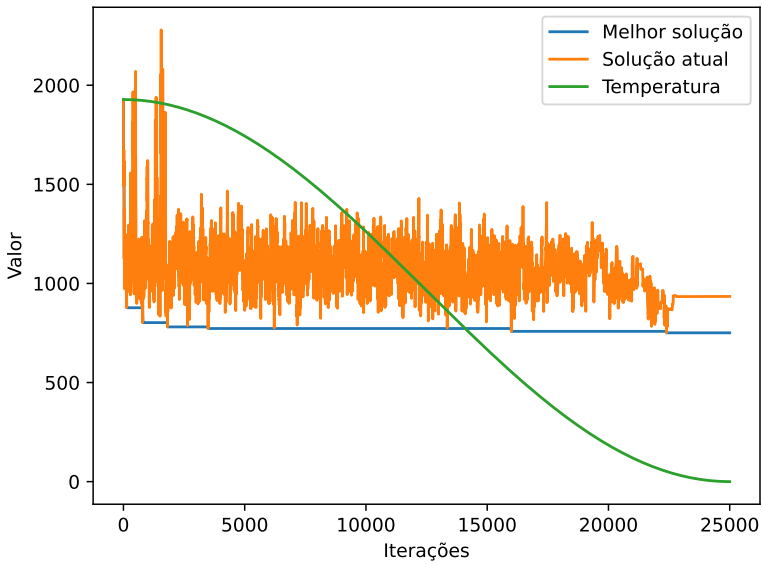

Figura 5 – Qualidade da solução e temperatura na instância c208-100. Temperatura esta na mesma escala de variação do valor da solução.

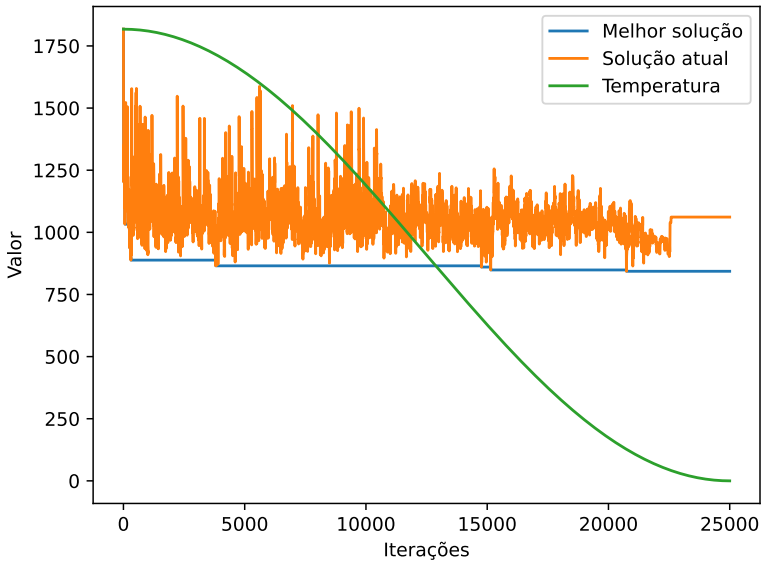

Figura 6 – Qualidade da solução e temperatura na instância cr211-100. Temperatura esta na mesma escala de variação do valor da solução.

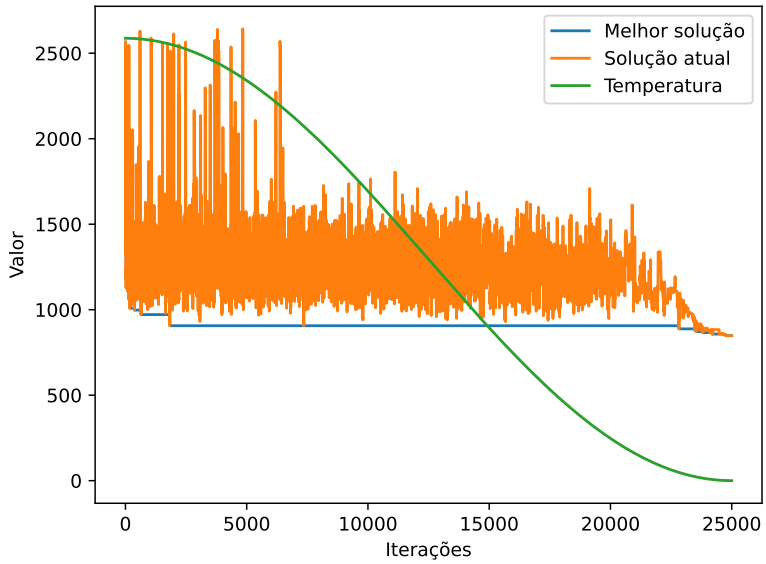

Figura 7 – Qualidade da solução e temperatura na instância rc208-100. Temperatura esta na mesma escala de variação do valor da solução.

local. Para instâncias menores, esse comportamento é comum e até aceitável devido ao espaço de busca restrito. Mas, para instâncias maiores, pode ser um indicativo de estagnação prematura da exploração do algoritmo proposto.

Na segunda análise, é observada a variação dos pesos dos operadores. As Figuras 8, 10 e 12 apresentam a variação para os operadores de destruição e as Figuras 9, 11 e 13 apresentam a variação para os operadores de inserção. A primeira característica a ser apontada é sobre a forma do gráfico. Os pesos sobem bastante no início do algoritmo e depois vão decaindo lentamente. É importante notar que quanto maior o peso do operador, mais sucesso obteve na melhora da solução. Os pesos aumentam muito no início, pois o algoritmo está convergindo muito rapidamente, ou seja, está encontrando vários ótimos globais rapidamente.

O operador de destruição *Remoção Aleatória* acaba se destacando como sendo o mais utilizado durante a execução do algoritmo e o *Remoção do pior custo* acabou sendo o que menos obteve sucesso. Interessante notar que o operador *Remoção de rota* foi bastante utilizado na instância rc208-100, mas pouco utilizado nas outras duas instâncias. Isso demonstra que utilizar o algoritmo *Adaptive Large Neighborhood Search* (ALNS) (que alterna a utilização dos operadores dependendo do cenário) se mostra útil, pois cada instância tem particularidades próprias. Em relação aos operadores de inserção, nenhum dos operadores se sobressai em comparação aos demais. Apenas é possível verificar que o operador *Inserção Gulosa* não teve bom desempenho, chegando a interromper sua utilização prematuramente.

Portanto, por meio da análise a partir dos gráficos, percebe-se a rápida con-

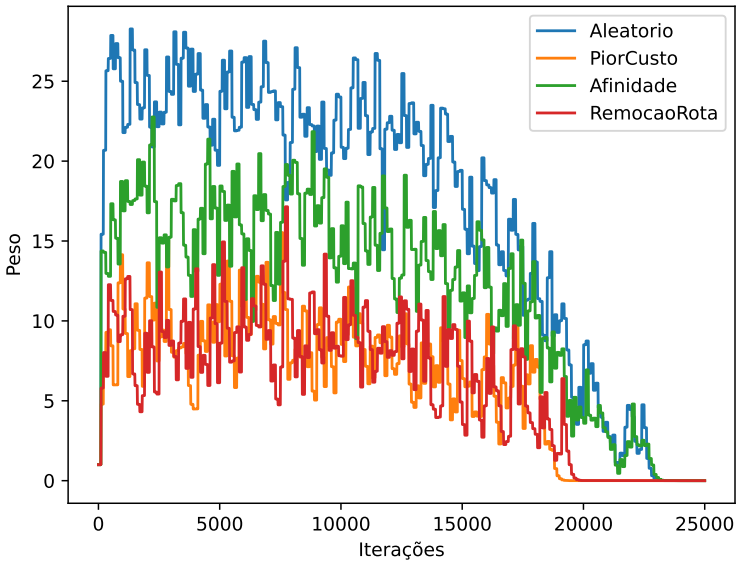

Figura 8 – Peso operadores destruição na instância c208-100.

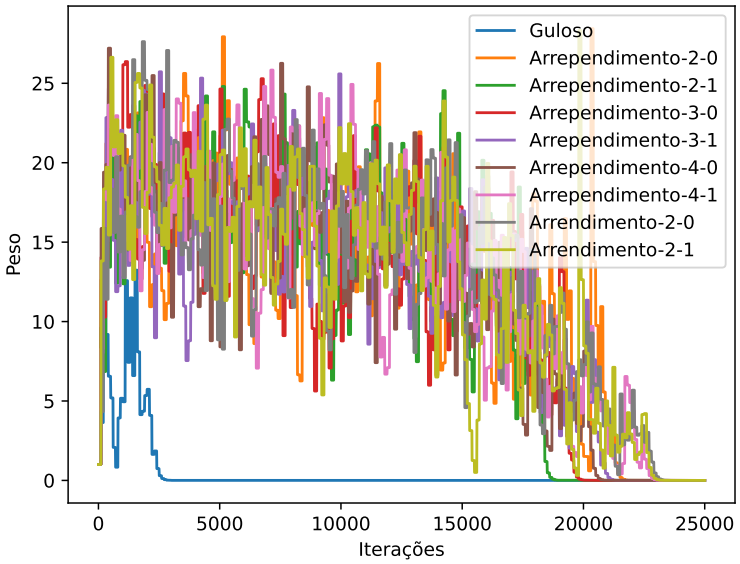

Figura 9 – Peso operadores reparo na instância c208-100

vergência do algoritmo no seu início. Também é possível notar a alta utilização do operador *Remoção aleatória*. Contudo, também é notório que cada instância pode ter variação na utilização de seus operadores. Com relação aos operadores de inserção, nenhum conseguiu se sobressair, mas o operador *Inserção Gulosa* teve o pior desempenho entre todos.

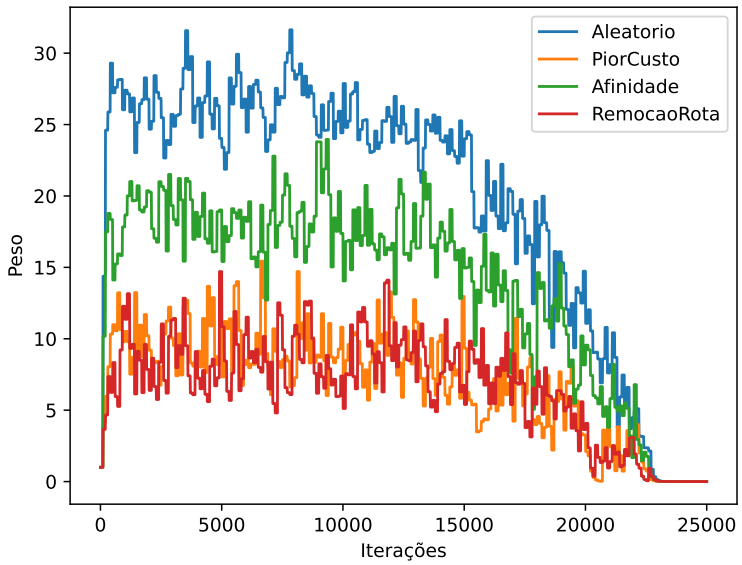

Figura 10 – Peso operadores destruição na instância r211-100

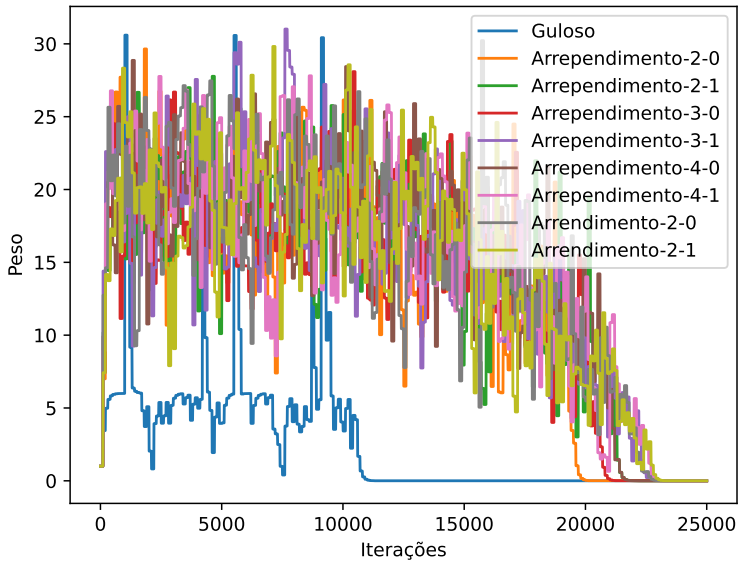

Figura 11 – Peso operadores reparo na instância r211-100

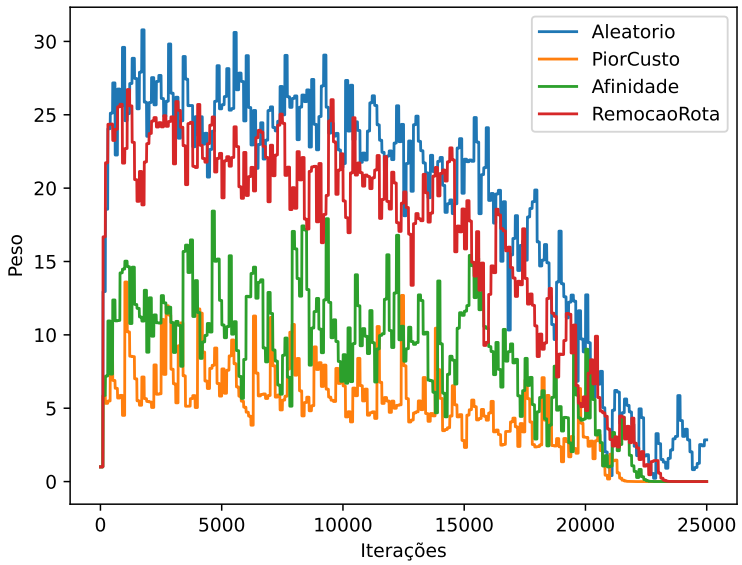

Figura 12 – Peso operadores destruição na instância rc208-100

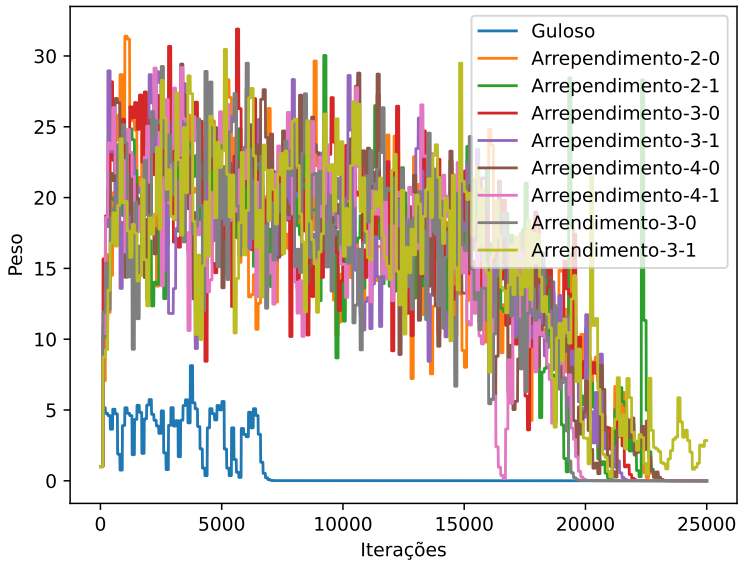

Figura 13 – Peso operadores reparo na instância rc208-100

5 CONCLUSÃO

Este trabalho apresentou a variante VRPPTW do problema de roteamento de veículos. Nela, além das restrições comuns do VRP, como por exemplo, capacidade da frota e tempo máximo da rota, foram adicionadas as restrições de janela de tempo e alteração da função objetivo que minimiza a soma entre a recompensa não coletada e custo total da frota. Para resolver o problema, foi apresentado um algoritmo baseado na metaheuristica ALNS. O ALNS tem como objetivo gerar uma solução inicial e aplicar operadores de destruição e reparo a fim de melhorar a solução existente e assim chegar a melhor solução global para o problema. O algoritmo é adaptativo e por isso vai alterando quais operadores são utilizados levando em consideração o sucesso dos mesmos em iterações anteriores. O SA funciona aplicando o conceito de aquecimento e resfriamento de metais para permitir que soluções piores que as atuais possam ser aceitas enquanto o algoritmo está no processo de exploração do espaço de busca.

O algoritmo proposto foi avaliado utilizando instâncias do problema VRPTW adaptadas com a adição de recompensa a cada requisição/cliente a ser visitado. As instâncias do VRPTW foram escolhidas por conter, entre outras restrições, janelas de atendimento. No primeiro experimento foi realizada uma comparação com a literatura, ainda sem considerar a recompensa na função objetivo. O algoritmo apresentou um desempenho inferior, abrindo espaço para melhorias em trabalhos futuros. Contundo no segundo experimento foi realizada uma comparação entre duas versões do algoritmo proposto, uma que considerava apenas o custo e outra que considerava o custo e a recompensa não coletada. Foi observado que apesar de haver um aumento do custo e houve uma diminuição da recompensa não coletada quando a recompensa faz parte da função objetivo, hipótese levantada por este trabalho.

Por último, foi realizada um análise da curva de convergência e da curva de pesos dos operadores do algoritmo, onde foi possível observar que o algoritmo apresenta uma convergência muito acelerada no início, ocasionando sua estagnação prematura. Isso pode ser o motivo pelo qual ele não obteve melhores soluções que os algoritmos estado da arte encontrados na literatura. Ainda, foi observado que os operadores de destruição conseguiram se adaptar melhor as particularidades das instâncias analisadas, enquanto os de reparo não tiveram a mesma resposta. Fato esse observado pelo comportamento do operador *Inserção Gulosa*, que teve sua utilização prematuramente interrompida, muito antes dos outros operadores.

Portanto, o trabalho traz como contribuições a formalização do VRPPTW como um problema de minimização da soma dos custos das rotas e da soma das recom-

pensas não coletas, fazendo, portanto, a simplificação do problema que originalmente foi modelado como uma maximização da diferença entre o custo das rotas e total de recompensas coletadas. Ainda, como contribuição adicional, o trabalho propôs uma adaptação ao *benchmark* para VRPTW, adicionando o valor da recompensa associada a cada requisição. Entretanto, é possível notar que o algoritmo ainda não atingiu um comportamento adequado para superar os algoritmos estado da arte na solução de VRPTW, portanto, ainda há espaço para a realização de melhorias nele.

5.1 TRABALHOS FUTUROS

Como sequência ao trabalho apresentado, os seguintes itens ficam em aberto para trabalhos futuros:

- Adaptar um *benchmark* onde seja possível realizar uma comparação com a literatura utilizando a função objetivo com custo e recompensa não coletada.
- Utilizar outros operadores no ALNS. Por exemplo, apenas dois operadores de inserção foram utilizados, sendo alguns variações do operador de Inserção por Arrependimento. Assim, utilizar outros poderia impactar o algoritmo.
- Aplicar um gerador de soluções iniciais diferente. O gerador atual apenas gera uma solução inicial ruim.
- Aplicar outras funções de aceitação. Atualmente o algoritmo tem uma tendência de convergência no inicio de sua execução. Ajustar essa convergência com outra função poderia levar o algoritmo a aceitar soluções que o levariam a melhores resultados finais.
- Verificar quais operadores geram impacto na execução do algoritmo. A forma de verificação pode ser executar as instâncias com apenas um ou um subconjunto de operadores e verificar a relevância estatística do resultado em comparação com a execução da instância com todos os operadores.
- Modelar e executar as instâncias em um *solver* de programação inteira para encontrar a solução ótima das instâncias VRPTW com adição de recompensa.

REFERÊNCIAS

AARTS, E.; AARTS, E. H.; LENSTRA, J. K. **Local search in combinatorial optimization**. Princeton, Estados Unidos: Princeton University Press, 2003.

AKSEN, D.; ARAS, N. Customer selection and profit maximization in vehicle routing problems. In: **Operations research proceedings 2005**. [S.l.]: Springer, 2006. p. 37–42.

ARCHETTI, C. et al. The capacitated team orienteering and profitable tour problems. **Journal of the Operational Research Society**, Springer, Nova York, Estados Unidos, v. 60, n. 6, p. 831–842, 2009.

ARCHETTI, C.; SPERANZA, M. G.; VIGO, D. Chapter 10: Vehicle routing problems with profits. In: **Vehicle Routing: Problems, Methods, and Applications, Second Edition**. Philadelphia, Estados Unidos: SIAM, 2014. p. 273–297.

BAYER, F. R. **Proposta de algoritmo de busca larga em vizinhança para o problema de roteamento de veículos com suporte a múltiplas janelas de tempo.** Dissertação de mestrado — Universidade do Estado de Santa Catarina, 2021.

BEN-SAID, A.; EL-HAJJ, R.; MOUKRIM, A. A variable space search heuristic for the capacitated team orienteering problem. **Journal of Heuristics**, Springer, v. 25, n. 2, p. 273–303, 2019.

BERALDI, P. et al. The risk-averse traveling repairman problem with profits. **Soft Computing**, Springer, v. 23, p. 2979–2993, 2019.

Brian Luke. **Simulated Annealing Cooling Schedules**. 2017. Https://web.archive.org/web/20171231002349/http://www.btluke.com/simanf1.html. Acessado em 19/07/2023.

Bridget McCrea. **Transportation Best Practices/Trends: Make me a match**. 2019. Https://www.logisticsmgmt.com/article/transportation_best_practices_trends_make_me_a_match/motorfreight. Acessado em 02/05/2022.

ČERNỲ, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. **Journal of optimization theory and applications**, Springer, v. 45, n. 1, p. 41–51, 1985.

CHAO, I.-M.; GOLDEN, B. L.; WASIL, E. A. The team orienteering problem. **European journal of operational research**, Elsevier, v. 88, n. 3, p. 464–474, 1996.

Christelle Guéret et al. **Instâncias do VRP**. 2023. Http://www.vrp-rep.org/variants/item/vrptw.html. Acessado em 02/07/2023.

CHRISTOFIDES, N. The vehicle routing problem. **Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle**, EDP Sciences, v. 10, n. V1, p. 55–70, 1976.

DANG, D.-C.; GUIBADJ, R. N.; MOUKRIM, A. An effective pso-inspired algorithm for the team orienteering problem. **European Journal of Operational Research**, Elsevier, v. 229, n. 2, p. 332–344, 2013.

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

DESROCHERS, M.; DESROSIERS, J.; SOLOMON, M. A new optimization algorithm for the vehicle routing problem with time windows. **Operations research**, INFORMS, v. 40, n. 2, p. 342–354, 1992.

EL-HAJJ, R. et al. A pso based algorithm with an efficient optimal split procedure for the multiperiod vehicle routing problem with profit. **Annals of Operations Research**, Springer, v. 291, n. 1, p. 281–316, 2020.

FISCHETTI, M.; GONZALEZ, J. J. S.; TOTH, P. Solving the orienteering problem through branch-and-cut. **INFORMS Journal on Computing**, INFORMS, v. 10, n. 2, p. 133–148, 1998.

GENDREAU, M.; POTVIN, J.-Y. et al. **Handbook of metaheuristics**. United States: Springer, 2010. v. 2.

KALLEHAUGE, B. et al. Vehicle routing problem with time windows. In: **Column generation**. Nova York, Estados Unidos: Springer, 2005. p. 67–98.

KIRKPATRICK, S.; JR, C. D. G.; VECCHI, M. P. Optimization by simulated annealing. **science**, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983.

KOHL, N. et al. 2-path cuts for the vehicle routing problem with time windows. **Transportation science**, INFORMS, v. 33, n. 1, p. 101–116, 1999.

KONTORAVDIS, G.; BARD, J. F. A grasp for the vehicle routing problem with time windows. **ORSA journal on Computing**, INFORMS, v. 7, n. 1, p. 10–23, 1995.

LABADIE, N. et al. The team orienteering problem with time windows: An lp-based granular variable neighborhood search. **European Journal of Operational Research**, Elsevier, v. 220, n. 1, p. 15–27, 2012.

LAKATOS, E. M.; MARCONI, M. d. A. Fundamentos de metodologia científica. 5. reimp. **São Paulo: Atlas**, v. 310, 2007.

LAPORTE, G. Fifty years of vehicle routing. **Transportation science**, INFORMS, v. 43, n. 4, p. 408–416, 2009.

LARSON, R.; FARBER, B. **Estatística Aplicada**. Brasil: Pearson Prentice Hall, 2010. v. 4.

LESTARI, F. Vehicle routing problem using sweep algorithm for determining distribution routes on blood transfusion unit. In: **Proceedings of the Second Asia Pacific International Conference on Industrial Engineering and Operations Management**. [S.l.]: Indonésia, 2021.

LIU, R.; TAO, Y.; XIE, X. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. **Computers & Operations Research**, Elsevier, v. 101, p. 250–262, 2019.

LYSGAARD, J.; LETCHFORD, A. N.; EGLESE, R. W. A new branch-and-cut algorithm for the capacitated vehicle routing problem. **Mathematical Programming**, Springer, v. 100, n. 2, p. 423–445, 2004.

LóPEZ-IBáñEZ, M. et al. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43–58, 2016.

NACCACHE, S.; CÔTÉ, J.-F.; COELHO, L. C. The multi-pickup and delivery problem with time windows. **European Journal of Operational Research**, Elsevier, v. 269, n. 1, p. 353–362, 2018.

ORLIS, C. et al. Distribution with quality of service considerations: The capacitated routing problem with profits and service level requirements. **Omega**, Elsevier, v. 93, p. 102034, 2020.

PEDRO, O.; SALDANHA, R.; CAMARGO, R. A tabu search approach for the prize collecting traveling salesman problem. **Electronic Notes in Discrete Mathematics**, Elsevier, v. 41, p. 261–268, 2013.

PISINGER, D.; ROPKE, S. A general heuristic for vehicle routing problems. **Computers & operations research**, Elsevier, v. 34, n. 8, p. 2403–2435, 2007.

PSARAFTIS, H. N. Dynamic vehicle routing: Status and prospects. **Annals of operations research**, Springer, v. 61, n. 1, p. 143–164, 1995.

Rodrigo Martucci. **Covid-19: o impacto da pandemia no comportamento de compra online**. 2021. Https://www.ecommercebrasil.com.br/artigos/covid-19-o-impacto-da-pandemia-no-comportamento-de-compra-online/. Acessado em 18/04/2022.

ROPKE, S.; PISINGER, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. **Transportation science**, Informs, v. 40, n. 4, p. 455–472, 2006.

SACRAMENTO, D.; PISINGER, D.; ROPKE, S. An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. **Transportation Research Part C: Emerging Technologies**, Elsevier, v. 102, p. 289–315, 2019.

SCHMITT, J. P. **Proposta de algoritmo de busca larga adaptativa em vizinhança para o problema de roteamento de veículos com coleta e entrega, janela de tempo e requisições dinâmicas.** Dissertação de mestrado — Universidade do Estado de Santa Catarina, 2020.

SHAW, P. Using constraint programming and local search methods to solve vehicle routing problems. In: SPRINGER. **International conference on principles and practice of constraint programming**. [S.l.], 1998. p. 417–431.

SOLOMON, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. **Operations research**, Informs, v. 35, n. 2, p. 254–265, 1987.

STAVROPOULOU, F.; REPOUSSIS, P. P.; TARANTILIS, C. D. The vehicle routing problem with profits and consistency constraints. **European Journal of Operational Research**, Elsevier, v. 274, n. 1, p. 340–356, 2019.

Steve Banker. **The UPS-Workhorse Group Deal Highlights Advances In Last-Mile Routing**. 2018. Https://www.forbes.com/sites/stevebanker/2018/02/23/the-upsworkhorse-group-deal-highlights-advances-in-last-mile-routing/?sh=63a3be6a55ed. Acessado em 02/05/2022.

TALBI, E.-G. **Metaheuristics: from design to implementation**. United States: John Wiley & Sons, 2009. v. 74.

TOTH, P.; VIGO, D. An overview of vehicle routing problems. **The vehicle routing problem**, SIAM, p. 1–26, 2002.

TRIPP, D. Pesquisa-ação: uma introdução metodológica. **Educação e pesquisa**, Faculdade de Educação, v. 31, n. 03, p. 443–466, 2005.

APÊNDICES

A REPOSITÓRIO

O código fonte, instâncias e *scripts* do algoritmo ALNS-SA estão em um repositório do gitlab. Conteúdo das pastas:

- algorithm: classes principais do programa e seus operadores.
- build: pasta onde ficam os arquivos compilados do java.
- instances: pasta com as instâncias. Os arquivos ChangerA.java, ChangerB.java e ChangerC.java são os arquivos que geram as recompensas nas instâncias do VRPTW.
- irace: pasta onde aconteceu a calibração dos parâmetros sigma.
- lib: dependencias do projeto.
- model: Classes modelos do projeto.
- scripts: pasta com scripts de processamento e logs de experimentos.
- test: pasta com testes do programa.
- utils: Classes utilitárias do projeto.

Arquivos importantes:

- App.java: classe principal do programa.
- build.sh: Arquivo de compilação e construção do jar.
- make.sh: Compila e executa o programa.

B RESULTADOS POR INSTÂNCIA NA COMPARAÇÃO COM *BENCHMARK*

Neste apêndice são apresentados, instância por instância, os resultados obtidos pelo experimento que realiza a comparação do algoritmo proposto neste trabalho com resultados da literatura em um *benchmark*. Mais detalhes podem ser obtidos na Seção 4.2.

Os resultados são apresentados nas Tabelas 7, 8, 9, 10, 11 e 12. Cada uma delas expõe o resultado do experimento de comparação para um conjunto de instâncias. Ao todo existem 6 conjuntos de instâncias: c1, c2, r1, r2, rc1 e rc2. Dentro de cada conjunto de instâncias existem instâncias com 25, 50 e 100 requisições. Os conjuntos c1 e c2 são *clusterizados*, ou seja, as requisições ficam agrupadas, próximas entre si. Os conjuntos r1 e r2 são randomizados, ou seja, as requisição não necessariamente ficam próximas umas das outras. Os conjuntos rc1 e rc2 são randômico-*clusterizados*, portanto, algumas requisições estão agrupadas enquanto outras não.

Nas tabelas mencionadas, a coluna *Instâncias* indica a instância comparada. A coluna *Estado da arte (EDA)* indica a média do melhor resultado para cada instância daquele conjunto obtido no estado da arte[1]. Os valores para comparação foram retirados dos trabalhos de Kohl et al. (1999) e Kallehauge et al. (2005). A coluna *Melhor* indica a média do melhor resultado encontrado para cada instância entre as vinte execuções do algoritmo proposto para aquele conjunto. A coluna *Média* indica a média do resultado médio alcançado nas vinte execuções do algoritmo. A coluna *Desvio Padrão* indica a média do desvio padrão apresentado durante as vinte execuções. A coluna *Diferença* indica a diferença entre as colunas *Estado da arte (EDA)* e *Melhor*. A coluna *RPD (%)* indica a métrica *Relative Percent Difference* (RPD) presente na Equação 4.2, onde F indica o valor da coluna *Melhor* e F_{EDA} indica o valor na coluna *Estado da arte (EDA)*. Essa métrica indica o quão distante o algoritmo ficou do resultado ótimo em porcentagem. Assim, se o valor for positivo (+), isso significa que o algoritmo proposto obteve um resultado pior que o do EDA; já se o valor for negativo (-), isso indica que o algoritmo proposto obteve melhor resultado. A coluna *Atingiu EDA* indica o número de instâncias daquele conjunto em que o algoritmo conseguiu atingir o estado arte.

[1] Foi utilizado o melhor resultado para cada instância e depois calculado o resultado médio para aquele conjunto.

Tabela 7 – Comparação do desempenho do algoritmo ALNS-SA com *benchmark* para o conjunto C1

Instância	Melhor	Média	Desvio padrão	EDA	Diferença	RPD %	Atingiu EDA
C101-025	191.30	191.30	0.00	191.30	0.00	0.00	Sim
C102-025	190.30	190.30	0.00	190.30	0.00	0.00	Sim
C103-025	190.30	190.30	0.00	190.30	0.00	0.00	Sim
C104-025	190.30	190.30	0.00	186.90	3.40	1.82	Não
C105-025	191.30	191.30	0.00	191.30	0.00	0.00	Sim
C106-025	191.30	191.30	0.00	191.30	0.00	0.00	Sim
C107-025	191.30	191.30	0.00	191.30	0.00	0.00	Sim
C108-025	191.30	191.30	0.00	191.30	0.00	0.00	Sim
C109-025	191.30	191.30	0.00	191.30	0.00	0.00	Sim
C101-050	415.40	415.40	0.00	362.40	53.00	14.62	Não
C102-050	387.50	387.50	0.00	361.40	26.10	7.22	Não
C103-050	383.80	383.80	0.00	361.40	22.40	6.20	Não
C104-050	382.40	383.40	0.69	357.25	25.15	7.04	Não
C105-050	395.90	395.90	0.00	362.40	33.50	9.24	Não
C106-050	406.30	406.30	0.00	362.40	43.90	12.11	Não
C107-050	395.90	395.90	0.00	362.40	33.50	9.24	Não
C108-050	389.90	389.90	0.00	362.40	27.50	7.59	Não
C109-050	388.20	388.23	0.13	362.40	25.80	7.12	Não
C101-100	915.70	916.72	1.27	827.30	88.40	10.69	Não
C102-100	900.00	904.12	0.97	827.30	72.70	8.79	Não
C103-100	878.50	879.23	1.84	826.30	52.20	6.32	Não
C104-100	867.50	871.50	3.17	822.90	44.60	5.42	Não
C105-100	903.80	904.22	0.26	827.30	76.50	9.25	Não
C106-100	910.70	912.74	0.88	827.30	83.40	10.08	Não
C107-100	900.90	901.40	0.17	827.30	73.60	8.90	Não
C108-100	886.10	886.24	0.28	827.30	58.80	7.11	Não
C109-100	880.20	880.46	0.26	825.64	54.56	6.61	Não

Tabela 8 – Comparação do desempenho do algoritmo ALNS-SA com *benchmark* para o conjunto C2

Instância	Melhor	Média	Desvio padrão	EDA	Diferença	RPD %	Atingiu EDA
C201-025	214.70	214.70	0.00	214.70	0.00	0.00	Sim
C202-025	214.70	214.70	0.00	214.70	0.00	0.00	Sim
C203-025	214.70	216.00	2.96	214.70	0.00	0.00	Sim
C204-025	213.10	213.50	0.69	213.10	0.00	0.00	Sim
C205-025	214.70	214.70	0.00	214.70	0.00	0.00	Sim
C206-025	214.70	215.27	2.05	214.70	0.00	0.00	Sim
C207-025	214.50	214.50	0.00	214.50	0.00	0.00	Sim
C208-025	214.50	214.50	0.00	214.50	0.00	0.00	Sim
C201-050	360.20	360.20	0.00	360.20	0.00	0.00	Sim
C202-050	360.20	360.20	0.00	360.20	0.00	0.00	Sim
C203-050	359.80	359.80	0.00	359.80	0.00	0.00	Sim
C204-050	350.10	350.10	0.00	350.10	0.00	0.00	Sim
C205-050	359.80	359.80	0.00	359.80	0.00	0.00	Sim
C206-050	359.80	359.80	0.00	359.80	0.00	0.00	Sim
C207-050	359.60	359.60	0.00	359.60	0.00	0.00	Sim
C208-050	350.50	350.50	0.00	350.50	0.00	0.00	Sim
C201-100	589.10	589.10	0.00	589.10	0.00	0.00	Sim
C202-100	589.10	589.10	0.00	589.10	0.00	0.00	Sim
C203-100	588.70	591.43	3.81	588.70	0.00	0.00	Sim
C204-100	591.60	595.91	2.78	588.10	3.50	0.60	Não
C205-100	586.40	586.40	0.00	586.40	0.00	0.00	Sim
C206-100	586.00	586.00	0.00	586.00	0.00	0.00	Sim
C207-100	585.80	585.80	0.00	585.80	0.00	0.00	Sim
C208-100	585.80	585.80	0.00	585.80	0.00	0.00	Sim

Tabela 9 – Comparação do desempenho do algoritmo ALNS-SA com *benchmark* para o conjunto R1

Instância	Melhor	Média	Desvio padrão	EDA	Diferença	RPD %	Atingiu EDA
R101-025	617.10	617.10	0.00	617.10	0.00	0.00	Sim
R102-025	547.10	547.12	0.13	546.33	0.77	0.14	Não
R103-025	454.60	454.60	0.00	454.60	0.00	0.00	Sim
R104-025	416.90	416.90	0.00	416.90	0.00	0.00	Sim
R105-025	530.50	530.50	0.00	530.50	0.00	0.00	Sim
R106-025	465.40	465.40	0.00	457.30	8.10	1.77	Não
R107-025	424.30	424.30	0.00	422.93	1.38	0.33	Não
R108-025	397.30	397.30	0.00	396.14	1.16	0.29	Não
R109-025	441.30	441.30	0.00	441.30	0.00	0.00	Sim
R110-025	444.10	444.12	0.11	437.30	6.80	1.55	Não
R111-025	428.80	428.80	0.00	423.79	5.01	1.18	Não
R112-025	393.00	393.48	0.52	384.20	8.80	2.29	Não
R101-050	1044.00	1046.29	1.25	1043.37	0.63	0.06	Não
R102-050	909.00	913.24	3.60	909.00	0.00	0.00	Sim
R103-050	772.90	775.82	2.48	765.95	6.95	0.91	Não
R104-050	626.40	637.77	6.15	616.50	9.90	1.61	Não
R105-050	902.60	905.79	2.36	892.12	10.48	1.17	Não
R106-050	793.00	793.50	1.06	791.37	1.63	0.21	Não
R107-050	711.10	722.70	6.68	704.44	6.66	0.95	Não
R108-050	617.90	627.55	5.62	617.70	0.20	0.03	Não
R109-050	786.80	793.02	4.50	775.10	11.70	1.51	Não
R110-050	697.00	716.76	7.68	692.58	4.42	0.64	Não
R111-050	707.20	712.44	3.69	707.20	0.00	0.00	Sim
R112-050	637.70	642.83	3.13	630.20	7.50	1.19	Não
R101-100	1647.00	1663.53	9.10	1631.15	15.85	0.97	Não
R102-100	1469.10	1485.24	8.69	1466.60	2.50	0.17	Não
R103-100	1212.60	1241.25	16.27	1208.70	3.90	0.32	Não
R104-100	986.20	1012.91	17.24	971.50	14.70	1.51	Não
R105-100	1358.90	1384.79	16.22	1346.14	12.76	0.95	Não
R106-100	1238.10	1260.00	16.43	1234.60	3.50	0.28	Não
R107-100	1066.90	1110.20	18.89	1064.60	2.30	0.22	Não
R108-100	944.70	982.49	17.08	944.00	0.70	0.07	Não
R109-100	1158.40	1186.58	15.94	1146.90	11.50	1.00	Não
R110-100	1094.50	1120.11	15.89	1068.00	26.50	2.48	Não
R111-100	1054.20	1081.71	14.23	1048.70	5.50	0.52	Não
R112-100	961.60	999.80	20.61	960.00	1.60	0.17	Não

Tabela 10 – Comparação do desempenho do algoritmo ALNS-SA com *benchmark* para o conjunto R2

Instância	Melhor	Média	Desvio padrão	EDA	Diferença	RPD %	Atingiu EDA
R201-025	463.30	465.97	3.69	463.30	0.00	0.00	Sim
R202-025	410.50	413.55	3.27	410.50	0.00	0.00	Sim
R203-025	391.40	391.40	0.00	391.40	0.00	0.00	Sim
R204-025	355.00	355.60	0.77	355.00	0.00	0.00	Sim
R205-025	393.00	393.92	1.68	393.00	0.00	0.00	Sim
R206-025	374.40	376.63	1.81	374.40	0.00	0.00	Sim
R207-025	361.60	363.00	1.92	361.60	0.00	0.00	Sim
R208-025	328.20	328.20	0.00	328.20	0.00	0.00	Sim
R209-025	370.70	371.74	0.53	370.70	0.00	0.00	Sim
R210-025	404.60	406.81	2.47	404.60	0.00	0.00	Sim
R211-025	350.90	351.36	2.21	350.90	0.00	0.00	Sim
R201-050	799.70	810.46	4.51	791.90	7.80	0.98	Não
R202-050	712.40	720.18	3.99	698.50	13.90	1.99	Não
R203-050	620.50	628.86	6.00	605.30	15.20	2.51	Não
R204-050	506.40	506.40	0.00	506.40	0.00	0.00	Sim
R205-050	695.70	717.41	7.76	690.10	5.60	0.81	Não
R206-050	638.60	658.47	5.64	632.40	6.20	0.98	Não
R207-050	582.60	587.32	2.40	500.00	82.60	16.52	Não
R208-050	487.70	491.87	2.86	485.00	2.70	0.56	Não
R209-050	605.80	629.14	9.29	600.60	5.20	0.87	Não
R210-050	654.00	661.31	4.63	645.60	8.40	1.30	Não
R211-050	554.10	558.69	6.11	535.50	18.60	3.47	Não
R201-100	1193.10	1215.16	11.71	1143.20	49.90	4.36	Não
R202-100	1051.70	1078.48	12.20	1000.00	51.70	5.17	Não
R203-100	894.40	917.69	10.89	850.00	44.40	5.22	Não
R204-100	738.50	769.88	11.08	700.00	38.50	5.50	Não
R205-100	964.60	1000.46	19.21	950.00	14.60	1.54	Não
R206-100	881.80	922.14	17.35	850.00	31.80	3.74	Não
R207-100	807.30	838.26	14.59	750.00	57.30	7.64	Não
R208-100	708.20	727.85	7.27	680.00	28.20	4.15	Não
R209-100	871.00	897.63	14.34	850.00	21.00	2.47	Não
R210-100	932.80	950.91	10.24	900.00	32.80	3.64	Não
R211-100	786.30	804.77	8.52	750.00	36.30	4.84	Não

Tabela 11 – Comparação do desempenho do algoritmo ALNS-SA com *benchmark* para o conjunto RC1

Instância	Melhor	Média	Desvio padrão	EDA	Diferença	RPD %	Atingiu EDA
RC101-025	461.10	461.10	0.00	406.62	54.48	13.40	Não
RC102-025	351.80	351.80	0.00	351.80	0.00	0.00	Sim
RC103-025	332.80	332.80	0.00	332.05	0.75	0.23	Não
RC104-025	306.60	306.60	0.00	305.82	0.78	0.25	Não
RC105-025	411.30	411.61	0.39	410.95	0.35	0.09	Não
RC106-025	345.50	345.61	0.28	342.83	2.67	0.78	Não
RC107-025	298.30	298.30	0.00	298.30	0.00	0.00	Sim
RC108-025	294.50	294.50	0.00	293.79	0.71	0.24	Não
RC101-050	944.00	957.02	4.50	850.02	93.98	11.06	Não
RC102-050	867.80	868.16	0.50	719.90	147.90	20.54	Não
RC103-050	791.50	800.24	4.57	643.13	148.37	23.07	Não
RC104-050	687.60	688.79	0.84	541.80	145.80	26.91	Não
RC105-050	855.30	857.88	2.09	754.44	100.86	13.37	Não
RC106-050	838.40	843.25	2.12	664.43	173.97	26.18	Não
RC107-050	745.10	747.96	1.65	591.48	153.62	25.97	Não
RC108-050	704.40	709.68	2.51	538.96	165.44	30.70	Não
RC101-100	1643.20	1679.93	15.65	1584.09	59.11	3.73	Não
RC102-100	1476.60	1500.50	10.16	1457.40	19.20	1.32	Não
RC103-100	1310.30	1345.99	19.32	1258.00	52.30	4.16	Não
RC104-100	1161.40	1190.89	13.49	1000.00	161.40	16.14	Não
RC105-100	1521.80	1566.19	18.43	1471.16	50.64	3.44	Não
RC106-100	1381.20	1413.60	18.48	1000.00	381.20	38.12	Não
RC107-100	1241.60	1287.57	22.40	1207.80	33.80	2.80	Não
RC108-100	1159.40	1180.78	17.09	1114.20	45.20	4.06	Não

Tabela 12 – Comparação do desempenho do algoritmo ALNS-SA com *benchmark* para o conjunto RC2

Instância	Melhor	Média	Desvio padrão	EDA	Diferença	RPD %	Atingiu EDA
RC201-025	360.20	360.20	0.00	360.20	0.00	0.00	Sim
RC202-025	338.00	338.00	0.00	338.00	0.00	0.00	Sim
RC203-025	326.90	328.47	5.29	326.90	0.00	0.00	Sim
RC204-025	299.70	309.71	5.66	299.70	0.00	0.00	Sim
RC205-025	338.00	338.00	0.00	338.00	0.00	0.00	Sim
RC206-025	324.00	324.00	0.00	324.00	0.00	0.00	Sim
RC207-025	298.30	299.85	3.46	298.30	0.00	0.00	Sim
RC208-025	269.10	271.84	7.18	269.10	0.00	0.00	Sim
RC201-050	684.80	685.04	0.41	684.80	0.00	0.00	Sim
RC202-050	613.60	617.14	2.57	613.60	0.00	0.00	Sim
RC203-050	557.90	567.25	6.65	490.12	67.78	13.83	Não
RC204-050	444.20	451.46	11.22	444.20	0.00	0.00	Sim
RC205-050	631.00	634.71	3.57	630.00	1.00	0.16	Não
RC206-050	612.20	621.32	3.58	610.00	2.20	0.36	Não
RC207-050	559.50	567.56	6.31	558.60	0.90	0.16	Não
RC208-050	499.40	515.76	7.77	490.00	9.40	1.92	Não
RC201-100	1277.10	1319.97	19.30	1261.80	15.30	1.21	Não
RC202-100	1109.30	1154.60	25.18	1092.30	17.00	1.56	Não
RC203-100	958.30	989.56	13.71	950.00	8.30	0.87	Não
RC204-100	795.90	829.66	12.03	780.00	15.90	2.04	Não
RC205-100	1164.90	1211.87	22.95	1154.00	10.90	0.94	Não
RC206-100	1088.10	1103.91	10.13	1000.00	88.10	8.81	Não
RC207-100	980.70	1022.17	13.88	970.00	10.70	1.10	Não
RC208-100	823.90	847.22	13.25	800.00	23.90	2.99	Não

C RESULTADOS POR INSTÂNCIA NO SEGUNDO EXPERIMENTO

Neste apêndice é apresentado, instância por instância, dos resultados obtidos pelo experimento que realiza a comparação entre uma execução do algoritmo com função objetivo que considera apenas o custo e outra execução do algoritmo que considera o custo e a recompensa em sua função objetivo. Mais detalhes podem ser obtidos na Seção 4.3.

Os resultados são apresentados nas Tabelas 13, 14, 15, 16, 17 e 18. Cada uma delas expõe o resultado do experimento de comparação para um conjunto de instâncias. Ao todo existem 6 conjuntos de instâncias: c1, c2, r1, r2, rc1 e rc2. Dentro de cada conjunto de instâncias existem instâncias com 25, 50 e 100 requisições. Os conjuntos c1 e c2 são *clusterizados*, ou seja, as requisições ficam agrupadas, próximas entre si. Os conjuntos r1 e r2 são randomizados, ou seja, as requisição não necessariamente ficam próximas umas das outras. Os conjuntos rc1 e rc2 são randômico-*clusterizados*, portanto, algumas requisições estão agrupadas enquanto outras não.

Nas tabelas mencionadas, a coluna *Instância* indica a instâncias. A coluna C_1 indica o valor do custo quando a função objetivo apenas considera o custo. A coluna C_2 indica o valor do custo quando a função objetivo considera o custo e a recompensa não coletada. As colunas R_1 e R_2 indicam os valores da recompensa não coletada para quando a função objetivo considera apenas o custo e para quando a função objetivo considera o custo e a recompensa não coletada, respectivamente. Importante frisar que o valor de R_1 são os valores de recompensa das requisições não atendidas na solução do problema quando a função objetivo considera apenas o custo. A coluna *DC* indica a diferença do custo entre C_2 e C_1 e a coluna *DR* indica a diferença entre as colunas R_2 e R_1. Todas as colunas apresentam a média do valor da melhor solução para as vinte execuções da instância. O valor na coluna *RPDC*, dado pela Equação 4.3, é obtido pelo calculo considerando o valor do custo para as duas versões da função objetivo. A Equação 4.4 considera o valor da recompensa não coletada para as duas versões da função objetivo para gerar o valor que está presente na coluna *RPDR*. As colunas *% ReqC* e *% ReqR* indicam a proporção de requisições utilizadas na solução pela função objetivo apenas considerando o custo e pela função objetivo considerando o custo e recompensa não coletada, respectivamente.

Tabela 13 – Comparação dos resultados para ambas funções objetivo para o conjunto C1

Instância	C_1	C_2	R_1	R_2	DC	DR	RPDC (%)	RPDR (%)	% ReqC	% ReqR
C101-025	95.10	157.52	117.90	33.90	62.41	-84.00	0.52	0.78	0.66	-0.71
C101-050	234.12	260.81	107.10	67.12	26.69	-39.97	0.64	0.79	0.11	-0.37
C101-100	688.20	699.66	93.75	73.05	11.47	-20.70	0.78	0.82	0.02	-0.22
C102-025	117.81	162.33	107.10	35.25	44.53	-71.85	0.57	0.80	0.38	-0.67
C102-050	283.78	297.79	86.40	63.52	14.01	-22.88	0.69	0.78	0.05	-0.26
C102-100	742.34	785.12	75.60	56.17	42.78	-19.42	0.81	0.85	0.06	-0.26
C103-025	138.63	163.84	81.67	29.25	25.21	-52.42	0.60	0.85	0.18	-0.64
C103-050	289.12	306.68	84.00	48.90	17.56	-35.10	0.68	0.81	0.06	-0.42
C103-100	808.80	822.72	55.20	39.98	13.92	-15.23	0.85	0.89	0.02	-0.28
C104-025	142.13	162.22	80.70	29.10	20.10	-51.60	0.55	0.83	0.14	-0.64
C104-050	307.73	324.22	62.33	37.35	16.49	-24.98	0.77	0.85	0.05	-0.40
C104-100	846.59	830.41	36.30	27.60	-16.18	-8.70	0.89	0.92	-0.02	-0.24
C105-025	124.70	159.86	105.90	33.45	35.16	-72.45	0.46	0.81	0.28	-0.68
C105-050	249.65	258.94	87.97	73.95	9.29	-14.02	0.70	0.76	0.04	-0.16
C105-100	744.78	761.24	71.40	57.45	16.46	-13.95	0.82	0.85	0.02	-0.20
C106-025	96.12	158.01	116.78	34.58	61.89	-82.20	0.52	0.79	0.64	-0.70
C106-050	240.39	258.83	99.83	68.55	18.44	-31.28	0.65	0.78	0.08	-0.31
C106-100	760.91	762.22	65.33	55.80	1.30	-9.53	0.83	0.85	0.00	-0.15
C107-025	130.62	159.21	97.80	33.15	28.58	-64.65	0.54	0.80	0.22	-0.66
C107-050	278.65	288.33	75.22	56.25	9.68	-18.97	0.75	0.82	0.03	-0.25
C107-100	811.83	795.28	49.35	44.10	-16.54	-5.25	0.87	0.89	-0.02	-0.11
C108-025	135.79	162.46	88.80	30.60	26.67	-58.20	0.56	0.83	0.20	-0.66
C108-050	294.30	297.42	71.92	54.30	3.11	-17.62	0.74	0.81	0.01	-0.25
C108-100	834.03	830.86	44.85	41.10	-3.17	-3.75	0.88	0.88	-0.00	-0.08
C109-025	148.41	166.08	73.28	28.88	17.67	-44.40	0.63	0.85	0.12	-0.61
C109-050	320.91	324.11	55.50	43.12	3.20	-12.38	0.78	0.83	0.01	-0.22
C109-100	860.61	857.42	32.55	29.48	-3.19	-3.07	0.91	0.92	-0.00	-0.09

Tabela 14 – Comparação dos resultados para ambas funções objetivo para o conjunto C2

Instância	C_1	C_2	R_1	R_2	DC	DR	RPDC (%)	RPDR (%)	% ReqC	% ReqR
C201-025	78.15	177.49	152.85	35.17	99.34	-117.67	0.28	0.79	1.27	-0.77
C201-050	362.66	380.40	42.15	16.65	17.74	-25.50	0.84	0.94	0.05	-0.60
C201-100	386.49	452.59	228.38	204.53	66.11	-23.85	0.39	0.45	0.17	-0.10
C202-025	178.04	202.37	54.52	13.43	24.33	-41.10	0.74	0.93	0.14	-0.75
C202-050	332.18	337.09	26.18	19.27	4.91	-6.90	0.90	0.93	0.01	-0.26
C202-100	516.96	540.78	191.55	165.82	23.82	-25.73	0.47	0.54	0.05	-0.13
C203-025	169.57	197.00	68.17	21.15	27.43	-47.02	0.66	0.90	0.16	-0.69
C203-050	339.34	343.77	19.35	14.32	4.43	-5.03	0.93	0.95	0.01	-0.26
C203-100	565.95	566.58	157.65	135.53	0.63	-22.12	0.55	0.61	0.00	-0.14
C204-025	167.77	183.33	60.00	18.90	15.56	-41.10	0.70	0.91	0.09	-0.69
C204-050	244.48	248.05	148.80	111.60	3.56	-37.20	0.42	0.56	0.01	-0.25
C204-100	578.64	590.77	138.75	122.25	12.13	-16.50	0.60	0.64	0.02	-0.12
C205-025	157.47	200.62	98.78	23.77	43.16	-75.00	0.53	0.86	0.27	-0.76
C205-050	389.81	398.56	27.38	13.12	8.75	-14.25	0.89	0.95	0.02	-0.52
C205-100	507.60	509.42	187.28	174.15	1.81	-13.12	0.49	0.52	0.00	-0.07
C206-025	139.74	209.70	134.47	23.77	69.96	-110.70	0.37	0.86	0.50	-0.82
C206-050	408.33	406.16	0.00	0.60	-2.17	0.60	1.00	1.00	-0.01	-
C206-100	523.17	556.72	175.43	144.38	33.55	-31.05	0.51	0.60	0.06	-0.18
C207-025	143.70	195.02	110.92	22.50	51.32	-88.42	0.48	0.87	0.36	-0.80
C207-050	391.01	394.17	1.57	1.80	3.17	0.23	0.99	0.99	0.01	0.14
C207-100	518.80	546.66	166.12	145.72	27.86	-20.40	0.53	0.59	0.05	-0.12
C208-025	175.28	223.86	78.75	4.65	48.58	-74.10	0.65	0.98	0.28	-0.94
C208-050	181.96	257.02	183.45	105.45	75.06	-78.00	0.29	0.59	0.41	-0.43
C208-100	542.05	586.46	150.68	135.90	44.40	-14.78	0.58	0.61	0.08	-0.10

Tabela 15 – Comparação dos resultados para ambas funções objetivo para o conjunto R1

Instância	C_1	C_2	R_1	R_2	DC	DR	RPDC (%)	RPDR (%)	% ReqC	% ReqR
R101-025	391.13	393.67	77.92	71.85	2.54	-6.08	0.66	0.69	0.01	-0.08
R101-050	730.68	746.22	89.25	80.10	15.54	-9.15	0.70	0.73	0.02	-0.10
R101-100	1269.55	1275.77	87.45	81.45	6.22	-6.00	0.80	0.81	0.00	-0.07
R102-025	336.81	350.25	74.70	55.35	13.44	-19.35	0.69	0.75	0.04	-0.26
R102-050	696.14	700.17	61.12	53.55	4.03	-7.58	0.80	0.82	0.01	-0.12
R102-100	1259.39	1260.62	53.17	51.23	1.24	-1.95	0.88	0.88	0.00	-0.04
R103-025	311.00	313.03	58.35	52.73	2.03	-5.62	0.75	0.76	0.01	-0.10
R103-050	592.81	592.05	57.67	54.00	-0.76	-3.67	0.81	0.82	-0.00	-0.06
R103-100	983.09	994.19	59.17	55.88	11.10	-3.30	0.86	0.87	0.01	-0.06
R104-025	257.38	267.12	82.65	68.78	9.74	-13.88	0.63	0.69	0.04	-0.17
R104-050	394.89	397.29	102.75	94.58	2.40	-8.17	0.61	0.65	0.01	-0.08
R104-100	839.48	844.88	67.65	59.85	5.41	-7.80	0.83	0.85	0.01	-0.12
R105-025	288.99	295.04	100.28	91.35	6.06	-8.93	0.57	0.59	0.02	-0.09
R105-050	604.81	606.86	99.00	89.03	2.05	-9.97	0.66	0.70	0.00	-0.10
R105-100	1095.13	1081.73	81.00	77.47	-13.40	-3.53	0.81	0.82	-0.01	-0.04
R106-025	319.15	324.60	68.40	59.33	5.45	-9.08	0.71	0.75	0.02	-0.13
R106-050	531.43	536.00	89.10	82.50	4.57	-6.60	0.69	0.72	0.01	-0.07
R106-100	971.31	979.29	86.03	77.33	7.97	-8.70	0.80	0.82	0.01	-0.10
R107-025	256.64	262.34	82.50	73.28	5.69	-9.22	0.63	0.67	0.02	-0.11
R107-050	473.34	480.74	88.72	81.83	7.39	-6.90	0.68	0.71	0.02	-0.08
R107-100	818.82	823.57	85.12	77.17	4.75	-7.95	0.80	0.82	0.01	-0.09
R108-025	273.51	275.31	70.58	63.90	1.80	-6.68	0.69	0.71	0.01	-0.09
R108-050	397.24	411.29	100.88	86.40	14.05	-14.47	0.61	0.69	0.04	-0.14
R108-100	839.75	847.77	54.83	47.70	8.02	-7.12	0.87	0.88	0.01	-0.13
R109-025	323.30	328.21	66.90	55.20	4.91	-11.70	0.71	0.75	0.02	-0.17
R109-050	553.20	566.62	91.88	77.40	13.42	-14.47	0.69	0.73	0.02	-0.16
R109-100	951.88	951.54	79.95	72.67	-0.35	-7.28	0.81	0.83	-0.00	-0.09
R110-025	344.07	357.22	55.20	35.62	13.15	-19.58	0.76	0.84	0.04	-0.35
R110-050	475.64	481.63	95.70	79.58	5.99	-16.12	0.65	0.70	0.01	-0.17
R110-100	806.73	806.24	111.30	103.50	-0.49	-7.80	0.73	0.75	-0.00	-0.07
R111-025	254.47	262.22	88.95	79.35	7.76	-9.60	0.60	0.64	0.03	-0.11
R111-050	469.48	472.61	92.03	83.62	3.13	-8.40	0.69	0.71	0.01	-0.09
R111-100	811.27	817.69	91.72	84.22	6.42	-7.50	0.78	0.79	0.01	-0.08
R112-025	273.39	286.33	82.28	54.00	12.94	-28.28	0.64	0.77	0.05	-0.34
R112-050	390.79	404.87	117.30	95.33	14.08	-21.97	0.58	0.65	0.04	-0.19
R112-100	805.90	818.95	83.33	69.08	13.05	-14.25	0.81	0.84	0.02	-0.17

Tabela 16 – Comparação dos resultados para ambas funções objetivo para o conjunto R2

Instância	C_1	C_2	R_1	R_2	DC	DR	RPDC (%)	RPDR (%)	% ReqC	% ReqR
R201-025	474.63	475.14	0.00	0.00	0.51	0.00	1.00	1.00	0.00	-
R201-050	819.34	819.78	0.00	0.00	0.44	0.00	1.00	1.00	0.00	-
R201-100	1199.21	1199.22	33.90	25.73	0.01	-8.17	0.92	0.94	0.00	-0.24
R202-025	456.48	456.18	0.00	0.00	-0.31	0.00	1.00	1.00	-0.00	-
R202-050	742.22	742.16	0.00	0.00	-0.06	0.00	1.00	1.00	-0.00	-
R202-100	1073.95	1072.86	19.12	14.03	-1.09	-5.10	0.96	0.97	-0.00	-0.27
R203-025	400.10	399.50	0.00	0.00	-0.60	0.00	1.00	1.00	-0.00	-
R203-050	601.85	608.17	30.75	22.57	6.32	-8.18	0.89	0.92	0.01	-0.27
R203-100	864.59	878.30	34.65	25.80	13.71	-8.85	0.91	0.94	0.02	-0.26
R204-025	241.82	277.40	100.50	50.62	35.58	-49.88	0.52	0.77	0.15	-0.50
R204-050	239.25	321.60	193.43	115.95	82.35	-77.48	0.27	0.56	0.34	-0.40
R204-100	734.64	753.18	17.02	13.65	18.54	-3.37	0.96	0.96	0.03	-0.20
R205-025	404.84	404.84	0.00	0.00	0.00	0.00	1.00	1.00	0.00	-
R205-050	697.14	705.20	25.27	19.95	8.06	-5.32	0.92	0.93	0.01	-0.21
R205-100	1019.29	1033.53	36.67	20.02	14.24	-16.65	0.91	0.95	0.01	-0.45
R206-025	288.14	322.75	79.50	37.27	34.61	-42.23	0.62	0.82	0.12	-0.53
R206-050	279.73	264.78	195.07	182.85	-14.95	-12.22	0.27	0.32	-0.05	-0.06
R206-100	726.67	769.99	130.20	88.58	43.31	-41.62	0.66	0.78	0.06	-0.32
R207-025	286.07	319.93	75.75	39.08	33.85	-36.67	0.64	0.81	0.12	-0.48
R207-050	278.31	338.80	184.28	140.40	60.49	-43.88	0.30	0.47	0.22	-0.24
R207-100	759.88	754.80	76.58	53.92	-5.09	-22.65	0.80	0.86	-0.01	-0.30
R208-025	294.88	301.99	36.75	18.82	7.10	-17.93	0.81	0.90	0.02	-0.49
R208-050	337.40	387.80	138.75	70.72	50.40	-68.03	0.47	0.73	0.15	-0.49
R208-100	698.47	705.09	22.20	13.57	6.62	-8.62	0.94	0.97	0.01	-0.39
R209-025	203.56	267.56	118.42	63.52	64.01	-54.90	0.43	0.69	0.31	-0.46
R209-050	607.25	616.08	27.38	17.48	8.83	-9.90	0.90	0.93	0.01	-0.36
R209-100	863.11	870.38	41.10	33.98	7.27	-7.12	0.89	0.91	0.01	-0.17
R210-025	172.15	208.76	142.95	112.50	36.61	-30.45	0.32	0.47	0.21	-0.21
R210-050	614.65	624.12	28.57	19.57	9.46	-9.00	0.90	0.93	0.02	-0.31
R210-100	908.36	922.63	44.48	30.52	14.28	-13.95	0.89	0.92	0.02	-0.31
R211-025	296.16	320.21	58.27	26.48	24.05	-31.80	0.73	0.88	0.08	-0.55
R211-050	306.97	328.08	165.22	145.72	21.11	-19.50	0.39	0.45	0.07	-0.12
R211-100	749.21	767.23	101.47	67.42	18.01	-34.05	0.72	0.82	0.02	-0.34

Tabela 17 – Comparação dos resultados para ambas funções objetivo para o conjunto RC1

Instância	C_1	C_2	R_1	R_2	DC	DR	RPDC (%)	RPDR (%)	% ReqC	% ReqR
RC101-025	249.22	291.65	120.38	62.62	42.43	-57.75	0.42	0.70	0.17	-0.48
RC101-050	562.58	576.42	108.30	94.50	13.83	-13.80	0.59	0.65	0.02	-0.13
RC101-100	1356.67	1358.78	81.38	69.90	2.10	-11.47	0.81	0.83	0.00	-0.14
RC102-025	164.85	191.51	132.90	89.17	26.66	-43.73	0.36	0.57	0.16	-0.33
RC102-050	582.10	602.16	83.47	57.52	20.07	-25.95	0.67	0.77	0.03	-0.31
RC102-100	1250.37	1274.81	60.52	47.77	24.43	-12.75	0.85	0.88	0.02	-0.21
RC103-025	177.56	203.25	123.67	82.50	25.69	-41.17	0.40	0.60	0.14	-0.33
RC103-050	481.73	500.69	92.62	64.58	18.95	-28.05	0.64	0.74	0.04	-0.30
RC103-100	1012.83	1023.16	88.65	73.12	10.33	-15.53	0.79	0.82	0.01	-0.18
RC104-025	179.73	196.61	109.65	79.58	16.88	-30.08	0.47	0.61	0.09	-0.27
RC104-050	365.36	382.10	112.42	82.72	16.74	-29.70	0.58	0.70	0.05	-0.26
RC104-100	926.21	935.48	77.10	66.00	9.27	-11.10	0.82	0.84	0.01	-0.14
RC105-025	248.66	296.03	112.35	48.67	47.37	-63.67	0.46	0.76	0.19	-0.57
RC105-050	568.30	588.88	93.60	71.33	20.58	-22.27	0.64	0.72	0.04	-0.24
RC105-100	1220.98	1247.14	75.83	64.65	26.16	-11.17	0.81	0.84	0.02	-0.15
RC106-025	168.61	192.32	135.15	93.00	23.71	-42.15	0.35	0.55	0.14	-0.31
RC106-050	574.45	602.58	79.42	47.92	28.13	-31.50	0.71	0.82	0.05	-0.40
RC106-100	1089.72	1101.05	96.00	79.42	11.34	-16.58	0.77	0.81	0.01	-0.17
RC107-025	161.21	195.95	123.38	69.08	34.74	-54.30	0.41	0.67	0.22	-0.44
RC107-050	459.75	468.44	81.08	71.78	8.69	-9.30	0.70	0.74	0.02	-0.11
RC107-100	1002.54	1016.90	93.30	70.65	14.37	-22.65	0.78	0.82	0.01	-0.24
RC108-025	175.63	193.99	108.60	69.15	18.36	-39.45	0.48	0.67	0.10	-0.36
RC108-050	360.46	380.43	117.08	82.80	19.96	-34.28	0.56	0.70	0.06	-0.29
RC108-100	907.54	915.66	105.00	83.62	8.12	-21.38	0.75	0.79	0.01	-0.20

Tabela 18 – Comparação dos resultados para ambas funções objetivo para o conjunto RC2

Instância	C_1	C_2	R_1	R_2	DC	DR	RPDC (%)	RPDR (%)	% ReqC	% ReqR
RC201-025	431.30	431.30	0.00	0.00	0.00	0.00	1.00	1.00	0.00	-
RC201-050	713.71	713.56	0.00	0.00	-0.14	0.00	1.00	1.00	-0.00	-
RC201-100	1371.85	1393.26	1.35	1.50	21.41	0.15	1.00	1.00	0.02	0.11
RC202-025	375.16	375.34	0.00	0.00	0.18	0.00	1.00	1.00	0.00	-
RC202-050	640.60	640.56	0.00	0.00	-0.03	0.00	1.00	1.00	-0.00	-
RC202-100	1196.74	1208.97	20.77	15.30	12.23	-5.47	0.95	0.96	0.01	-0.26
RC203-025	357.75	360.37	0.00	0.00	2.61	0.00	1.00	1.00	0.01	-
RC203-050	586.85	585.99	0.00	0.00	-0.87	0.00	1.00	1.00	-0.00	-
RC203-100	930.37	930.62	48.08	42.23	0.25	-5.85	0.87	0.89	0.00	-0.12
RC204-025	269.75	286.45	47.70	23.55	16.70	-24.15	0.77	0.89	0.06	-0.51
RC204-050	481.43	482.18	5.85	4.35	0.75	-1.50	0.98	0.98	0.00	-0.26
RC204-100	672.88	723.71	120.30	70.95	50.83	-49.35	0.68	0.82	0.08	-0.41
RC205-025	370.89	364.76	18.30	17.02	-6.12	-1.28	0.91	0.92	-0.02	-0.07
RC205-050	656.39	656.05	0.45	0.23	-0.35	-0.23	1.00	1.00	-0.00	-0.50
RC205-100	1207.41	1212.63	45.38	40.73	5.22	-4.65	0.89	0.90	0.00	-0.10
RC206-025	345.34	346.65	0.00	0.00	1.31	0.00	1.00	1.00	0.00	-
RC206-050	662.82	662.98	0.00	0.00	0.15	0.00	1.00	1.00	0.00	-
RC206-100	1133.94	1128.05	1.27	2.92	-5.89	1.65	1.00	0.99	-0.01	1.29
RC207-025	169.28	262.87	134.78	56.10	93.59	-78.68	0.34	0.72	0.55	-0.58
RC207-050	586.64	592.21	0.00	0.00	5.57	0.00	1.00	1.00	0.01	-
RC207-100	1044.51	1051.50	10.50	4.05	6.99	-6.45	0.97	0.99	0.01	-0.61
RC208-025	287.09	306.56	43.88	0.45	19.47	-43.42	0.79	1.00	0.07	-0.99
RC208-050	332.83	355.77	170.40	138.38	22.94	-32.03	0.33	0.45	0.07	-0.19
RC208-100	859.99	868.34	11.32	7.95	8.35	-3.37	0.97	0.98	0.01	-0.30