UNIVERSIDADE FEDERAL FLUMINENSE

Elias Lawrence Marques Júnior

Metaheurísticas para o Problema Multi-objetivo de Roteamento Verde de Drones em Grid

NITERÓI

2020

UNIVERSIDADE FEDERAL FLUMINENSE

Elias Lawrence Marques Júnior

Metaheurísticas para o Problema Multi-objetivo de Roteamento Verde de Drones em Grid

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização

Orientador:
Luiz Satoru Ochi

NITERÓI

2020

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

M357m Marques júnior, Elias Lawrence
Metaheurísticas para o Problema Multi-objetivo de Roteamento Verde de Drones em Grid / Elias Lawrence Marques júnior ; Luiz Satoru Ochi, orientador. Niterói, 2020.
63 f. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2020.

DOI: http://dx.doi.org/10.22409/PGC.2020.m.14730110781

1. Veículo Aéreo Não Tripulado. 2. Otimização Multi-objetivo. 3. VNS. 4. BRKGA. 5. Produção intelectual. I. Ochi, Luiz Satoru, orientador. II. Universidade Federal Fluminense. Instituto de Computação. III. Título.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

Elias Lawrence Marques Júnior

Metaheurísticas para o Problema Multi-objetivo de Roteamento Verde de Drones em Grid

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização

Aprovada em Fevereiro de 2020.

BANCA EXAMINADORA

Prof. Luiz Satoru Ochi - Orientador, UFF

Prof. Igor Machado Coelho, UFF

Prof. Nelson Maculan, UFRJ

Prof. Paulo Fernando Ferreira Rosa, IME

Dr. Vitor Nazário Coelho, OptBlocks Consultoria

Prof. Yuri Abitbol De Menezes Frota, UFF

Niterói

2020

Distância entre sonho e realidade é um passo.

Agradecimentos

Elias L. Marques Jr. gostaria de agradecer à agência brasileira FAPERJ. Este estudo foi financiado em parte pelo Subsistema de Pessoal de Área Superior - Brasil (CAPES) - Codigo Financeiro 001.

Resumo

Este trabalho trata do roteamento de veículos não tripulados (Unmaned Aerial Vehicles - UAV) em cenários de grade dinâmica com autonomia limitada de bateria e múltiplas estações de carregamento. O problema é inspirado por restrições do mundo real, especialmente projetadas para superar desafios de uma faixa limitada de direção de veículos. Recentemente, esses tipos de veículos começaram a ser usados em diversas aplicações incluindo a entrega e coleta de produtos (objeto de estudo deste trabalho), exigindo especialistas em diversas áreas do conhecimento para gerenciar essa nova logística. Inspirado por uma visão multicritério de sistemas reais, consideramos diferentes funções objetivo introduzidas na literatura. Uma variante multi-objetivo da Busca de Vizinhança Variável (VNS) é considerada para encontrar um conjunto de soluções não dominadas, respeitando a navegação sobre espaço aéreo restrito e também a autonomia do voo. Um caso de estudo foi desenvolvido onde UAV's têm que atender clientes espalhados por uma grade representando um mapa. Os drones começam em um determinado ponto do mapa (grid) com uma carga de bateria, onde o grid é composto por quatro diferentes tipos de pontos: um regular e três especiais (representando áreas proibidas, pontos de recarga e clientes). Qualquer sequência de pontos adjacentes válidos forma uma rota, mas como isso produz um grande número de combinações, uma técnica de pré-processamento é proposta para pré-calcular as distâncias em um determinado cenário dinâmico. Resultados computacionais demonstram o desempenho de diferentes variantes dos algoritmos propostos.

Palavras-chave: Veículo Aéreo Não Tripulado, Microgrids, Otimização Multi-objetivo, VNS, BRKGA.

Abstract

This paper deals with Unmanned Aerial Vehicle (UAV) routing in dynamic grid scenarios with limited battery autonomy and multiple charging stations. The problem is inspired by real-world constraints, specially designed for overcoming challenges of a limited vehicle driving range. Recently, these kinds of vehicles have started to be used in several applications including delivering and collecting products (object of study of this work), requiring experts in several knowledge fields to manage this novel logistics. Inspired by a multi-criteria view of real systems, we consider different objective functions introduced in the literature. A multi-objective variant of Variable Neighborhood Search is considered for finding a set of non-dominated solutions, while respecting the navigation over forbidden areas and also flight autonomy. A case of study was developed where UAVs have to attend clients spread throughout a grid representing a map. The drones start in a given grid point with a given battery charge, where the grid is composed by four different kinds of points: a regular one and three special (prohibited, recharge and client delivery). Any sequence of valid adjacent points forms a route, but since this yields a huge number of combinations, a preprocessing technique is proposed to pre-compute distances in a given dynamic scenario. Computational results demonstrate the performance of different variants of the proposed algorithms.

Keywords: Unmanned Aerial Vehicle, Microgrids, Multi-objective optimization, Variable Neighborhood Search, Biased Random Key Gentic Algorithm.

Lista de Figuras

2.1 Exemplo de instância e solução para o PMORVDG 5

3.1 Pré-aplicação da vizinhança *Swap*(1,1) inter-rota 17

3.2 Vizinhança *Swap*(1,1) inter-rota 17

3.3 Pré-aplicação da vizinhança *Shift*(1,0) inter-rota 18

3.4 Vizinhança *Shift*(1,0) inter-rota 18

3.5 Vizinhança *Swap* 19

3.6 Vizinhança *Remove Ponto de Recarga* 19

3.7 Vizinhança *Ponto de Recarga mais Próximo* 20

3.8 Vizinhança *Remove Repetido* 20

3.9 Vizinhança *Aumento da Velocidade da Seção* 20

3.10 *Recharge Random Increase* neighborhood 21

3.11 Codificação com chaves aleatórias 24

3.12 Decodificação por ordenação com chaves aleatórias 25

3.13 Decodificação das chaves aleatórias representando velocidade e taxa de recarga 27

4.1 Variação dos valores dos objetivos para instância eil51b2 em decorrência da duração da metaheurística: BRKGA (azul), GMOVND (preto) e GVND (vermelho) 42

4.2 Interface gráfica 47

Lista de Tabelas

4.1 Amostra eil51. 31

4.2 Amostra eil101. 32

4.3 Amostra eil51P5R1a1. 33

4.4 Amostra eil51P5R1a2. 33

4.5 Amostra eil51P5R1b1. 34

4.6 Amostra eil51P5R1b2. 34

4.7 Amostra eil101P5R1a1. 35

4.8 Amostra eil101P5R1a2. 35

4.9 Amostra eil101P5R1b1. 36

4.10 Amostra eil101P5R1b2. 36

4.11 Instância eil51 com 2 drones. 37

4.12 Instância eil101 com 2 drones. 38

4.13 Comparação do pré-processamento em instâncias da família eil51. 39

4.14 Comparação do pré-processamento em instâncias da família eil101. 40

4.15 Instância eil51 com 2 drones. 41

4.16 Comparação dos valores das funções objetivo nas instâncias padrões. 43

4.17 Comparação dos valores das funções objetivo nas instâncias sem pré-processamento. 43

4.18 Comparação dos valores das funções objetivo nas instâncias com 2 drones. 44

4.19 Comparação dos valores das funções objetivo nas instâncias eil51 com 2 drones e c_v igual a 0,1 (300s, 120s, 60s, 30s, 10s e 5s). 44

4.20 Comparação dos valores das funções objetivo nas instâncias eil101 com 2 drones e c_v igual a 0,1 (600s, 300s, 120s, 60s, 30s e 10s). 45

4.21 Comparação dos valores de hipervolume nas instâncias padrões. 45

4.22 Comparação dos valores de hipervolume nas instâncias sem pré-processamento. 46

4.23 Comparação dos valores de hipervolume nas instâncias com 2 drones. . . . 46

4.24 Comparação dos valores de hipervolume nas instâncias eil51 com 2 drones e c_v igual a 0,1 (300s, 120s, 60s, 30s, 10s e 5s). 46

Sumário

1 Introdução **1**

2 O Problema Multi-Objetivo de Roteamento Verde de Drones em Grid **4**

2.1 MILP 5

2.1.1 Conjuntos, parâmetros, variáveis auxiliares e de decisão 5

2.1.2 Objetivos a serem otimizados 7

2.1.3 Restrições e requisitos operacionais do modelo 8

2.2 Restrições adicionais considerando a autonomia dos veículos e estações de recarga 10

2.2.1 Conjuntos, parâmetros, variáveis auxiliares e de decisão 10

2.2.2 Restrições do modelo e requisitos operacionais das baterias 11

2.3 Revisão da Literatura 12

3 Metodologia **13**

3.1 Algoritmo A* 13

3.2 G-VND 15

3.2.1 Construção 16

3.2.2 Busca Local 16

3.2.2.1 Vizinhanças Inter-rotas 17

3.2.2.2 Vizinhanças Intrarrotas 18

3.2.2.3 Pareto 22

3.2.2.4 Critério de Aceitação 22

3.2.2.5 Implementação do VND 23

3.2.3 BRKGA 24

3.2.3.1 Implementação 26

4 Experimentos Computacionais **28**

4.1 Multi-objetivo 30

4.1.1 Medidas de Comparação 30

4.1.2 Resultados Computacionais 30

4.2 Mono Objetivo 38

4.3 Interface Gráfica 47

5 Conclusões **48**

Referências **50**

Capítulo 1

Introdução

Inovações tecnológicas como a miniaturização de sistemas de controle eletrônico e a redução de custos de componentes eletrônicos [10] resultaram em um aumento na disponibilidade de veículos não tripulados (UV), também conhecidos como drones, ou sistemas não tripulados (US) .

Embora o drone seja frequentemente relacionado a indústrias amadores, de entretenimento e fotográficas, seus usos se espalharam para aplicações militares, civis e comerciais. Vigilância aérea, reconhecimento e rastreamento de objetos são algumas das muitas outras aplicações que estão surgindo com potencial para o uso do UV. Inúmeros outros podem surgir da criatividade humana no futuro próximo [4]. Já existem alguns trabalhos que mostram aplicações no dia a dia:

- Inspeção de infraestrutura [19], [18], [16]: O UV pode seguir um caminho predeterminado ou pode mover-se por *servoing visual* e detecta por meio de tratamento de imagem o tamanho e localização de defeitos e rachaduras. O uso de um drone para inspeção nos dará várias vantagens em relação ao método de inspeção tradicional:
  - Reduzindo o risco de acidente de trabalho;
  - Redução orçamentária: menos logística e menos horas de trabalho;
  - Operações menos invasivas: a ponte não será fechada para o tráfego enquanto a inspeção estiver concluída.
- Inspeção de Linha de Energia [1], [7]:
  - A inspeção aérea de linhas de transmissão de energia elétrica é normalmente realizada usando helicópteros pilotados por humanos, que é um procedimento

caro e propenso a acidentes, trazendo riscos à vida dos seres humanos. Nesse sentido, o drone é uma solução de baixo custo com vários benefícios potenciais.

- Vigilância de um espaço-alvo usando veículos aéreos é um tópico de interesse de pesquisa atual para aplicações como monitoramento do clima, pesquisas geográficas e, talvez, exploração extraterrestre [21];
- A grande flexibilidade dos UVs pode permitir novas abordagens durante a coleta de dados de sensoriamento remoto, que por exemplo integra mapeamento em tempo real e navegação autônoma [14];
- O monitoramento ambiental é o amplo campo de pesquisa para soluções UV individuais, onde o monitoramento do ambiente é realizado apenas por um veículo [15].

O setor de transporte de carga, em particular, já está mostrando algum interesse e investimentos em aplicações de UV. O crescimento do comércio eletrônico sustentou esse interesse de grandes empresas. Os drones de transporte são capazes de decolar e pousar com segurança nas proximidades de edifícios e humanos, melhorando a qualidade do serviço atual em áreas congestionadas ou remotas [10].

Quando o serviço de entrega é discutido, estamos, implicitamente, falando sobre um Problema do Caixeiro Viajante (TSP) e suas variações, como o Problema de Roteamento de Veículos, por exemplo. O que brevemente significa um problema de projetar rotas ótimas de um ou vários depósitos para um número de cidades geograficamente dispersas, clientes ou pontos estratégicos, sujeitos a restrições laterais [17]. Embora existam muitos trabalhos na literatura relacionados às variações do TSP [13], os que se aproximam do roteamento UV ainda são poucos.

No entanto, não podemos nos concentrar apenas nos avanços tecnológicos e simplesmente esquecer os danos ao ambiente que eles podem causar. É por isso que a comunidade científica tem se preocupado tanto em desenvolver tecnologias verdes e isso não difere na computação. O Green Vehicle Routing Problem (G-VRP), por exemplo, proposto por Erdoğan et al. [9], adiciona ao VRP original, restrições sobre a economia de combustível e/ou a escolha do combustível menos danoso ao meio ambiente.

O TSP com seleção de hotel [26] é uma variação do TSP com semelhanças com o problema abordado neste trabalho. O objetivo principal é minimizar o número de viagens e tempo total de viagem. Esse problema é encontrado em cenários reais, como a entrega de produtos por veículos elétricos que precisam ser recarregados ao longo de um passeio.

Para abordar sistemas reais, muitas vezes é necessário projetar um problema multi-objetivo fornecendo um conjunto de soluções não dominadas (pool de soluções) com diferentes rotas e horários possíveis. Quanto mais funções objetivo e restrições conter, o modelo tende a ser mais semelhante ao mundo real e mais complexo.

As principais contribuições deste trabalho atual são:

- Proposta de novas heurísticas para um problema de roteamento UV dependente do tempo, considerando as seguintes restrições:
  - respeitando os requisitos operacionais dos UVs;
  - abordando o microespaço aéreo considerando um cenário de inspeção de pontos e evitando pontos proibidos (restrições de atracação) [6];
  - integrando os UVs aos novos conceitos de sistemas mini/microgrid, nos quais os veículos podem ser carregados em diferentes pontos das futuras cidades inteligentes;
  - rotas dinâmicas considerando drones já em movimento: instâncias com bateria inicial diferente de 100 %, ponto de origem aleatório e um número de clientes já visitados.

O restante desta dissertação está organizado da seguinte forma. O capítulo 2 descreve o modelo proposto e o intervalo de parâmetros reais considerados em nossas análises, enquanto o capítulo 3 contém a metodologia empregada para resolver o problema. No capítulo 4, são descritos os experimentos computacionais comparando as diferentes implementações, instâncias, variáveis e resultados. Finalmente, o capítulo 5 conclui o trabalho e apresenta futuras direções de pesquisa.

Capítulo 2

O Problema Multi-Objetivo de Roteamento Verde de Drones em Grid

O caso de estudo do Problema Multi-Objetivo de Roteamento Verde de Drones em Grid (PMORVDG), projetado aqui, é composto de um espaço aéreo dividido em faixas horizontais e verticais, onde o drone pode se mover seguindo a distância de Chebyshev, onde as distâncias entre quaisquer pontos adjacentes são as mesmas. Essa distância foi utilizada, pois simplifica os cálculos sem perda de generalidade e sem prejuízo para a construção das rotas. Estações de energia estão espalhadas na área de roteamento e acessadas pelo drone para recarregar suas baterias. Para representar áreas proibidas, a grade também é composta por pontos proibidos que o UAV não pode acessar, caso contrário, isso invalida a rota. Um exemplo de instância para o problema pode ser visto na Figura 2.1.

Como um problema de roteamento, o veículo deve atender a clientes que estão espalhados pela rede. Para isso, o ponto correspondente ao cliente deve fazer parte da rota final. Isso significa que as coordenadas X e Y do cliente devem fazer parte da matriz que representa a solução. No Capítulo 3, discutimos uma alternativa para pré-processar distâncias mais curtas e armazená-las em estrutura de dados auxiliares, de modo que apenas os pontos de origem e destino precisem ser considerados entre os pontos de carregamento e entrega. No entanto, devido à natureza dinâmica do problema, pode ser necessário processá-los novamente, após ocorrerem alterações nos dados de entrada [1].

[1] O trabalho atual considera que os dados dinâmicos são passados como entrada, para que nenhuma alteração precise ser realizada durante a pesquisa. Como as instâncias já consideram a localização e a capacidade inicial do drone arbitrário (carga da bateria), uma variante dependente do tempo pode ser considerada como uma extensão desse trabalho (consulte o Capítulo 5)

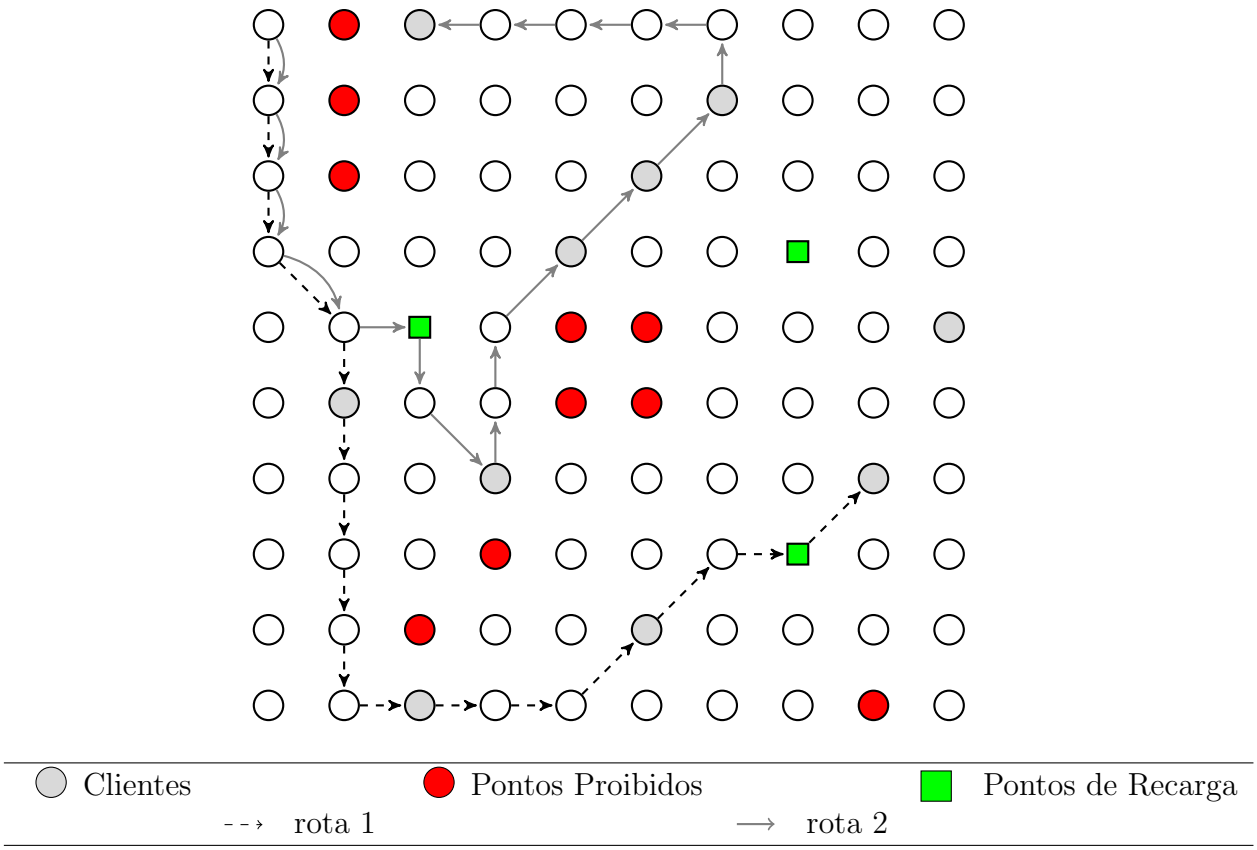

Figura 2.1: Exemplo de instância e solução para o PMORVDG

2.1 MILP

2.1.1 Conjuntos, parâmetros, variáveis auxiliares e de decisão

Os parâmetros seguintes foram definidos e considerados para o modelo:

T: conjunto de intervalos de tempo discretizados $\{1, \ldots, t_{max}\}$;

$TimeUnit$: unidade de tempo fixa (hour) em qualquer instante entre $t \in T$ a $t+1 \in T$;

Z_x: comprimento (em km) do mapa a ser considerado pelo modelo;

Z_y: largura (em km) do mapa a ser considerado pelo modelo;

C: conjunto de clientes, contendo as seguintes informações para cada cliente c:

$C_{x,c}$: coordenada x para $c \in C$;

$C_{y,c}$: coordenada y para $c \in C$;

P: conjunto de pontos do mapa por onde o drone não pode passar.

$P_{x,p}$: coordenada x para $p \in P$;

$P_{y,p}$: coordenada y para $p \in P$;

U: conjunto de UAVs disponíveis de tamanho pré-determinado na instância;

I_x: coordenada x dos UAVs no início da rota ($t = 1$);

I_y: coordenada y dos UAVs no início da rota ($t = 1$);

V_{max} : velocidade máxima dos UAVs (km/h);

As seguintes variáveis de decisão usadas para prover uma solução completa para o modelo proposto.

$uC_{u,c,t}$: variável binária que indica se o cliente c foi visitado pelo UAV u no tempo t.

vC_c: variável binária que indica se o cliente c foi visitado.

$uP^x_{u,p,t}$: variável binária que indica se o UAV u passou pela mesma coordenada x do ponto proibido p no tempo t.

$uP^y_{u,p,t}$: variável binária que indica se o UAV u passou pela mesma coordenada y do ponto proibido p no tempo t.

$pos^x_{u,t}$: coordenada x do UAV u no tempo t;

$pos^y_{u,t}$ coordenada y do UAV u no tempo t;

$v_{u,t}$: velocidade do UAV u no tempo t (começando no intervalo $t - 1$ até $t \in T$);

Algumas variáveis auxiliares são necessárias para checar requisitos operacionais, restrições ou auxiliar no cálculo de uma métrica desejada:

t^u_{max}: intervalo final da rota de cada veículo;

2.1.2 Objetivos a serem otimizados

Como mencionado anteriormente, este trabalho aborda um problema multi-objetivo, no qual queremos encontrar um conjunto de soluções que retornem o melhor resultado possível para cada função objetiva. Em um problema real de roteamento de UAV, uma grande quantidade de variáveis deve ser considerada para obter a solução mais adequada. No entanto, para transformar este problema real em um problema computável, mapeamos três objetivos principais que resumem, de maneira satisfatória, como o sistema real deve se comportar.

$finalCharge$ (O1): é interessante finalizar a rota com a taxa máxima de carga possível, garantindo que o drone esteja preparado para uma rota futura, uma vez que o trabalho atual possui uma natureza dinâmica onde as instâncias já consideram a localização inicial e capacidade do drone arbitrário. O que significa que um drone pode iniciar uma nova rota no final de uma antiga.

time (O2): a rota total deve ser realizada no menor tempo possível.

cons (O3): é desejável que durante a rota, o veículo consuma o mínimo possível de bateria/combustível.

clients: número de clientes visitados.

Como pode ser visto, o algoritmo proposto neste artigo se concentra em encontrar um equilíbrio entre soluções, já que um elemento pode afetar outro. Quanto menor o tempo, maior a velocidade, maior o consumo. Maior carga final significa mais tempo gasto em recarga/abastecimento, o que resulta em tempos maiores.

O tempo final da rota é calculado na Eq. (2.1).

$$time = max(\sum_{t \in T} \frac{1}{v_{u,t}} + \sum_{t \in T} \sum_{e \in E} charge_{u,e,t} RC_{u,e,t} DOR \qquad \forall u \in U) \tag{2.1}$$

As Eqs. (2.2) e (2.3) medem a quantidade de energia que os veículos apresentam ao final da rota.

$$t^u_{max} \geq \sum_{t \in T} uC_{u,c,t} \cdot t \qquad \forall u \in U \tag{2.2}$$

$$finalCharge \leq \sum_{u \in U} batRate_{u,t^u_{max}} \tag{2.3}$$

A Eq. (2.4) mede o número de clientes visitados.

$$clients = \sum_{c \in C} vC_c \tag{2.4}$$

Finalmente, Eq. (2.5) mede o consumo de todos veículos. O esclarecimento das variáveis é apresentada na Seção 2.2.2.

$$cons = \sum_{t \in T} TimeUnit(\frac{VEVv_{u,t}}{V_{max}} + on_{u,t}FEV_u) \qquad \forall u \in U \tag{2.5}$$

2.1.3 Restrições e requisitos operacionais do modelo

A posição inicial dos UAVs é atualiizada nas Eqs. (2.6) e (2.7).

$$pos^x_{u,1} = I_x \qquad \forall u \in U \tag{2.6}$$

$$pos^y_{u,1} = I_y \qquad \forall u \in U \tag{2.7}$$

As Eqs. (2.8) e (2.9) garantem que os drones só podem se movimentar para pontos adjacentes a cada turno.

$$|pos^x_{u,t} - pos^x_{u,t-1}| \leq 1 \qquad \forall u \in U, c \in C, t \in T : t \geq 1 \tag{2.8}$$

$$|pos^y_{u,t} - pos^y_{u,t-1}| \leq 1 \qquad \forall u \in U, c \in C, t \in T : t \geq 1 \tag{2.9}$$

Da Eq. (2.10) a Eq. (2.13) mostra que um cliente é visitado se um UAV passa por suas coordenadas.

$$pos^x_{u,t} - C_{x,c} \leq Z_x(1 - uC_{u,c,t}) \qquad \forall u \in U, c \in C, t \in T \tag{2.10}$$

$$-pos^x_{u,t} + C_{x,c} \leq Z_x(1 - uC_{u,c,t}) \qquad \forall u \in U, c \in C, t \in T \tag{2.11}$$

$$pos^{y}_{u,t} - C_{y,c} \leq Z_y(1 - uC_{u,c,t}) \qquad \forall u \in U, c \in C, t \in T \tag{2.12}$$

$$-pos^{y}_{u,t} + C_{y,c} \leq Z_y(1 - uC_{u,c,t}) \qquad \forall u \in U, c \in C, t \in T \tag{2.13}$$

Um cliente é visitado se qualquer drone o visita a qualquer momento, como demonstrado na Eq. (2.14).

$$vC_c \geq uC_{u,c,t} \qquad \forall u \in U, c \in C, t \in T \tag{2.14}$$

As Eqs. (2.15) a (2.19) dizem respeito a proibição da passagem de drones em pontos informados como proibidos (docking constraint [6]). Na vida real, há áreas onde os drones não têm permissão para acessar ou atravessar. Esta situação foi representada por pontos especiais espalhados pela rede. Se a rota contiver esses pontos, a solução será inválida.

$$pos^{x}_{u,t} - P_{x,c} \leq Z_x(1 - uP^{x}_{u,p,t}) \qquad \forall u \in U, p \in P, t \in T \tag{2.15}$$

$$-pos^{x}_{u,t} + P_{x,c} \leq Z_x(1 - uP^{x}_{u,p,t}) \qquad \forall u \in U, p \in P, t \in T \tag{2.16}$$

$$pos^{y}_{u,t} - P_{y,c} \leq Z_y(1 - uP^{y}_{u,p,t}) \qquad \forall u \in U, p \in P, t \in T \tag{2.17}$$

$$-pos^{y}_{u,t} + P_{y,c} \leq Z_y(1 - uP^{y}_{u,p,t}) \qquad \forall u \in U, p \in P, t \in T \tag{2.18}$$

$$uP^{x}_{u,p,t} + uP^{y}_{u,p,t} \leq 1 \qquad \forall u \in U, p \in P, t \in T \tag{2.19}$$

Finalmente, Eq. (2.20) limita a velocidade dos UAVs ao máximo definido.

$$V_{max} \geq v_{u,t} \qquad \forall u \in U, t \in T : t \geq 2 \tag{2.20}$$

2.2 Restrições adicionais considerando a autonomia dos veículos e estações de recarga

2.2.1 Conjuntos, parâmetros, variáveis auxiliares e de decisão

O conjunto de estações de recarga E tem apenas 2 parâmetros, os quais são as suas coordenadas:

E: conjunto de estações de recarga e:

CS_e^x: coordenada x da estação de recarga de energia;

CS_e^y : coordenada y da estação de recarga de energia;

Cinco parâmetros adicionais foram incluídos:

DOR : duração do processo de recarga (horas) por carga;

VEV : consumo varável de energia (%) relacionado a velocidade do drone (por hora);

FEV : consumo fixo (%) em qualquer instante entre $t \in T$ a $t + 1 \in T$;

$RC_{u,e,t}$: taxa de recarga do UAV u na estação e por hora (%);

BI: porcentagem da bateria do drone no início (%);

A principal decisão variável a respeito do problema de atualizar a taxa da bateria é feita pela variável $charge_{u,e,t}$, que se trata de uma variável binária que indica se o UAV u está carregando em e no turno t:

$charge_{u,e,t}$: UAV u está carregando em e no turno t.

Além disso, três variáveis auxiliares adicionais foram necessárias para atualizar a taxa da bateria do UAV em cada intervalo t, assim como checar se os UAVs estão funcionando em cada tempo t.

$batRate_{u,t}$: UAV battery rate at time t, ≥ 0 and ≤ 100;

$on_{u,t}$: UAV is running, turned on, at time t, binary;

2.2.2 Restrições do modelo e requisitos operacionais das baterias

Restrições relacionadas às estações de recarga dos UAVs e os requisitos para o valor da taxa de recarga são listados abaixo. Em particular, as Eqs. (2.21) e (2.22), são inspiradas pelo trabalho de Coelho et al. [5], que focam em atualizar a carga da bateria de Plug-in Electric Vehicles. Eq. (2.21) define a carga inicial da bateria dos veículos, enquanto a Eq. (2.22) atualiza a taxa de bateria a cada turno, aumento-a caso tenha sido recarregada e diminuindo-a devido sua velocidade e consumo fixo.

$$batRate_{u,1} = BI \tag{2.21}$$

A velocidade no trecho influencia não apenas o tempo total da rota, mas também o consumo, uma vez maior a velocidade (v), maior será o consumo. O nível de combustível/bateria no final do trecho é um resultado do nível de combustível/bateria no início do trecho diminuído pelo consumo fixo (c_f) e a velocidade multiplicada pelo coeficiente de consumo variável como mostrado na Eq. (2.22).

O tempo na estação de energia também influencia. Se o veículo gastar mais tempo, ele pode acumular mais combustível/energia para sua bateria. O nível de combustível/bateria no final do trecho é resultado do nível de combustível/bateria no início do alongamento, aumentado pela quantidade de combustível/energia recarregada.

$$\begin{aligned} batRate_{u,t} = batRate_{u,t-1} - TimeUnit(\frac{VEVv_{u,t}}{V_{max}} - on_{u,t}FEV_u + \\ \sum_{e \in E} charge_{u,e,t}RC_{u,e,t}) \qquad \forall u \in U, t \geq 2 \in T \end{aligned} \tag{2.22}$$

Como pode ser verifcado na Eq. (2.23), o UAV é considerado ligado quando se movimenta.

$$on_{u,t} * M = v_{u,t} \qquad \forall u \in U, t \in T \tag{2.23}$$

2.3 Revisão da Literatura

O PMORVDG foi proposto por Coelho et al. [4], onde um modelo de programação matemática é utilizado para resolver problemas-teste criados pelos autores.

O problema deriva do TSPD (Problema do Caixeiro Viajante com Drones) [2] e do PRVD (Problema do Roteamento de Veículos com Drones) [28]. Embora ambos os problemas abordem a visita de clientes por meio de drones, ao contrário do PMORVDG, nesses trabalhos, os drones são utilizados como veículos complementares a um principal (como por exemplo um caminhão). Além disso, esses problemas são representados através de grafos, o que se diferencia da representação em grid abordada nesse trabalho.

Como o PRVD é uma extensão do PRV ao trabalhar com drones concomitantemente a veículos terrestres nas rotas; logo, trata-se também de um problema de otimização NP-difícil. Consequentemente, o uso exclusivo de uma formulação MILP para obter ótimos em um período de tempo razoável só é possível em instâncias de pequeno porte. Portanto, para abordar instâncias de grande escala do PRVD, alguns trabalhos propõe uma metaheurística baseada no Variable Neighborhood Search (VNS), como em Schermer et al. [24].

Capítulo 3

Metodologia

Nesta dissertação, três metaheurísticas foram propostas. Uma metaheurística GRASP com uma busca local VND multi-objetivo (MOVND), denominada G-MOVND, uma variante mono-objetivo (G-VND) e uma metaheurística BRKGA.

3.1 Algoritmo A*

Quando trabalhamos com problemas de roteamento em grafo, apenas a ordem de visitação já é suficiente para montar a solução, uma vez que a instância já fornece uma matriz de distâncias entre quaisquer dois pontos. Todavia, problemas em grid, não possuem essa informação extra e, portanto, deve-se gerar o caminho entre cada dois nós para podermos ter uma solução completa. Para nossa metaheurística, utilizamos o algoritmo A*.

O algoritmo foi publicado pela primeira vez em 1968 por Peter Hart, Nils Nilsson e Bertram Raphael, Stanford Research Institute (hoje SRI International) [22]. Ele pode ser visto como uma extensão do algoritmo de Edsger Dijkstra, de 1959.

A* é um algoritmo de busca que visa encontrar um caminho com o menor custo (menor distância percorrida, menor tempo etc.) entre dois nós. Isso é feito mantendo uma árvore de caminhos originados no nó inicial e estendendo esses caminhos uma aresta por vez até que seu critério de finalização seja atendido.

A cada iteração de seu loop principal do algoritmo A*, determina-se qual de seus caminhos deve-se estender. Essa escolha é feita com base no custo do caminho e uma estimativa do custo necessário para estender o caminho até a meta. Especificamente, A* seleciona o caminho que minimiza $f(n)$.

$$f(n) = g(n) + h(n) \tag{3.1}$$

Na equação 3.1, n é o próximo nó no caminho, $g(n)$ é o custo do caminho desde o nó inicial até n, e $h(n)$ é uma função heurística que estima o custo do melhor caminho de n até o objetivo. A* termina quando o caminho alcança o nó objetivo ou quando não há mais caminhos elegíveis para estender. A função heurística é específica ao problema a que se deseja aplicar o algoritmo. Se a função heurística não superestimar o valor real ao objetivo, o algoritmo A* retornará garantidamente o caminho menos custoso entre o nó inicial e o objetivo.

Algoritmo 1 Reconstrói Caminho

```
procedure ReconstroiCaminho(origem, atual)
    caminho ← {atual}
    while atual ∈ origem.Chaves do
        atual ← origem[atual]
        caminho.Insere(atual)
    end while
    retorna caminho
end procedure
```

Algoritmo 2 Busca A*

```
procedure A*(inicio, fim, h)
    openSet ← inicio
    origem ← map vazio
    g ← map com valor padrão de infinito
    g[inicio] ← 0
    f ← map com valor padrão de infinito
    f[inicio] ← h(inicio)
    while openSet ≠ Ø do
        if atual = fim then
            retorna ReconstroiCaminho(origem, atual)
        end if
        openSet.Remove(atual)
        for each vizinho of atual do
            tempG ← g[atual] + d(atual, vizinho)
            if tempG < g[vizinho] then
                origem[vizinho] ← atual
                g[vizinho] ← tempG
                f[vizinho] ← g[vizinho] + h(vizinho)
                if vizinho ∉ openSet then
                    openSet.Insere(vizinho)
                end if
            end if
        end for
    end while
    retorna erro
end procedure
```

Implementações do A* usam uma fila de prioridade para fazer as escolhas repetidamente dos nós de menor custo estimado para poder executar a expansão. Essa fila de

prioridade é conhecida como *open set*. A cada passo do algoritmo, o nó de menor valor $f(x)$ é removido da fila como na linha 12 do algoritmo 2. Os valores f e g de seus vizinhos são atualizados e esses vizinhos são adicionados à fila, como apresentado no loop que começa na linha 13.

Para encontrar a sequência de passos, o algoritmo precisa ser facilmente revisado de forma que cada nó faça referência a seu predecessor, por isso a necessidade do algoritmo 1. Assim, no fim, o nó final referencia seu antecessor, e assim por diante, até o nó inicial.

No PMORVDG, assim como em outros problemas em que se deseja procurar rotas mínimas em um mapa, $h(n)$ deve representar a distância em linha reta (distância euclidiana) de n até o objetivo, uma vez que essa é a menor distância possível entre dois pontos quaisquer.

Na linha 14, $d(atual, vizinho)$ é a distância entre *atual* e *vizinho* e *tempG* é a distância do nó inicial até *vizinho* através do nó *atual*. Se o caminho para *vizinho* é melhor que qualquer caminho anterior, deve-se salvar *atual* em $origem[vizinho]$.

Para cada nó n, $g[n]$ representa o custo do melhor caminho desde *inicio* até n que se conhece até então.

O algoritmo termina de duas formas, caso alcance a linha 25, *openSet* está vazio, mas o objetivo não foi alcançado. Por outro lado, as linhas 9 e 10 representam que o objetivo foi alcançado e então retorna-se o caminho completo.

3.2 G-VND

A metaheurística G-VND pode ser vista como uma metaheurística de multi-partida para problemas de otimização combinatória, em que cada iteração consiste basicamente em duas fases, como mostrado no algoritmo 3: construção e busca local.

Algoritmo 3 G-VND

```
1: procedure G-VND(Vizinhanca)
2:     repeat
3:         E ← {}
4:         s_i ← ConstrutorGRASP()          ▷ solução inicial gerada pelo GRASP
5:         E ← Atualiza(E, s_i)
6:         E ← MOVND(E, Vizinhanca)                          ▷ busca local
7:     until tempo termina
8:     return E
9: end procedure
```

3.2.1 Construção

O GRASP, proposto por Mauricio Resende [23], foi utilizado para encontrar uma solução inicial para o problema abordado neste trabalho. Em uma construção gulosa desse problema, escolhemos iterativamente o cliente mais próximo da posição atual. No entanto, a fase de construção deste algoritmo é um procedimento guloso com componentes aleatórias e adaptativas, o que significa que em vez de escolher sempre o mais próximo, selecionamos um conjunto de clientes k mais próximos da posição atual e, então, escolhemos um cliente aleatoriamente deste conjunto (lista restrita de candidatos) a ser inserida na solução inicial. A velocidade e a taxa de recarga em toda as rotas é setada para o máximo.

Algoritmo 4 Construção

```
1: procedure CONSTRUÇÃO(k)
2:     o ← origem aleatória
3:     s ← s ∪ {o}
4:     Inicializa Lista de Candidatos CL
5:     if o é um cliente then
6:         CL ← CL − {o}
7:     end if
8:     r ← o
9:     while CL ≠ ∅ do
10:        Ordena CL em ordem crescente de acordo com sua distância em relação a r
11:        Atualiza RCL considerando apenas os k melhores candidatos de CL
12:        Escolhe c ∈ RCL aleatoriamente
13:        s ← s ∪ {c}
14:        r ← c
15:        CL ← CL − {r}
16:    end while
17:    return s
18: end procedure
```

3.2.2 Busca Local

O VND (Variable Neighborhood Descent), desenvolvido por N. Mladenović e Pierre Hansen [20], é um método que tem se mostrado ser muito eficiente e capaz de navegar pelo espaço de solução baseado no uso de soluções vizinhas. A heurística de busca local geralmente usa uma estrutura de vizinhança (por exemplo, Hill Climbing), ao contrário da VND que se utiliza de um conjunto de vizinhanças. Essa metaheurística funciona aplicando um método de busca local referente a uma estrutura de vizinhança a uma solução

inicial x. Se a solução x' obtida é melhor que a primeira, nós atribuímos x' a x (x: = x'), e continuamos a busca com a estrutura atual da vizinhança; caso contrário, nós a mudamos.

O VND foi implementado com 10 estruturas de vizinhança. Vizinhanças diferentes afetam diferentes funções objetivo. Em cada iteração, uma solução inicial é gerada pelo GRASP, que agora passa por buscas exaustivas VND/MOVND.

3.2.2.1 Vizinhanças Inter-rotas

- **Swap (1,1)**

Consiste na troca de um cliente i da rota r_1 com um dado cliente j pertencente a rota r_2. A figura 3.2 mostra a aplicação do movimento para os clientes 2 e 5 da solução s representada na figura 3.1.

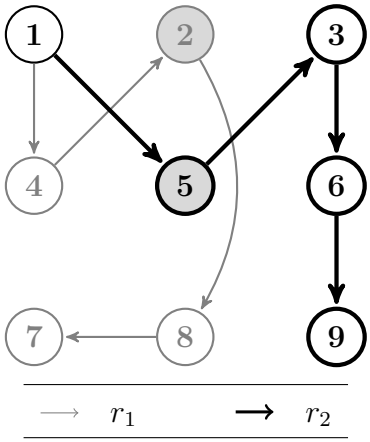

Figura 3.1: Pré-aplicação da vizinhança *Swap*(1,1) inter-rota

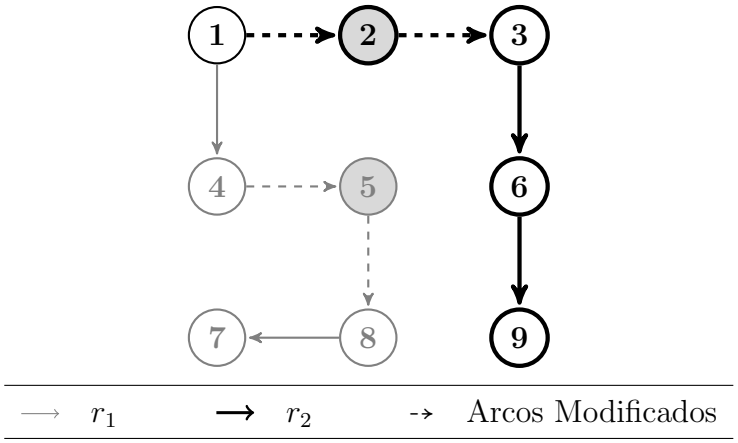

Figura 3.2: Vizinhança *Swap*(1,1) inter-rota

- **Shift (1,0)**

Um cliente i é removido da rota r_1 e inserido na rota r_2. Na figura 3.4, o cliente 6 é removido da rota r_1 e reinserido na rota r_2 entre os clientes 3 e 9 na solução.

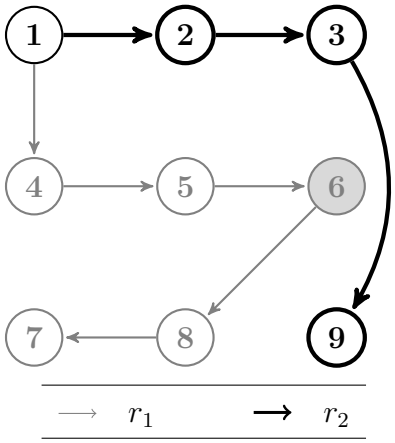

Figura 3.3: Pré-aplicação da vizinhança *Shift*(1,0) inter-rota

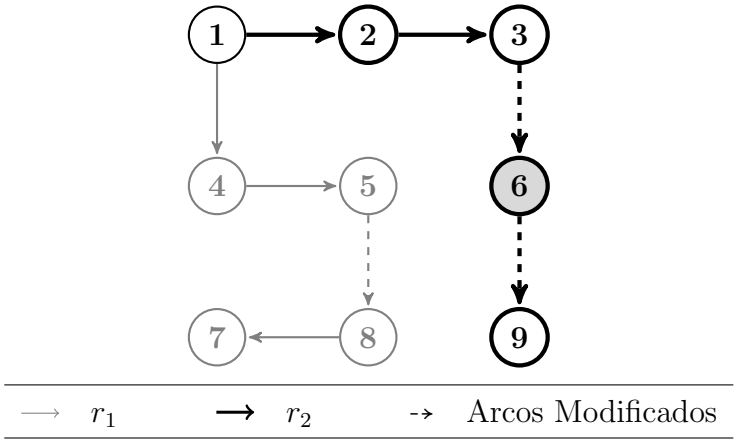

Figura 3.4: Vizinhança *Shift*(1,0) inter-rota

3.2.2.2 Vizinhanças Intrarrotas

- **Swap (1,1)**

Esta vizinhança é responsável por mudar as posições dos clientes na rota. Se a solução gerada não for dominada ou igual a qualquer outra rota no pool atual de soluções, a nova solução será adicionada a ela.

Como esta é a vizinhança mais onerosa, foram implementadas outras versões, de modo que foi possível comparar a que fornece melhores resultados durante os experimentos

computacionais. A diferença das versões era que, em vez de trocar um cliente da rota entre si (S1), a troca só ocorre com os k clientes mais próximos (S2), mais distantes (S3) ou aleatórios (S4).

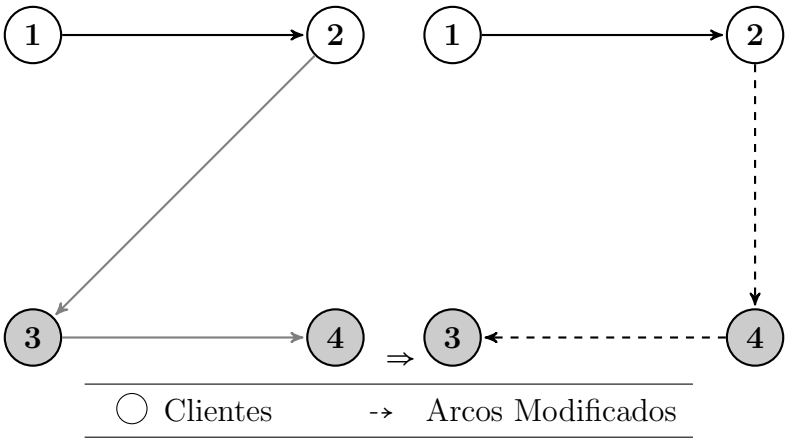

Figura 3.5: Vizinhança *Swap*

- **Remove Ponto de Recarga**

Esta vizinhança é responsável por remover possíveis pontos de recarga em cada subcaminho (caminho entre clientes) e verificar se melhora a solução atual. A ideia é que, removendo esse trecho, o UAV chegaria ao final da rota em menos tempo.

Encontramos um caminho que liga dois clientes ou a origem a um cliente e o remove da rota. Depois, ligamos diretamente os dois clientes. Dessa forma, se houvesse um ponto de recarga nesse subpath, ele seria removido.

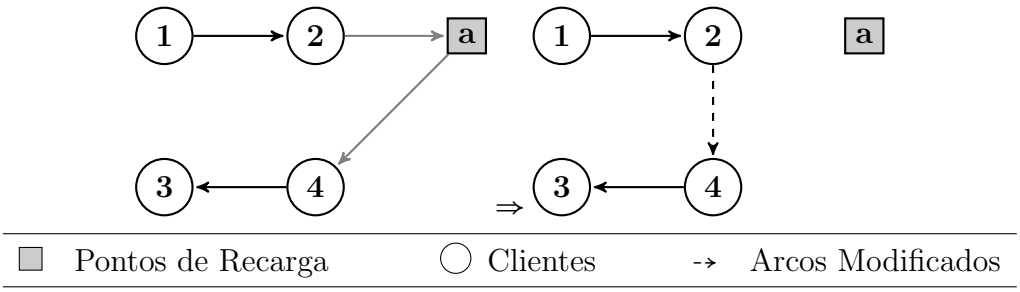

Figura 3.6: Vizinhança *Remove Ponto de Recarga*

- **Ponto de Recarga mais Próximo**

Depois de remover os pontos de recarga desnecessários, tentamos adicionar outros que poderiam melhorar a solução atual.

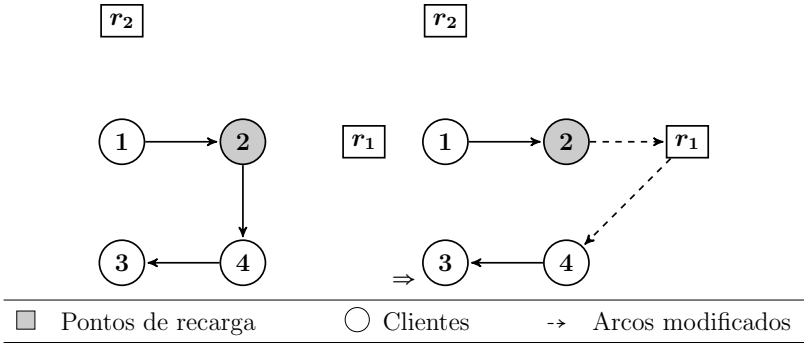

Figura 3.7: Vizinhança *Ponto de Recarga mais Próximo*

- **Remove Repetido**

Depois que todas as operações anteriores foram executadas na rota, poderíamos ter inserido clientes repetidos na rota. A ideia deste bairro é remover os repetidos, tentando reduzir o tamanho da rota e, em conseqüência, reduzir o consumo e/ou o tempo.

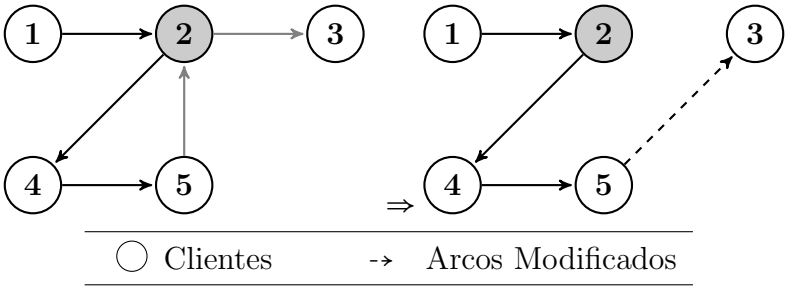

Figura 3.8: Vizinhança *Remove Repetido*

- **Aumento Velocidade da Seção**

Aumenta a velocidade em 1 unidade em cada seção inteira da rota que liga dois pontos importantes (clientes e/ou ponto de recarga).

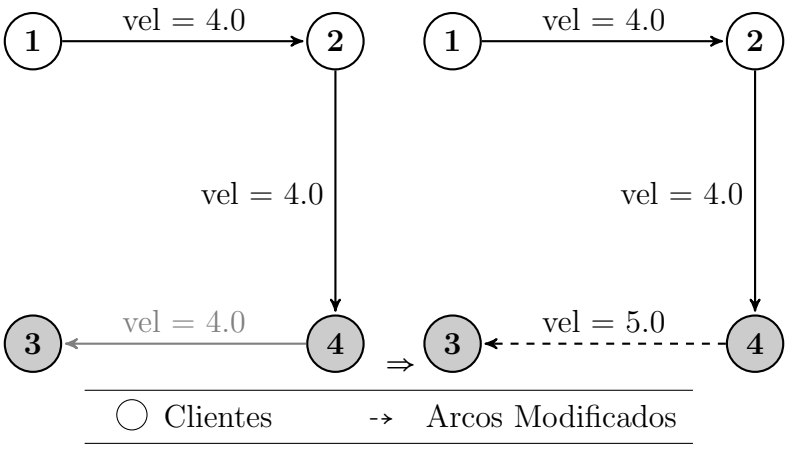

Figura 3.9: Vizinhança *Aumento da Velocidade da Seção*

- **Redução Velocidade da Seção**

Diminui a velocidade em 1 unidade em cada seção inteira da rota que liga dois pontos importantes (clientes e/ou ponto de recarga).

- **Aumento randômico da Velocidade**

Aumenta a velocidade em 1 unidade em cada segmento de um subcaminho escolhido aleatoriamente.

- **Redução randômica da Velocidade**

Diminui a velocidade em 1 unidade em cada segmento de um subcaminho escolhido aleatoriamente.

- **Aumento randômico da Taxa de Recarga**

Aumenta em 1 unidade a porcentagem de carga em cada ponto de recarga de um subcaminho escolhido aleatoriamente.

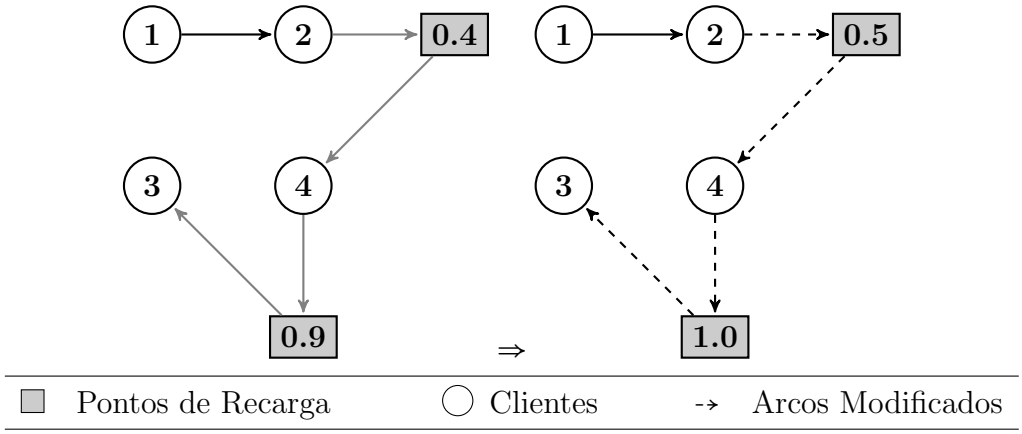

Figura 3.10: *Recharge Random Increase* neighborhood

- **Redução randômica da Taxa de Recarga**

Diminui em 1 unidade a porcentagem de carga em cada ponto de recarga de um subcaminho escolhido aleatoriamente.

3.2.2.3 Pareto

Em problemas de otimização de maximização, dizemos que x domina fracamente y se $x_i \geqslant y_i$ para todas coordenadas i [27]. Em caso de dominância forte, deve-se existir desigualdade estrita para pelo menos uma coordenada. Para problemas de miinimização, x domina fracamente y se $x_i \leqslant y_i$ para todas coordenadas i.

Uma frente de Pareto é considerada um conjunto de soluções não dominadas, ou seja, as soluções não dominam fortemente umas às outras duas a duas.

3.2.2.4 Critério de Aceitação

Depois de gerar um vizinho da solução atual, a rota atual é avaliada e comparada às rotas no pool de soluções. Ele será inserido se não for dominado por nenhum outro presente no pool (conjunto) de soluções não dominadas. Se a nova rota dominar outra, esta será removida do conjunto de soluções.

Neste caso de estudo, limitamos o tamanho do conjunto de soluções não dominadas, limitando, em conseqüência, o número de operações (busca local). Se o conjunto estiver cheio, a nova rota será inserida apenas se dominar fortemente pelo menos uma solução do pool atual. Todas soluções dominadas fortemente serão removidas do pool.

Uma variante mono-objetivo também foi implementada. A diferença consiste no tamanho fixo do *pool* em uma solução e o critério de comparação. A dominância não é levada em consideração, mas uma função objetivo explicitada na Eq. (3.2), onde $t(x)$ representa o maior tempo dentre os veículos que constituem a solução; $c(x)$, o consumo de todos veículos e $cf(x)$, a menor carga dentre os veículos ao final do percurso.

$$f(x) = \alpha * t(x) + \beta * c(x) + \gamma * cf(x) \tag{3.2}$$

Como queremos minimizar o tempo e o consumo e, por outro lado, maximizar o valor da carga final, α e β devem possuir sinal oposto ao de γ. Além disso, como os valores de $cf(x)$ variam entre 0 e 100, enquanto $t(x)$ e $c(x)$ chegam a valores muito maiores, o módulo de γ deve ser maior que os demais coeficientes. De forma empírica chegamos ao valores de 1, 1 e -5 respectivamente para α, β e γ. A função fitness fica então como demonstrado na Eq. (3.3).

$$f(x) = t(x) + c(x) - 5 * cf(x) \tag{3.3}$$

3.2.2.5 Implementação do VND

Para resolver o problema proposto, gerando as rotas, diferentes versões de um algoritmo foram implementadas em C++. Quatro versões do método SWAP foram implementadas, já que é a vizinhança mais dispendiosa em termos computacionais. As versões, como descritas na seção 3.2.2.2, foram denominadas S1, S2, S3 e S4. O tamanho do pool também foi considerado como uma variável e testes diferentes foram feitos para verificar seu impacto nos resultados.

O MOVND [8], variante multi-objetivo do VND, com um pool de soluções, é conhecido por executar um loop para cada solução e dentro de um loop para cada vizinhança. O Algoritmo 5 demonstra como essa busca local é implementada.

Algoritmo 5 MOVND

```
procedure MOVND(localPool, Vizinhanca)
    while localPool ≠ ∅ do
        S ← pop(localPool)
        insere(globalPool, S)
        for all k ∈ Vizinhanca do
            S' ← vizinho(S, k)
            if domina(S, S') then
                insere(localPool, S')
                k ← 0
            end if
        end for
    end while
    return globalPool
end procedure
```

O algoritmo recebe como entrada um pool ($localPool$) com a solução inicial gerada pelo construtor, além de um vetor com as estruturas de vizinhança. O $localPool$ é utilizado como um pool temporário e quando ele fica vazio, a execução é interrompida e o $globalPool$ é retornado. O método de inserção é implementado conforme descrito na seção anterior que discute os critérios de aceitação das soluções.

A atividade de calcular rotas pode ser realmente custosa quando executada muitas vezes através do algoritmo. Portanto, um pré-processamento foi implementado para fazer comparações se trouxer ganhos para os resultados. O método consiste em pré-calcular as melhores rotas entre os pontos importantes do mapa (clientes, área proibida e pontos de recarga). As rotas são salvas e depois lidas como parte da entrada do problema.

3.2.3 BRKGA

Outro algoritmo heurístico proposto neste trabalho é baseado em algoritmos genéticos (Genetic Algortihm - GA). GAs são métodos adaptativos utilizados para resolver problemas de busca e otimização. Eles evoluem a população (conjunto de um número fixo de indivíduos) aplicando o princípio de sobrevivência do mais forte de Darwin. Uma série de gerações são produzidas pelo algoritmo. O indivíduo mais apto, mais forte (maior *fitness*) da última geração é a solução final que o método deve retornar.

Alguns conceitos básicos de metaheurísticas baseadas em GA são:

- Indivíduo: cromosomo (série de genes) que representa a solução para o problema em análise.
- *Crossover*: indivíduos de uma geração são combinados para produzir descendentes para a próxima geração, passando adiante características de cada pai.
- Mutação: aleatoriamente muda o cromossomo dos indivíduos.

Para resolver o PMORVDG, problema proposto deste trabalho, utilizamos o BRKGA (Biased Random Key Genetic Algorithm) [12], metaheurística baseada em algoritmos genéticos como pode-se inferir de seu nome. O conceito de algoritmos genéticos e chaves aleatórias foi introduzido por Bean [3] para problemas sequenciais. Algumas aplicações dessa metaheurística são: conjunto de cobertura, empacotamento de retângulos e roteamento de pacotes na Internet.

Uma chave aleatória é um número real aleatório no intervalo contínuo $[0, 1)$. Soluções de problemas de otimização podem ser codificadas por chaves aleatórias. Um vetor de tamanho igual ao tamanho dos indivíduos é gerado e em cada posição, uma chave aleatória é gerada, como o exemplo de um cromossomo com 5 genes na figura 3.11. Cada gene, por exemplo, pode representar a ordem de visitação dos clientes em uma rota.

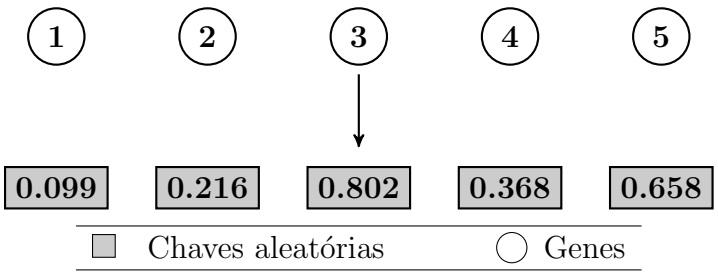

Figura 3.11: Codificação com chaves aleatórias

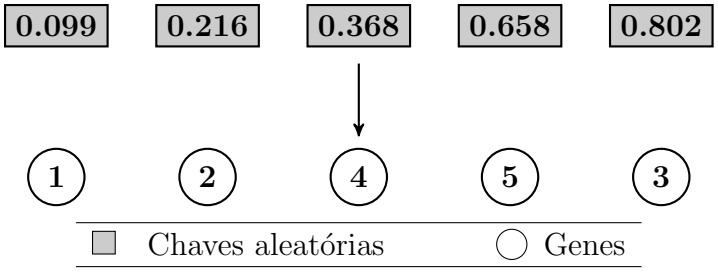

Figura 3.12: Decodificação por ordenação com chaves aleatórias

Um decodificador é um algoritmo determinístico que recebe um vetor de chaves aleatórias como entrada e retorna uma solução do problema de otimização. Bean [3] propôs decodificadores baseados na ordenação do vetor de chaves aleatórias para produzir uma sequência (vide figura 3.12).

BRKGA é um Algoritmo Genético de Chaves Aleatórias (RKGA). Porém, BRKGA e RKGA diferem em como pares de soluções são escolhidos e quão parametrizado o *crossover* é aplicado. Ambos os pais são escolhidos aleatoriamente, mas um é escolhido da população de soluções elite (soluções com maior *fitness*).

Algoritmo 6 BRKGA

```
procedure BRKGA(n, c, e, m, ρ, G)
    populacao ← geraPopulacaoInicial(n, c)
    rank(populacao)
    numGeracao ← 0
    while numGeracao < G do
        crossover(populacao, e)
        mutacao(populacao, m)
        rank(populacao)
        numGeracao ← numGeracao + 1
    end while
    retorna populacao[0]
end procedure
```

O algoritmo 6, da metaheurística, recebe como entrada os tamanhos da população (n), cromossomo (c), população elite (e), população de mutantes (m), número máximo de gerações (G) e ρ, que se refere ao fator de escolha do indivíduo elite.

A população inicial é formada por n vetores com c chaves aleatórias cada. O método $rank()$ utiliza o valor *fitness* de cada indivíduo para fazer a ordenação. Um loop é executado até atingir o número máximo de gerações. A cada iteração, os e primeiros indivíduos são mantidos na população, assim como outros m indivíduos aleatórios (mutação).

O crossover trata-se do cruzamento entre uma solução da população elite (com fator ρ) com outra solução da população para gerar um filho. Os pais são escolhidos aleatoriamente. São executados $(n - e - m)$ cruzamentos e os filhos gerados são inseridos nas

posições restantes da população.

Ao final de cada iteração, a população é rankeada. O algoritmo 7 refere-se ao método de comparação das soluções.

Algoritmo 7 Comparação

```
1: procedure COMPARAÇÃO(a, b)
2:     if numCargasInvalidas(a) ≠ numCargasInvalidas(b) then
3:         retorna numCargasInvalidas(a) < numCargasInvalidas(b)
4:     else
5:         retorna fitness(a) < fitness(b)
6:     end if
7: end procedure
```

O método recebe como entrada duas soluções da população atual e retorna *true* se a solução a é melhor que a solução b. O primeiro critério de comparação é a quantidade de vezes que o veículo estava com carga menor ou igual a zero ($numCargasInvalidas()$). A cada movimento do drone, é avaliada sua carga de combustível/energia, caso seja menor igual a zero, incrementa-se o valor dessa variável.

O segundo critério de comparação é uma função fitness propriamente dita, explicitada na equação 3.2. Mesma função utilizada no G-VND, facilitando a comparação posteriormente dos métodos.

3.2.3.1 Implementação

Para adaptar a metaheurística ao problema abordado nesse trabalho, duas principais mudanças tiveram que ser realizadas.

- Estágios

O algoritmo foi dividido em dois estágios. No primeiro estágio, a metaheurística observava o problema como um problema de roteamento em grafos simples (onde apenas a ordem de visitação importa), dessa forma, não era necessário calcular o caminho entre cada nó, utilizando apenas a distância euclidiana para isso. A ideia desse estágio inicial é acelerar a convergência da metaheurística, uma vez que trabalhar com roteamento em grid é mais custoso.

No segundo estágio, então, passamos a observar o problema original (em grid). Essa fase é responsável por refinar a solução gerada no estágio anterior. O que muda, na

metaheurística, na prática, é apenas a decodificação que passa a ser integrada ao método A* para gerar uma solução válida e o método de avaliação.

- Indivíduo

Cada indivíduo é representado por três vetores de chaves aleatórias: um representando a ordem de visitação, outro, a velocidade e por último a taxa de recarga do veículo. Os vetores possuem dimensão igual ao número de clientes mais número de estações de energia na instância.

Dessa forma, o fim da rota é determinado pela posição do último cliente representado no primeiro vetor. A velocidade entre dois pontos especiais (cliente/cliente, cliente/ponto de recarga) A e B é constante e igual ao valor representado no segundo vetor, na mesma posição do ponto A, na ordem de visitação. A taxa de recarga em um ponto C da rota segue o mesmo padrão, onde será determinada pelo terceiro vetor na mesma posição do ponto C, na ordem de visitação.

As chaves aleatórias representando a ordem de visitação segue o princípio básico do RKGA aplicado a roteamento. Dessa forma, a codificação e decodificação é realizada pela ordenação das chaves. Ao final da decodificação, então, temos a ordem de visitação dos clientes, como no exemplo da Figura 3.12.

Os vetores de velocidade e taxa de recarga seguem a mesma ideia entre si. A decodificação funciona multiplicando o valor da chave aleatória pelo valor máximo da variável. Assim, ao final da decodificação, temos um array de velocidades e taxas de recarga para cada trecho da rota, como no exemplo da Figura 3.13 com valor máximo da variável igual a 10.

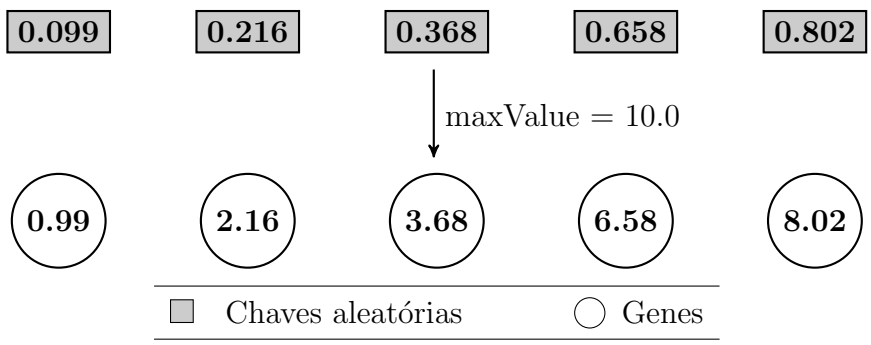

Figura 3.13: Decodificação das chaves aleatórias representando velocidade e taxa de recarga

Capítulo 4

Experimentos Computacionais

Nossos experimentos foram executados com uma máquina virtual com 2 GB de RAM virtual rodando uma versão 64-bit do Ubuntu 18.04 no VirtualBox 5.0.10 hypervisor com Windows 10 como sistema operacional host. A configuração do hardware do host consiste em um Intel Core i5-6400 CPU com 16 GB de RAM.

Como não existe de nosso conhecimento uma biblioteca bem estabelecida de instâncias para o problema abordado neste trabalho, com todas as restrições aqui consideradas, três instâncias de TSP bem conhecidas no formato euclideano 2D foram usadas como base dos testes: Christofides/Eilon eil51, eil101 e rat195. Essas instâncias possuem 51, 101 e 195 clientes, respectivamente. A partir desses, matrizes foram geradas representando a área que compreende todos os clientes, onde o valor de x varia do valor mínimo de x da instância ao máximo. O mesmo acontece com as coordenadas y. A discretização utilizada foi a unitária, ou seja, o intervalode uma unidade em x ou y na grid gerada representa exatamente uma unidade na instância original.

Tanto para o método G-MOVND quanto para o G-VND, as diferentes implementações do método swap intrarrota foram levadas em consideração para critérios comparativos. Como descrito na seção 3.2.2.2, S1 representa o swap completo, S2 o swap apenas entre os k pontos mais próximos, S3 entre os k mais distantes e S4 entre k randomicamente escolhidos. No caso dos experimentos, utilizamos o valor de $k = 3$. O método S2-S3 significa que tanto o método S2 quanto S3 foram utilizados como estruturas de vizinhança. O mesmo se aplica para S2-S4.

Depois que a matriz é gerada, pontos nela são escolhidos aleatoriamente e definidos como pontos de recarga e pontos proibidos. Para questão de testes, foi estipulada uma taxa de 5% da matriz para pontos proibidos e 1% para pontos de recarga. Para cada

instância, dois mapas diferentes foram gerados.

Cada amostra foi executada 3 vezes, cada uma por um tempo padrão de acordo com o tamanho da instância. Considerou-se o conjunto de soluções não dominadas de todas as execuções para questão de resultados.

Para resumir as diferentes implementações, um conjunto de 92 instâncias foram geradas, considerando os seguintes parâmetros:

- Instância
  - eil51A e eil51B: ambas com 51 clients, 5% de pontos proibidos e 1% de pontos de recarga. A diferença entre eles é devido à posição dos pontos proibidos e de recarga gerados aleatoriamente.
  - eil101A e eil101B: o mesmo que antes, mas com 101 clientes
  - rat195A e rat195B: 195 clientes
- Ponto de origem
  - Para cada instância, dois pontos de origem diferentes e aleatórios dos drones foram escolhidos (ex: eil51A1 e eil51A2)
- Número de drones
  - 1
  - 2
- Consumo variável (c_v)
  - O consumo variável é uma constante dependente da velocidade. Quanto maior a velocidade, maior será o consumo, mas menor será a duração da rota. Dois valores foram utilizados nos experimentos (0.05 e 0.1)
- Tempo limite de execução da metaheurística
  - Instâncias de 51 clientes: 5s, 10s, 30s, 60s, 120s, e 300s (padrão)
  - 101 clientes: 10s, 30s, 60s, 120s, 300s e 600s (padrão)
  - 195 clientes: 900s (padrão) e 1800s
- Pré-processamento
  - Sim
  - Não

4.1 Multi-objetivo

4.1.1 Medidas de Comparação

De modo a comparar os resultados obtidos durante os experimentos, foram utilizadas três medidas referentes à qualidade das soluções: hipervolume e cobertura.

De acordo com [25], o hipervolume é um indicador associado a uma aproximação dada pelo volume da porção do espaço objetivo que é fracamente dominado por um conjunto. Este indicador precisa da especificação de um ponto de referência Z que denota um limite superior sobre todos os objetivos. Neste problema, a função objetivo normalizada foi usada. O código do cálculo do hipervolume é fornecido por [11].

A outra medida é referente à quantidade de soluções na referência de Pareto ($PRef$) gerada por um método específico. Cobertura, conforme o algoritmo 8, representa o número de soluções do conjunto de pareto fracamente dominadas dividido pelo número total de soluções do conjunto de pareto.

A referência de Pareto é gerada a partir das soluções de todas execuções de todos métodos para cada instância. As soluções não dominadas, então, integram esse conjunto.

Algoritmo 8 Cobertura

```
 1: procedure COBERTURA(ConjuntoAtual, PRef)
 2:     solDominadas ← 0
 3:     for all b ∈ PRef do
 4:         for all a ∈ ConjuntoAtual do
 5:             if a.dominaFracamente(b) then
 6:                 solDominadas ← solDominadas + 1
 7:                 break
 8:             end if
 9:         end for
10:     end for
11:     return solDominadas / tamanho(PRef)
12: end procedure
```

4.1.2 Resultados Computacionais

Os primeiros testes foram desenvolvidos para verificar como o pré-processamento afeta os resultados. Para tanto, foram utilizadas amostras com pré-processamento e sem pré-processamento com pool de tamanho fixo de 5.

Tabela 4.1: Amostra eil51.

Instância	Métodos	Com PP		Sem PP	
		Hipervolume	Cobertura	Hipervolume	Cobertura
	S1	0.35	0.05	0.08	0
	S2	0.24	0	0.11	0.1
eil51P5R1a1	S2-S3	0.23	0	0.25	0
	S2-S4	0.34	0.11	0.29	0
	S4	**0.35**	**0.26**	0.34	0.05
	S1	0.23	0	0.002	0
	S2	0.25	0.09	0.2	0
eil51P5R1a2	S2-S3	0.26	0.05	**0.37**	0.14
	S2-S4	0.19	0	0.19	0
	S4	0.36	0.05	0.34	**0.23**
	S1	0.13	0	0.19	0
	S2	0.35	0	0.39	**0.36**
eil51P5R1b1	S2-S3	0.21	0	0.34	0.09
	S2-S4	0.31	0	0.34	0
	S4	**0.40**	0.09	0.33	0
	S1	**0.43**	0	0.2	0
	S2	0.32	0	0.31	0
eil51P5R1b2	S2-S3	0.34	0	0.3	0
	S2-S4	0.3	0.07	0.32	0
	S4	0.4	**0.36**	0.38	0

Tabela 4.2: Amostra eil101.

Instância	Métodos	Com PP		Sem PP	
		Hipervolume	Cobertura	Hipervolume	Cobertura
	S1	0.13	0	0.15	0
	S2	0.48	0.16	0.52	0.16
eil101P5R1a1	S2-S3	0.125	0	**0.53**	0.12
	S2-S4	0.30	0	0.51	**0.2**
	S4	0.37	0	0.27	0
	S1	0.1	0	0.07	0
	S2	0.29	0	0.17	0
eil101P5R1a2	S2-S3	0.16	0	0.09	0
	S2-S4	0.41	0.17	0.46	**0.22**
	S4	**0.54**	0.17	0.35	0
	S1	0.002	0	0.01	0
	S2	0.26	0.06	0.25	0
eil101P5R1b1	S2-S3	0.28	0	0.31	0
	S2-S4	**0.44**	**0.29**	0.4	0
	S4	0.44	0.18	0.4	0
	S1	0.03	0	0.12	0
	S2	0.11	0	0.44	0
eil101P5R1b2	S2-S3	0.27	0.15	0.45	0.15
	S2-S4	0.41	0	**0.51**	**0.25**
	S4	0.47	0.05	0.48	0.2

As Tabelas 4.1 e 4.2 mostram que o pré-processamento resulta em melhores valores de hipervolume e cobertura na maioria dos testes. Isso significa que o pré-cálculo de rotas gera um conjunto de soluções mais diversificadas, o que é um bom resultado quando estamos falando de um problema multiobjetivo.

Além disso, também podemos observar que os melhores resultados foram provenientes das metaheurísticas com os métodos S2-S4 e S4 na busca local. Esses, são os métodos mais rápidos, pois S4 trata-se de um método randômico, logo, não exaustivo. Por isso, em um mesmo intervalo de tempo, acabam chegando em resultados melhores nesses testes.

Os próximos testes visavam avaliar como o tamanho do pool afetava o resultado final.

Tabela 4.3: Amostra eil51P5R1a1.

Tamanho	Método	Hipervolume	Cobertura
	S1	0.002	0
	S2	0.09	0
01	S2-S3	0.12	0
	S2-S4	0.31	0
	S4	0.37	0
	S1	0.53	0.05
	S2	0.44	0
05	S2-S3	0.44	0
	S2-S4	0.52	0.1
	S4	0.52	**0.24**
	S1	0.5	0
	S2	0.42	0
10	S2-S3	0.39	0
	S2-S4	0.42	0
	S4	**0.53**	0.05

Tabela 4.4: Amostra eil51P5R1a2.

Tamanho	Método	Hipervolume	Cobertura
	S1	0.02	0
	S2	0.001	0
01	S2-S3	0.25	0
	S2-S4	0.38	0
	S4	0.37	0
	S1	0.37	0
	S2	0.45	0
05	S2-S3	0.46	0.03
	S2-S4	0.33	0
	S4	0.48	0
	S1	0.44	0
	S2	0.48	0
10	S2-S3	0.38	0
	S2-S4	0.49	0
	S4	**0.57**	**0.16**

Tabela 4.5: Amostra eil51P5R1b1.

Tamanho	Método	Hipervolume	Cobertura
	S1	0.34	0
	S2	0.12	0
01	S2-S3	0.07	0
	S2-S4	0.35	0
	S4	0.36	0
	S1	0.29	0
	S2	0.46	0
05	S2-S3	0.35	0
	S2-S4	0.43	0
	S4	0.48	0.09
	S1	0.44	0
	S2	0.42	0
10	S2-S3	0.42	0
	S2-S4	0.48	0
	S4	**0.52**	**0.45**

Tabela 4.6: Amostra eil51P5R1b2.

Tamanho	Método	Hipervolume	Cobertura
	S1	0.25	0
	S2	0.18	0
01	S2-S3	0.22	0
	S2-S4	0.03	0
	S4	0.27	0
	S1	0.45	0
	S2	0.34	0
05	S2-S3	0.36	0
	S2-S4	0.32	0.05
	S4	0.41	**0.2**
	S1	0.43	0
	S2	0.37	0
10	S2-S3	0.35	0
	S2-S4	0.38	0
	S4	**0.47**	**0.2**

Tabela 4.7: Amostra eil101P5R1a1.

Tamanho	Método	Hipervolume	Cobertura
	S1	0.06	0
	S2	0.1	0
01	S2-S3	0.004	0
	S2-S4	0.05	0
	S4	0.10	0
	S1	0.11	0
	S2	0.46	0.04
05	S2-S3	0.12	**0.18**
	S2-S4	0.28	0
	S4	0.36	0
	S1	0.04	0
	S2	0.44	0
10	S2-S3	0.32	0
	S2-S4	**0.47**	0.14
	S4	0.47	0

Tabela 4.8: Amostra eil101P5R1a2.

Tamanho	Método	Hipervolume	Cobertura
	S1	0.05	0
	S2	0.25	0
01	S2-S3	0.26	0
	S2-S4	0.12	0
	S4	0.12	0
	S1	0.04	0
	S2	0.26	0
05	S2-S3	0.13	0
	S2-S4	0.38	0.1
	S4	**0.5**	0.1
	S1	0.02	0
	S2	0.34	0
10	S2-S3	0.28	0
	S2-S4	0.48	0.03
	S4	0.47	**0.13**

Tabela 4.9: Amostra eil101P5R1b1.

Tamanho	Método	Hipervolume	Cobertura
	S1	0.22	0
	S2	0.04	0
01	S2-S3	0.38	0
	S2-S4	0.30	0
	S4	0.41	0
	S1	0.38	0
	S2	0.10	0
05	S2-S3	0.37	0.03
	S2-S4	**0.5**	**0.16**
	S4	0.5	0.09
	S1	0.1	0
	S2	0.45	0.03
10	S2-S3	0.40	0.06
	S2-S4	0.38	0
	S4	0.49	0.03

Tabela 4.10: Amostra eil101P5R1b2.

Tamanho	Método	Hipervolume	Cobertura
	S1	0.12	0
	S2	0.23	0
01	S2-S3	0.39	0
	S2-S4	0.25	0
	S4	0.35	0
	S1	0.14	0
	S2	0.46	0
05	S2-S3	0.48	0.08
	S2-S4	**0.52**	**0.13**
	S4	0.49	0.1
	S1	0.07	0
	S2	0.42	0.03
10	S2-S3	0.27	0.08
	S2-S4	0.5	0.08
	S4	0.43	0

Em termos de tamanho máximo do *pool*, nas amostras com instâncias menores, pools maiores, melhores os resultados. E com instâncias maiores, o *pool* de tamanho intermediário (cinco soluções) produziu melhores soluções, já que o tamanho da instância requer mais tempo computacional durante a busca local e um *pool* maior aumenta essa complexidade da busca local. Todavia, *pools* menores geram menor diversidade.

O mesmo conjunto de testes foi executado adicionando-se um veículo dessa vez às instâncias.

Tabela 4.11: Instância eil51 com 2 drones.

Instância	Método	Hipervolume	Cobertura
	S1	0.32	**0.2**
	S2	0.28	0.05
eil51P5R1a1	S2-S3	0.13	0
	S2-S4	**0.33**	**0.2**
	S4	0.28	0.05
	S1	0.16	0.2
	S2	0.09	0.2
eil51P5R1a2	S2-S3	0.12	0.1
	S2-S4	**0.19**	**0.5**
	S4	0.06	0
	S1	0.22	0.25
	S2	0.14	0
eil51P5R1b1	S2-S3	**0.24**	**0.42**
	S2-S4	0.17	0
	S4	0.07	0.08
	S1	**0.30**	**0.31**
	S2	0.19	0
eil51P5R1b2	S2-S3	0.27	**0.31**
	S2-S4	0.22	0.15
	S4	0.21	0.08

Tabela 4.12: Instância eil101 com 2 drones.

Instância	Método	Hipervolume	Cobertura
	S1	0.01	0
	S2	0.04	0
eil101P5R1a1	S2-S3	0.01	0
	S2-S4	**0.12**	**1**
	S4	0.04	0
	S1	**0.08**	**0.25**
	S2	0.05	0
eil101P5R1a2	S2-S3	0.05	0
	S2-S4	0.02	0
	S4	0.06	0
	S1	0	0
	S2	0.1	0.08
eil101P5R1b1	S2-S3	**0.27**	**0.31**
	S2-S4	0.26	0.23
	S4	0.09	0
	S1	**0.12**	**0.6**
	S2	0.07	0.2
eil101P5R1b2	S2-S3	0.01	0
	S2-S4	0.07	0
	S4	0.02	0

Os resultados não apontam para um método dominante perante os outros nas instâncias com 2 veículos.

4.2 Mono Objetivo

Os mesmos testes da Seção 4.1 foram executados utilizando-se o método BRKGA e o GVND (mono-objetivo). Para ambas metaheurísticas, foi utilizada a Eq. (3.2) como função objetivo.

As Tabelas 4.13 e 4.14 mostram que o pré-processamento, assim como no caso multi-objetivo, resulta em melhores soluções em quase todos os testes.

Tabela 4.13: Comparação do pré-processamento em instâncias da família eil51.

Instância	Métodos	Com PP				Sem PP			
		O1	O2	O3	F.O.	O1	O2	O3	F.O.
eil51P5R1a1	GVND S1	97	551	274,8	340,8	98	699	335,4	544,4
	GVND S2	98	798	374,4	682,4	93	687	333,6	555,6
	GVND S2-S3	**99**	634	300,6	439,6	95	710	331,8	566,8
	GVND S2-S4	97	**542**	**267,6**	**324,6**	95	583	281,4	389,4
	GVND S4	97	586	282,6	383,6	98	562	269,4	341,4
	BRKGA	**99**	989	467	961	90	1056	511	1117
eil51P5R1a2	GVND S1	95	656	315	496	91	809	364,8	718,8
	GVND S2	91	729	345,6	619,6	87	724	337,8	626,8
	GVND S2-S3	97	708	332,55	555,55	94	746	364,8	640,8
	GVND S2-S4	**98**	**563**	**269,4**	**342,4**	98	625	297,6	432,6
	GVND S4	95	640	307,8	472,8	98	599	287,1	396,1
	BRKGA	**98**	1094	523	1127	91	1238	596	1379
eil51P5R1b1	GVND S1	97	605	290,4	410,4	98	617	303	430
	GVND S2	98	729	342,6	581,6	98	709	348,6	567,6
	GVND S2-S3	98	651	309,6	470,6	98	616	296,4	422,4
	GVND S2-S4	95	589	283,2	397,2	98	583	279	372
	GVND S4	**99**	**563**	**271,8**	**339,8**	98	574	274,8	358,8
	BRKGA	97	1102	539	1156	**99**	1268	616	1389
eil51P5R1b2	GVND S1	97	625	304,8	444,8	94	781	361,8	672,8
	GVND S2	97	779	358,05	652,05	91	746	352,95	643,95
	GVND S2-S3	95	671	287,55	483,55	96	689	323,4	532,4
	GVND S2-S4	**99**	561	271,2	337,2	98	628	298,2	436,2
	GVND S4	97	**533**	**257,4**	**305,4**	98	596	286,8	392,8
	BRKGA	92	1119	548	1207	92	1466	717	1723

Tabela 4.14: Comparação do pré-processamento em instâncias da família eil101.

Instância	Métodos	Com PP				Sem PP			
		O1	O2	O3	F.O.	O1	O2	O3	F.O.
eil101P5R1a1	GVND S1	91	2133	984,6	2662,6	-	-	-	-
	GVND S2	**99**	1232	585,15	1322,15	98	1100	523,2	1133,2
	GVND S2-S3	98	1254	602,4	1366,4	99	1235	579,15	1319,15
	GVND S2-S4	98	1103	540	1153	98	1293	620,4	1423,4
	GVND S4	97	**1090**	**526,8**	**1131,8**	-	-	-	-
	BRKGA	86	1755	598	1923	81	1487	749	1831
eil101P5R1a2	GVND S1	**98**	1877	844,2	2231,2	-	-	-	-
	GVND S2	95	1258	593,4	1376,4	-	-	-	-
	GVND S2-S3	89	1218	570	1343	95	1280	590,7	1395,7
	GVND S2-S4	94	1096	515,1	1141,1	97	1012	469,95	996,95
	GVND S4	95	1052	499,2	1076,2	93	**944**	**454,8**	**933,8**
	BRKGA	89	1646	580	1781	**98**	3018	1461	3989
eil101P5R1b1	GVND S1	92	1623	735,6	1898,6	**99**	1992	897,6	2394,6
	GVND S2	96	1261	590,55	1371,55	**99**	1233	582,15	1320,15
	GVND S2-S3	92	1286	615	1441	**99**	1264	596,4	1365,4
	GVND S2-S4	**99**	1003	477,75	985,75	**99**	1024	492,15	1021,15
	GVND S4	**99**	**948**	**460,2**	**913,2**	98	1087	522,6	1119,6
	BRKGA	94	1381	665	1576	94	1442	710	1682
eil101P5R1b2	GVND S1	96	2013	941,4	2474,4	-	-	-	-
	GVND S2	95	1200	570,6	1295,6	94	1337	621,6	1488,6
	GVND S2-S3	**99**	1086	517,2	1108,2	95	1223	579,45	1327,45
	GVND S2-S4	95	1039	493,8	1057,8	95	1026	494,4	1045,4
	GVND S4	98	**976**	**473,4**	**959,4**	94	1146	567,6	1243,6
	BRKGA	95	1907	604	2036	66	1924	945	2539

A Tabela 4.15 representa as soluções para as instâncias da família eil51 com 2 veículos. Podemos, então, verificar que o método S1 gera as melhores soluções. Além disso, se aliarmos às informações dessa tabela, os gráficos da Figura 4.1, fica fácil notar que apesar do GVND gerar as melhores soluções, essa metaheurística demora a gerar soluções válidas. O contrário, acontece com o BRKGA, que consegue gerar soluções válidas muito mais rapidamente que os outros métodos, mas demora pra chegar em soluções tão boas quanto a dos outros métodos.

Tabela 4.15: Instância eil51 com 2 drones.

Instância	Métodos	O1	O2	O3	F.O.
eil51P5R1a1	GVND S1	89	**304**	498,3	357,3
	GVND S2	**97**	404	506,5	425,5
	GVND S2-S3	**97**	389	518,3	422,3
	GVND S2-S4	95	314	499,8	**338,8**
	GVND S4	94	331	550	411
	BRKGA	82	1229	**435**	1254
eil51P5R1a2	GVND S1	89	**301**	**493,9**	**349,9**
	GVND S2	**95**	327	513,7	365,7
	GVND S2-S3	91	316	529,1	390,1
	GVND S2-S4	93	318	528	381
	GVND S4	90	409	590,9	549,9
	BRKGA	84	2417	488	2485
eil51P5R1b1	GVND S1	**99**	306	**481,7**	**292,7**
	GVND S2	96	295	498,3	313,3
	GVND S2-S3	96	339	565,4	424,4
	GVND S2-S4	**99**	380	513,8	398,8
	GVND S4	96	**305**	504	329
	BRKGA	80	1454	542	1596
eil51P5R1b2	GVND S1	95	**303**	**499,4**	**327,4**
	GVND S2	97	310	506,2	331,2
	GVND S2-S3	91	330	521,4	396,4
	GVND S2-S4	**98**	382	561,6	453,6
	GVND S4	95	312	518,1	355,1
	BRKGA	60	1145	714	1559

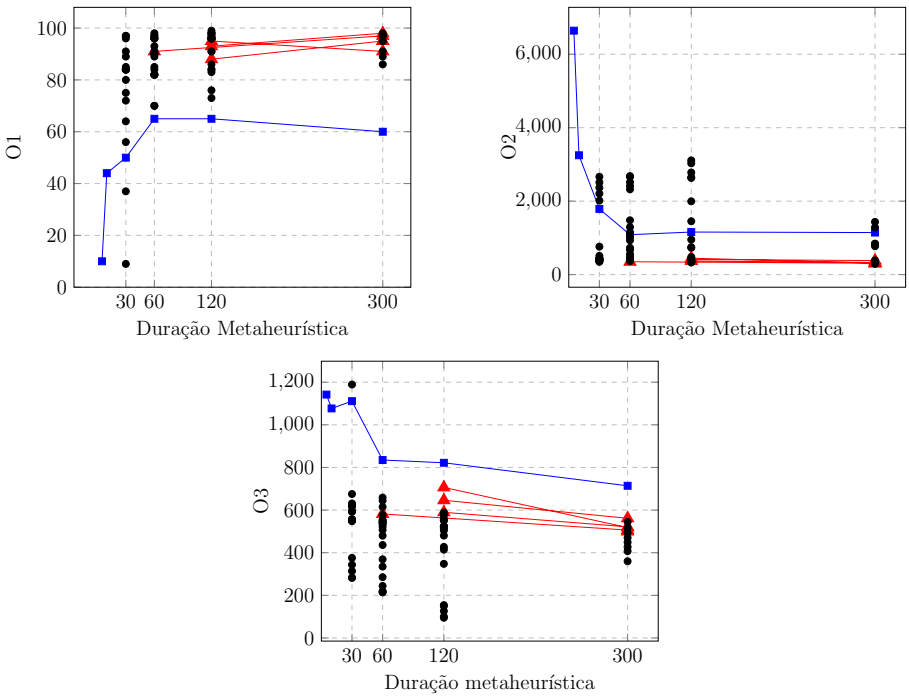

Figura 4.1: Variação dos valores dos objetivos para instância eil51b2 em decorrência da duração da metaheurística: BRKGA (azul), GMOVND (preto) e GVND (vermelho)

As Tabelas 4.16 a 4.20 resumem os experimentos computacionais comparando os algoritmos propostos nesse trabalho (BRKGA, G-VND e G-MOVND) através dos valores de cada objetivo (carga final, tempo da rota e consumo) para todas instâncias geradas.

A primeira tabela compreende as instâncias básicas com tempo limite de execução padrão, a exceção da instância rat195 que foi executada com 900 e 1800 segundos, uma vez que, pelo tempo padrão, nem o G-VND nem o G-MOVND geraram soluções válidas. O método multiobjetivo, por conter um pool de soluções, consegue gerar soluções boas que atendem a cada objetivo separadamente, o que pode ser observado na tabela, pois quando encontra soluções válidas, consegue ganhar dos outros métodos em O1, O3 e aproximar-se dos resultados do G-VND em O2.

Essa versão mono-objetivo, por outro lado, trabalha em cima de apenas uma solução e por isso encontra um pouco mais dificuldade na variabilidade de sua solução, perdendo em O1 e O3 em boa parte das instâncias e em O2 nas instâncias mais difíceis como das tabelas 4.19 e 4.20. Todavia, justamente por trabalhar com apenas uma solução, ela consegue ser melhor (em termos de dominância) do que cada uma, individualmente, gerada pelo método multiobjetivo.

O BRKGA, embora não se destaque pela qualidade de suas soluções, possui méritos

por gerar soluções válidas de maneira muito mais rápida que os outros dois métodos. Essa afirmativa é validada pelas instâncias mais complexas, como as baseadas no rat195 e nas instâncias com c_v igual a 0,1.

Tabela 4.16: Comparação dos valores das funções objetivo nas instâncias padrões.

	O1			O2			O3		
Instância	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND
eil51a1_pp_1d_005_300	**99**	**99**	**99**	989	**542**	570	466	268	**121**
eil51a2_pp_1d_005_300	98	97	**99**	1094	**588**	601	523	268	**78**
eil51b1_pp_1d_005_300	97	98	**99**	1102	**543**	560	539	265	**179**
eil51b2_pp_1d_005_300	92	98	**99**	1119	**547**	570	548	263	**183**
eil101a1_pp_1d_005_600	86	98	**99**	1755	**943**	1078	598	453	**275**
eil101a2_pp_1d_005_600	89	**99**	**99**	1646	**915**	1130	580	434	**264**
eil101b1_pp_1d_005_600	94	**99**	**99**	1381	**942**	989	665	442	**181**
eil101b2_pp_1d_005_600	95	97	**99**	1907	**922**	989	604	432	**181**
rat195a1_pp_1d_005_900	**94**	-	-	**59055**	-	-	**9827**	-	-
rat195a2_pp_1d_005_900	**99**	-	-	**87504**	-	-	**10866**	-	-
rat195b1_pp_1d_005_900	**89**	-	-	**97415**	-	-	**11559**	-	-
rat195b2_pp_1d_005_900	**39**	-	-	**63650**	-	-	**9449**	-	-
rat195a1_pp_1d_005_1800	**94**	-	-	**91888**	-	-	**7677**	-	-
rat195a2_pp_1d_005_1800	**96**	-	-	**123963**	-	-	**12201**	-	-
rat195b1_pp_1d_005_1800	**87**	-	-	**95716**	-	-	**9707**	-	-
rat195b2_pp_1d_005_1800	**46**	-	-	**144515**	-	-	**12572**	-	-
Vitórias/Empates	**9**	3	8	**8**	**8**	0	**8**	0	**8**

Tabela 4.17: Comparação dos valores das funções objetivo nas instâncias sem pré-processamento.

	O1			O2			O3		
Instância	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND
eil51a1_1d_005_300	90	**99**	**99**	1056	**562**	584	511	269	**145**
eil51a2_1d_005_300	91	98	**99**	1238	599	**588**	596	287	**168**
eil51b1_1d_005_300	99	98	**99**	1269	574	**535**	616	275	**114**
eil51b2_1d_005_300	92	**99**	**99**	1466	596	**595**	717	286	**237**
eil101a1_1d_005_600	81	98	**99**	1487	**981**	1059	749	477	**198**
eil101a2_1d_005_600	98	**99**	**99**	3018	**981**	1276	1461	463	**221**
eil101b1_1d_005_600	94	**99**	**99**	1442	1107	**1039**	710	528	**404**
eil101b2_1d_005_600	66	95	**99**	1924	**1092**	1116	945	509	**312**
rat195a1_1d_005_900	**69**	-	-	**155419**	-	-	**10323**	-	-
rat195a2_1d_005_900	-	-	-	-	-	-	-	-	-
rat195b1_1d_005_900	**96**			**184292**			**10628**		
rat195b2_1d_005_900	-	-	-	-	-	-	-	-	-
Vitórias/Empate	2	4	**8**	2	**4**	**4**	2	0	**8**

Tabela 4.18: Comparação dos valores das funções objetivo nas instâncias com 2 drones.

	O1			O2			O3		
Instância	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND
eil51a1_pp_2d_005_300	85	95	**98**	804	**267**	297	659	239	**152**
eil51a2_pp_2d_005_300	91	92	**98**	1066	**258**	261	918	**238**	241
eil51b1_pp_2d_005_300	98	98	**99**	867	**266**	**266**	778	242	**203**
eil51b2_pp_2d_005_300	89	97	**99**	1115	279	**278**	968	238	**152**
eil101a1_pp_2d_005_600	83	**51**	55	2359	**723**	728	2221	**620**	677
eil101a2_pp_2d_005_600	86	**91**	**91**	3342	**665**	781	3090	**595**	697
eil101b1_pp_2d_005_600	87	94	**98**	2220	**523**	528	2073	472	**413**
eil101b2_pp_2d_005_600	75	90	**94**	3241	**538**	703	2565	**506**	639
rat195a1_pp_2d_005_900	**44**	-	-	**73438**	-	-	**11938**	-	-
rat195a2_pp_2d_005_900	-	-	-	-	-	-	-	-	-
rat195b1_pp_2d_005_900	**63**	-	-	**134721**	-	-	**13887**	-	-
rat195b2_pp_2d_005_900	**34**			**108718**			**14632**		
Vitórias/Empate	3	2	**7**	3	**7**	2	3	**4**	**4**

Tabela 4.19: Comparação dos valores das funções objetivo nas instâncias eil51 com 2 drones e c_v igual a 0,1 (300s, 120s, 60s, 30s, 10s e 5s).

	O1			O2			O3		
Instância	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND
eil51a1_pp_2d_010_300	82	98	**99**	1229	288	**275**	435	407	**243**
eil51a2_pp_2d_010_300	84	**98**	97	2417	**283**	285	488	416	**159**
eil51b1_pp_2d_010_300	80	**99**	**99**	1454	**292**	309	542	445	**207**
eil51b2_pp_2d_010_300	60	97	**98**	1145	311	**292**	714	420	**360**
eil51a1_pp_2d_010_120	88	97	**99**	2013	310	**292**	494	515	**225**
eil51a2_pp_2d_010_120	67	**98**	91	17411	319	**293**	442	540	**239**
eil51b1_pp_2d_010_120	65	**99**	**99**	1744	301	**295**	535	499	**232**
eil51b2_pp_2d_010_120	65	98	**99**	1159	418	**328**	822	575	**95**
eil51a1_pp_2d_010_60	74	**99**	95	2466	339	**338**	526	366	**288**
eil51a2_pp_2d_010_60	73	**92**	**92**	3703	375	**313**	639	590	**201**
eil51b1_pp_2d_010_60	48	97	**99**	3071	344	**319**	549	556	**263**
eil51b2_pp_2d_010_60	66	95	**98**	1087	**349**	370	835	558	**214**
eil51a1_pp_2d_010_30	70	-	**96**	171691	-	**344**	574	-	**245**
eil51a2_pp_2d_010_30	**82**	-	76	3254	-	**312**	869	-	**525**
eil51b1_pp_2d_010_30	67	-	**97**	1925	-	**359**	841	-	**385**
eil51b2_pp_2d_010_30	50	-	**97**	1788	-	**347**	1111	-	**282**
eil51a1_pp_2d_010_10	**83**	-	-	**3694**	-	-	**960**	-	-
eil51a2_pp_2d_010_10	**61**	-	-	**46142**	-	-	**889**	-	-
eil51b1_pp_2d_010_10	**42**	-	-	**3104**	-	-	**1145**	-	-
eil51b2_pp_2d_010_10	**44**	-	-	**3249**	-	-	**1077**	-	-
eil51a1_pp_2d_010_5	**86**	-	-	**5612**	-	-	**1055**	-	-
eil51a2_pp_2d_010_5	**72**	-	-	**12997**	-	-	**1178**	-	-
eil51b1_pp_2d_010_5	**18**	-	-	**6813**	-	-	**1110**	-	-
eil51b2_pp_2d_010_5	**10**	-	-	**6642**	-	-	**1142**	-	-
Vitórias/Empates	9	6	**12**	8	3	0	8	0	**13**

Tabela 4.20: Comparação dos valores das funções objetivo nas instâncias eil101 com 2 drones e c_v igual a 0,1 (600s, 300s, 120s, 60s, 30s e 10s).

	O1			O2			O3		
Instância	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND
eil101a1_pp_2d_010_600	**75**	-	-	**4275**	-	-	**1279**	-	-
eil101a2_pp_2d_010_600	**77**	-	-	**3327**	-	-	**13882**	-	-
eil101b1_pp_2d_010_600	**94**	-	-	**2844**	-	-	**1298**	-	-
eil101b2_pp_2d_010_600	**88**	-	-	**3382**	-	-	**1485**	-	-
eil101a1_pp_2d_010_300	**70**	-	-	**3415**	-	-	**1538**	-	-
eil101a2_pp_2d_010_300	**70**	-	-	**5681**	-	-	**1591**	-	-
eil101b1_pp_2d_010_300	**80**	-	-	**2602**	-	-	**1826**	-	-
eil101b2_pp_2d_010_300	**83**	-	-	**5456**	-	-	**1599**	-	-
eil101a1_pp_2d_010_120	**94**	-	-	**5734**	-	-	**1751**	-	-
eil101a2_pp_2d_010_120	**59**	-	-	**9791**	-	-	**1928**	-	-
eil101b1_pp_2d_010_120	**63**	-	-	**5469**	-	-	**2894**	-	-
eil101b2_pp_2d_010_120	**76**	-	-	**3663**	-	-	**2480**	-	-
eil101a1_pp_2d_010_60	**95**	-	-	**6853**	-	-	**1973**	-	-
eil101a2_pp_2d_010_60	**40**	-	-	**16252**	-	-	**1859**	-	-
eil101b1_pp_2d_010_60	**97**	-	-	**11621**	-	-	**2070**	-	-
eil101b2_pp_2d_010_60	**58**	-	-	**6563**	-	-	**2245**	-	-
eil101a1_pp_2d_010_30	**29**	-	-	**12176**	-	-	**2336**	-	-
eil101a2_pp_2d_010_30	**7**	-	-	**19280**	-	-	**2166**	-	-
eil101b1_pp_2d_010_30	**63**	-	-	**9231**	-	-	**2543**	-	-
eil101b2_pp_2d_010_30	**46**	-	-	**7188**	-	-	**2699**	-	-
eil101a1_pp_2d_010_10	**46**	-	-	**23792**	-	-	**2945**	-	-
eil101a2_pp_2d_010_10	**2**	-	-	**76780**	-	-	**2855**	-	-
eil101b1_pp_2d_010_10	**70**	-	-	**231696**	-	-	**2961**	-	-
eil101b2_pp_2d_010_10	**56**	-	-	**11341**	-	-	**3398**	-	-
Vitórias/Empates	**24**	0	0	**24**	0	0	**24**	0	0

De forma a realizar uma comparação na eficiência das diferentes implementações do método swap, as tabelas 4.21 a 4.24 resumem o valor de hipervolume em todas instâncias tanto para o G-VND quanto para versão multiobjetivo. É interessante notar que para as instâncias mais simples (tabelas 4.21 e 4.22), o método S4 (randômico), menos custoso, gera as melhores soluções, enquanto para as instâncias mais complexas (tabelas 4.23 e 4.24) o método que gera as melhores soluções é o S1 (swap completo), mais custoso.

Tabela 4.21: Comparação dos valores de hipervolume nas instâncias padrões.

	G-VND					G-MOVND				
Instância	S1	S2	S2-S3	S2-S4	S4	S1	S2	S2-S3	S2-S4	S4
eil51a1_pp_1d_005_300	0.020795	2e-06	0.030861	**0.023091**	0.016475	0.134317	0.061298	0.091998	0.15867	**0.211702**
eil51a2_pp_1d_005_300	0.010786	0.000325	0.005431	**0.057932**	0.013954	0.149194	0.020133	0.101524	0.077179	**0.18835**
eil51b1_pp_1d_005_300	0.025811	0.004155	0.019824	0.010168	**0.06474**	0.108522	0.049915	0.107278	0.167067	**0.220386**
eil51b2_pp_1d_005_300	0.025261	0.001832	0.007746	**0.078744**	0.058592	0.239336	0.133163	0.115436	0.169022	**0.29454**
eil101a1_pp_1d_005_600	0	0.154562	0.128263	**0.174743**	0.159414	0.079605	**0.441809**	0.172888	0.255015	0.2897
eil101a2_pp_1d_005_600	1e-06	0.057373	0.009536	0.081314	**0.105031**	0.03035	0.20937	0.136948	0.332207	**0.372734**
eil101b1_pp_1d_005_600	0.004005	0.059317	0.025549	0.175163	**0.196375**	0.002264	0.190141	0.209588	**0.388912**	0.380655
eil101b2_pp_1d_005_600	8e-06	0.132503	0.20232	0.190885	**0.239937**	0.002315	0.350836	0.298897	0.403424	**0.40505**
Vitórias/Empates	0	0	0	**4**	**4**	0	1	0	1	**6**

Tabela 4.22: Comparação dos valores de hipervolume nas instâncias sem pré-processamento.

	G-VND					G-MOVND				
Instância	S1	S2	S2-S3	S2-S4	S4	S1	S2	S2-S3	S2-S4	S4
eil51a1_1d_005_300	0.00257	0.000583	0.001212	0.016974	**0.044605**	0.079965	0.145023	0.092194	0.18989	**0.20687**
eil51a2_1d_005_300	0.000739	0.001257	0.003751	0.05076	**0.063572**	0.00175	0.127341	0.245604	0.139505	**0.288231**
eil51b1_1d_005_300	0.00582	1e-06	0.006716	0.012041	**0.013661**	0.076428	0.231991	0.002398	**0.156403**	0.141462
eil51b2_1d_005_300	0.001511	0.000914	0.016922	0.045074	**0.05892**	0.174241	0.038657	0.171226	0.21436	**0.26774**
eil101a1_1d_005_600	0	0.013875	0.184596	0.204001	**0.257515**	0.097376	**0.445235**	0.372797	0.321633	0.244023
eil101a2_1d_005_600	0	0.067694	0.046191	**0.157666**	0.13613	0.050911	0.002937	0.069462	**0.398372**	0.323383
eil101b1_1d_005_600	0.002451	0.134832	0.087889	0.050036	**0.14625**	0	0.0157	0.110997	**0.163167**	0.138303
eil101b2_1d_005_600	0	0.116751	0.156813	**0.207567**	0.014252	0	0.100817	0.20806	0.33971	**0.375293**
Vitórias/Empates	0	0	0	2	**6**	0	1	0	3	**4**

Tabela 4.23: Comparação dos valores de hipervolume nas instâncias com 2 drones.

	G-VND					G-MOVND				
Instância	S1	S2	S2-S3	S2-S4	S4	S1	S2	S2-S3	S2-S4	S4
eil51a1_pp_2d_005_300	**0.033723**	0.021324	0.008385	0.016154	0.000514	0.319313	0.278176	0.127342	**0.333434**	0.282017
eil51a2_pp_2d_005_300	0.046482	**0.067863**	0.032031	0.069292	0.02343	0.155661	0.092021	0.121542	**0.194389**	0.057058
eil51b1_pp_2d_005_300	**0.017558**	0.007942	0.002682	0.00724	0.003639	0.220793	0.142356	**0.239217**	0.172656	0.067615
eil51b2_pp_2d_005_300	**0.01701**	0.000464	0.005443	0.000128	0.01946	**0.302163**	0.185784	0.267395	0.219764	0.211709
eil101a1_pp_2d_005_600	0.023204	0.013148	**0.046031**	0.007826	0.003264	0.007485	0.042657	0.010769	**0.119539**	0.040367
eil101a2_pp_2d_005_600	0.021231	0.012546	**0.037505**	0.041488	0.001901	**0.082906**	0.047043	0.052569	0.022703	0.054758
eil101b1_pp_2d_005_600	0.142191	0.057098	0.051016	**0.152692**	0.002819	0	0.097262	**0.269367**	0.258223	0.094262
eil101b2_pp_2d_005_600	**0.032291**	0.01123	1e-06	0.000647	1e-06	**0.122753**	0.073414	0.01136	0.065715	0.022717
Vitórias/Empates	**4**	1	2	1	0	**3**	0	2	**3**	0

Tabela 4.24: Comparação dos valores de hipervolume nas instâncias eil51 com 2 drones e c_v igual a 0,1 (300s, 120s, 60s, 30s, 10s e 5s).

	G-VND					G-MOVND				
Instância	S1	S2	S2-S3	S2-S4	S4	S1	S2	S2-S3	S2-S4	S4
eil51a1_pp_2d_010_300	**0.069485**	0.021716	0.015454	0.062854	0.000548	0.131346	**0.281926**	0.149923	0.199639	0.110301
eil51a2_pp_2d_010_300	**0.099269**	0.06019	0.079028	0.030441	0.000709	0.025053	0.124247	0.132469	**0.245771**	0.213089
eil51b1_pp_2d_010_300	0.065252	**0.081112**	0.039958	0.020731	0.044003	0.268067	0.290394	0.223338	0.284001	**0.299283**
eil51b2_pp_2d_010_300	**0.06844**	0.065438	0.06374	0.054796	0.026421	**0.253256**	0.095927	0.158712	0.154838	0.230625
eil51a1_pp_2d_010_120	**0.057691**	0.039874	0.028219	0.004429	0.011841	0.249306	0.097837	**0.387598**	0.304051	0.257775
eil51a2_pp_2d_010_120	0	0.003552	0.000195	**0.014564**	0.020997	**0.502308**	0.330587	0.161815	0.50082	0.400457
eil51b1_pp_2d_010_120	0.022257	0.007968	0.010326	**0.038498**	1e-06	**0.255684**	0.24811	0.18098	0.184642	0.241851
eil51b2_pp_2d_010_120	**0.069421**	0.015079	0.020671	0.004019	1e-06	**0.488216**	0.231857	0.137772	0.37673	0.244011
eil51a1_pp_2d_010_60	0.200294	**0.24779**	0.122142	0.150754	0.136076	0.141028	**0.213526**	0.162016	0.15151	0.167547
eil51a2_pp_2d_010_60	**0.234699**	0.170401	0.105089	0.003862	0	**0.44086**	0.431543	0.37044	0.128351	0.355009
eil51b1_pp_2d_010_60	0.005943	0.052966	0.051464	**0.059508**	0.001448	**0.313255**	0.111483	0.17727	0.038938	0.15267
eil51b2_pp_2d_010_60	0.083241	**0.122473**	2e-06	0.118052	0.000142	0.020816	0.054042	0.144838	0.159713	**0.311948**
eil51a1_pp_2d_010_30	-	-	-	-	-	0	**0.092966**	0.028953	0	0.073083
eil51a2_pp_2d_010_30	-	-	-	-	-	**0.049724**	0.027657	0.023998	0.018424	0.0079
eil51b1_pp_2d_010_30	-	-	-	-	-	0	0.03002	**0.061249**	0.036549	0.000435
eil51b2_pp_2d_010_30	-	-	-	-	-	0.37616	0.365801	0.369478	**0.398487**	0.159026
Vitórias/Empates	**6**	3	0	3	0	**7**	3	2	2	2

4.3 Interface Gráfica

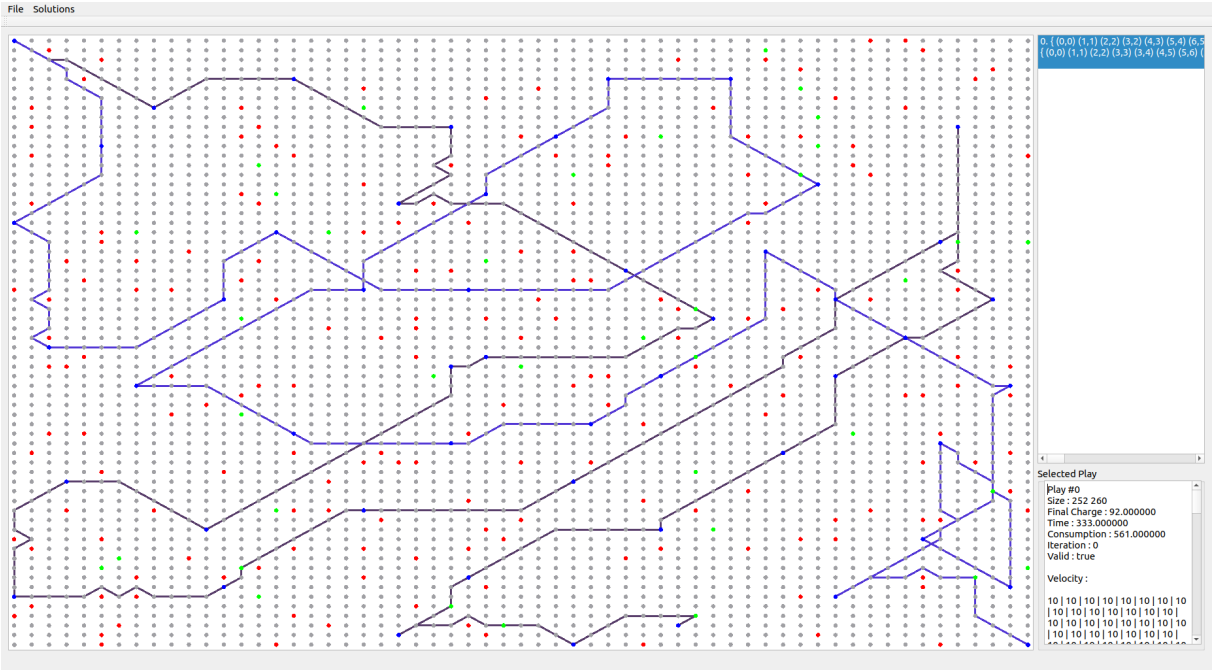

Figura 4.2: Interface gráfica

Uma interface gráfica, demonstrada na Figura 4.2, foi implementada de forma a facilitar a visualização das soluções encontradas. Na lateral direita, temos dois painéis, o superior que apresenta uma lista de soluções, representando o pool gerado pela execução atual. A solução selecionada nesse paiinel é demonstrada no canvas principal. O painel inferior apresenta as principais informações da solução selecionada, como o valor das funções objetivo, a velocidade dos drones em cada trecho e a taxa de recarga do drone em cada estação de energia.

No canvas principal, temos a grid, constituída por pontos vermelhos (áreas proibidas), pontos verdes (estações de energia), pontos azuis (clientes) e pontos cinzas (pontos sem funcionalidade especial).

Capítulo 5

Conclusões

Neste trabalho, abordamos o PMORVDG, considerando um leque de restrições que a literatura não tem explorado com frequência, além de utilizar um modelo com várias funções objetivo. O multi-objetivo, grid, restrições de áreas proibidas (docking constraint), a preocupação com o consumo (Green Computing) e o dinamismo deste problema mostram uma abordagem muito prática, para aplicações reais. As instâncias consideraram posições iniciais arbitrárias de drones e também capacidades iniciais de bateria variáveis, de modo que é possível integrar essa ferramenta em um solucionador on-line que resolve uma série de instâncias considerando mudanças nas características dinâmicas do cenário (devido a condições de vento, logística e outras restrições operacionais).

Para solucionar o problema abordado, propomos três algoritmos (BRKGA, G-VND e G-MOVND) e através dos experimentos computacionais realizados, podemos concluir que para o PMORVDG, o BRKGA, apesar de gerar soluções viáveis de maneira muito rápida, suas soluções geradas perdem em qualidade para os outros métodos. Já, na comparação GVND (mono-objetivo) e GMOVND (multiobjetivo), podemos ver que o G-VND gera as melhores soluções mas demora a encontrar soluções válidas. Dessa forma, o G-MOVND, transita entre os outros dois métodos quanto a vantagens e desvantagens, sendo mais rápido que o método mono-objetivo a encontrar soluções válidas e gerando soluções melhores que as do BRKGA.

Este trabalho também mostra um potencial de crescimento se aproximando cada vez mais de uma situação real. Assim, visualizamos trabalhos futuros levando em conta mais camadas de grade representando o movimento vertical do drone. Além disso, um problema com drones heterogêneos e pontos temporários proibidos são algumas outras variáveis que poderiam ser exploradas em trabalhos futuros envolvendo esse problema.

Próximos passos para este trabalho também requerem um estudo de métodos exatos para este problema com um e vários drones, constituído de formulações matemáticas e métodos híbridos conjugando metaheurísticas com métodos exatos (geração de colunas) para o caso de múltiplos drones.

Métodos de solução para o problema baseados em redes neurais também estão em nossos planos. Poderemos, assim, tratar o problema em tempo real, no qual não seria gerada uma rota prévia, mas ela iria se formando dinamicamente, transformando, então, os veículos em inteligentes.

Referências

[1] Adabo, G. J. Long range unmanned aircraft system for power line inspection of brazilian electrical system. *Journal of Energy and Power Engineering 8*, 2 (2014).

[2] Agatz, N.; Bouman, P.; Schmidt, M. Optimization approaches for the traveling salesman problem with drone. *Transportation Science 52*, 4 (2018), 965–981.

[3] Bean, J. C. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing 6*, 2 (1994), 154–160.

[4] Coelho, B. N.; Coelho, V. N.; Coelho, I. M.; Ochi, L. S.; Zuidema, D.; Lima, M. S.; da Costa, A. R., et al. A multi-objective green uav routing problem. *Computers & Operations Research 88* (2017), 306–315.

[5] Coelho, V. N.; Coelho, I. M.; Coelho, B. N.; Reis, A. J.; Enayatifar, R.; Souza, M. J.; Guimarães, F. G. A self-adaptive evolutionary fuzzy model for load forecasting problems on smart grid environment. *Applied Energy 169* (2016), 567 – 584.

[6] Coelho, V. N.; Grasas, A.; Ramalhinho, H.; Coelho, I. M.; Souza, M. J.; Cruz, R. C. An ils-based algorithm to solve a large-scale real heterogeneous fleet vrp with multi-trips and docking constraints. *European Journal of Operational Research 250*, 2 (2016), 367–376.

[7] Deng, C.; Wang, S.; Huang, Z.; Tan, Z.; Liu, J. Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications. *J. Commun 9*, 9 (2014), 687–692.

[8] Duarte, A.; Pantrigo, J. J.; Pardo, E. G.; Mladenovic, N. Multi-objective variable neighborhood search: an application to combinatorial optimization problems. *Journal of Global Optimization 63*, 3 (Nov 2015), 515–536.

[9] Erdoğan, S.; Miller-Hooks, E. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review 48*, 1 (2012), 100–114.

[10] Floreano, D.; Wood, R. J. Science, technology and the future of small autonomous drones. *Nature 521*, 7553 (2015), 460.

[11] Fonseca, C. M.; Paquete, L.; López-Ibánez, M. An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE international conference on evolutionary computation* (2006), IEEE, pp. 1157–1163.

[12] Gonçalves, J. F.; Resende, M. G. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics 17*, 5 (2011), 487–525.

[13] Gutin, G.; Punnen, A. P. *The traveling salesman problem and its variations*, vol. 12. Springer Science & Business Media, 2006.

[14] Haala, N.; Cramer, M.; Weimer, F.; Trittler, M. Performance test on uav-based photogrammetric data collection. *Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 38*, 1/C22 (2011), 7–12.

[15] Harris, A.; Sluss, J. J.; Refai, H. H.; LoPresti, P. G. Alignment and tracking of a free-space optical communications link to a uav. In *Digital Avionics Systems Conference, 2005. DASC 2005. The 24th* (2005), vol. 1, IEEE, pp. 1–C.

[16] Irizarry, J.; Gheisari, M.; Walker, B. N. Usability assessment of drone technology as safety inspection tools. *Journal of Information Technology in Construction (ITcon) 17*, 12 (2012), 194–212.

[17] Laporte, G. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research 59*, 3 (1992), 345–358.

[18] Máthé, K.; Buşoniu, L. Vision and control for uavs: A survey of general methods and of inexpensive platforms for infrastructure inspection. *Sensors 15*, 7 (2015), 14887–14916.

[19] Metni, N.; Hamel, T. A uav for bridge inspection: Visual servoing control law with orientation limits. *Automation in construction 17*, 1 (2007), 3–10.

[20] Mladenović, N.; Hansen, P. Variable neighborhood search. *Computers & operations research 24*, 11 (1997), 1097–1100.

[21] Nigam, N.; Kroo, I. Persistent surveillance using multiple unmanned air vehicles. In *Aerospace Conference, 2008 IEEE* (2008), IEEE, pp. 1–14.

[22] Nilsson, N. J. *The quest for artificial intelligence*. Cambridge University Press, 2009.

[23] Resende, M. G.; Ribeiro, C. C. Grasp: Greedy randomized adaptive search procedures. In *Search methodologies*. Springer, 2014, pp. 287–312.

[24] Schermer, D.; Moeini, M.; Wendt, O. A variable neighborhood search algorithm for solving the vehicle routing problem with drones. Tech. rep., Technical Report Technische Universität Kaiserslautern, 2018.

[25] Talbi, E.-G. *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.

[26] Vansteenwegen, P.; Souffriau, W.; Sörensen, K. The travelling salesperson problem with hotel selection. *Journal of the Operational Research Society 63*, 2 (2012), 207–217.

[27] Voorneveld, M. Characterization of pareto dominance. *Operations Research Letters 31*, 1 (2003), 7–11.

[28] Wang, X.; Poikonen, S.; Golden, B. The vehicle routing problem with drones: several worst-case results. *Optimization Letters 11*, 4 (2017), 679–697.