Algoritmo VNS para o Problema de Roteamento Híbrido com Veículo-Drone para Serviço de Entrega e Coleta

Anderson Zudio[1], **Igor Machado Coelho**[1], **Luiz Satoru Ochi**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (IC-UFF)
Av. Gal. Milton Tavares de Souza, s/nº Campus Praia Vermelha – Niterói/RJ – Brasil

azudio@id.uff.br, {imcoelho, satoru}@ic.uff.br

Abstract. *The Hybrid Vehicle drone Routing Problem (HVDRP) is a logistic problem with many applications in smart cities, like surveillance and package transportation. The HVDRP models a vehicle equipped with many drones to serve customers with delivery and pickup demands. The vehicle serves as a mobile depot for the drones, limited by carrying capacity and flight distance. In this context, this work presents a Variable Neighborhood Search (VNS) procedure for HVDRP. The proposed algorithm was computationally tested with publicly available instances. The computational result shows that the proposed VNS achieves superior or equivalent solutions to another method proposed in the literature while achieving optimal solutions for small instances.*

Resumo. *O problema de roteamento híbrido com veículo-drone para serviço de coleta e entrega (HVDRP) modela diversas aplicações industriais de logística em cidades inteligentes, como serviços de monitoramento e de transporte de pacotes. No HVDRP, um veículo é equipado com vários drones para servir clientes com demandas de coleta e entrega. O veículo transita entre estações para efetuar o despacho dos drones. Este trabalho apresenta uma abordagem baseada no Variable Neighborhood Search (VNS) para o HVDRP, que foi submetido a um teste computacional com instâncias disponíveis publicamente. Os resultados mostram que o VNS proposto obtém soluções equivalentes ou superiores quando comparado com as de outra abordagem da literatura e que as soluções das pequenas instâncias são ótimas.*

1. Introdução

O problema de roteamento híbrido com veículo-drone para serviço de entrega e coleta é uma extensão do problema clássico de roteamento de veículos amplamente conhecido [Golden et al. 2008], esses problemas são abreviados respectivamente do inglês como HVDRP e VRP. O VRP foi um dos primeiros problemas a ser caracterizado como NP-Difícil [Solomon 1987], desta forma, como o HVDRP pode ser reduzido polinomialmente para o VRP, ele também é da classe NP-Difícil. O HVDRP usa um veículo equipado com múltiplos drones para servir múltiplos clientes que precisam ser atendidos com serviços de entrega e coleta. Com a evolução dos drones, eles podem ser usados em diversas atividades relacionadas a cidades inteligentes, por exemplo, transporte [Menouar et al. 2017], segurança pública [Chowdhury et al. 2017], preservação florestal [Getzin et al. 2012], assistência médica [Kim et al. 2017] e monitoramento [Da Silva et al. 2017]. Particularmente, o uso de drones para logísticas que

envolvem entrega ou coleta de produtos aumentou substancialmente nos últimos anos [Murray and Raj 2020, Taniguchi et al. 2020, Moshref-Javadi et al. 2020].

No contexto de cidades inteligentes, o HVDRP é um problema de logística que modela inúmeras aplicações práticas, como serviços de entrega, de supervisão, de monitoramento, de irrigação e qualquer outra que envolve um sistema com um único veículo e múltiplos drones de vários tipos. O HVDRP provê um modelo que difere de outros sistemas que utilizam somente um drone, como o problema do cacheiro viajante com ajudante voador [Murray and Chu 2015] e o problema de roteamento de veículos com dois escalões [Luo et al. 2017, Kitjacharoenchai et al. 2020, Liu et al. 2020]. O HVDRP foi inicialmente proposto por [Karak and Abdelghany 2019, Karak 2020], onde os autores resolveram o problema computacionalmente com heurísticas e uma modelagem matemática de programação linear inteira mista (MILP). Recentemente, o trabalho [Zudio et al. 2021] propôs um BRKGA para o HVDRP, que utiliza uma heurística construtiva na fase de decodificação da meta-heurística aplicada. Além disso, os autores do último trabalho citado disponibilizam instâncias públicas para HVDRP com algumas soluções de uma implementação própria do MILP proposto por [Karak and Abdelghany 2019]. No âmbito de cidades inteligentes, este trabalho apresenta uma heurística baseada na meta-heurística *Variable Neighborhood Search* (VNS) [Hansen et al. 2019] para resolver o HVDRP, comparando os resultados computacionais obtidos com o BRKGA mencionado.

Meta-heurísticas são algoritmos para problemas de otimização que aplicam estratégias que independem do mesmo. Elas são compostas por estratégias e mecanismos que são capazes de achar soluções sem se restringir a ótimos locais. Na literatura, as meta-heurísticas são abordagens comuns para solucionar problemas de otimização caracterizados como NP-Difícil com instâncias razoavelmente grandes em relação às aplicações práticas [Epstein and Stee 2007, Glover and Kochenberger 2006, Gendreau et al. 2010, Taillard 2016].

O HVDRP trabalha com um veículo equipado com drones heterogêneos que servem múltiplos clientes que demandam serviços de entrega e coleta. Uma instância do HVDRP é descrita por um grafo direcionado $G = \{N, A\}$ que detalha uma rede multimodal, um conjunto D de drones e um conjunto N_c de clientes. O conjunto de nós $N = \{\{n_d\} \cup N_v \cup N_c\}$ inclui o ponto de partida n_d, um conjunto de estações N_v e um conjunto de clientes N_c. As estações e o ponto de partida somente podem ser acessados pelo veículo, e os clientes somente podem ser visitados pelos drones. O conjunto de arcos $A = (i, j)$, $i, j \in N$, descreve cada ligação entre os pontos da rede. Cada arco deve ter uma distância associada. Os custos de travessia do veículo e de cada drones são especificados como um fator multiplicativo em função da distância de cada arco. Cada drone tem um alcance máximo r_d e uma capacidade de carga w_d, $d \in D$. Cada cliente tem dois pesos associados p_i e q_i referentes aos pacotes de coleta e de entrega respectivamente, $i \in N_c$. O objetivo do HVDRP é minimizar o custo de travessia do veículo e dos drones para servir todos os clientes.

O veículo pode ser visto como um depósito móvel para carregar os drones. O ponto de partida n_d é uma estação depósito em que não ocorre despacho de drones. Uma vez que o veículo volte ao depósito, todos os drones devem estar dentro dele. O veículo pode visitar cada estação de N_v somente uma vez conforme necessário. Os drones podem ser despachados quando o veículo está estacionado em alguma estação. Cada drone pode

servir múltiplos clientes em uma mesma rota e podem voltar para a mesma estação de partida ou alguma outra no caminho adiante do veículo. A capacidade de peso de cada drone e o seu alcance máximo devem ser respeitados. O veículo pode despachar vários drones de uma mesma estação. Assume-se que a bateria dos drones são trocadas no final de cada voo e que os pacotes de coleta e entrega são administrados imediatamente quando eles voltam para o veículo. Um mesmo drone pode ser despachado e coletado múltiplas vezes de uma estação. O veículo não pode sair da estação até que todos os voos programados sejam concluídos. Finalmente, qualquer drone que retornar para uma estação a frente do veículo deve aguarda-lo.

Para sumarizar o problema, a seguinte lista de itens revisa o parágrafo anterior:

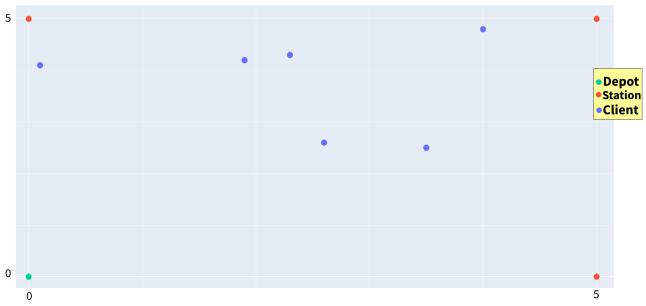
1. Os clientes são descritos por sua localização e sua demanda de coleta e entrega. Cada pacote tem um peso associado. Somente os drones servem os clientes.
2. Os drones são enviados de estações quando o veículo está estacionado. Eles podem voar múltiplas vezes a partir de uma mesma estação. Drones diferentes podem voar simultaneamente.
3. O veículo pode transitar somente entre as estações, visitando elas uma única vez. Não é preciso visitar todas as estações.
4. Cada drone é descrito por sua capacidade de carga e seu alcance máximo. Eles podem visitar vários clientes durante um mesmo voo, porém os limites dos drones devem ser respeitados.
5. O drone que está realizando um voo pode retornar para qualquer estação ao longo do caminho do veículo ou para a localização em que ele está estacionado.
6. Quando um drone retorna para o veículo, a bateria é trocada imediatamente e todos os pacotes são manipulados apropriadamente para o próximo voo e para o estoque.
7. O veículo espera os drones e vice versa. Consequentemente, nenhum drone pode ser deixado para trás.

As Figuras 1 e 2 ilustram duas instâncias do HVDRP com duas possíveis soluções sem os detalhes dos drones e dos clientes. Os vértices coloridos de vermelho são estações para o veículo, os azuis representam clientes e o verde é o depósito. Na solução, o caminho do veículo é assinalado com a cor vermelha e o voo dos drones são representados com verde.

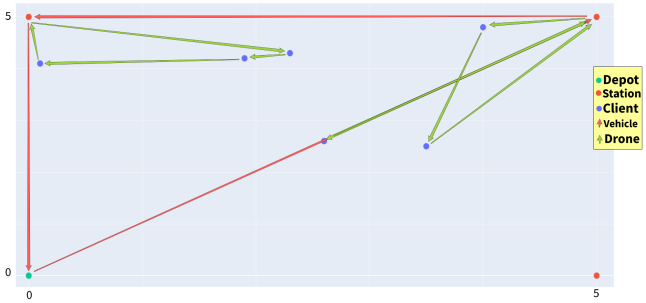
O restante deste trabalho é estruturado da seguinte forma: a Seção 2 descreve o algoritmo VNS proposto, a Seção 3 mostra os resultados computacionais obtidos com a heurística proposta e a Seção 4 conclui este trabalho.

2. Algoritmo VNS

A abordagem proposta neste trabalho para o HVDRP utiliza o *General Variable Neighborhood Search* (GVNS), que é uma variante do VNS básico [Hansen et al. 2019]. Nesta variante, o *Variable Neighborhood Descent* (VND) é utilizado na etapa de busca local da iteração do VNS. A ideia é utilizar estruturas de vizinhanças que alteram a rota do veículo durante a fase de *shake* e vizinhanças que ajustam o voo dos drones durante o VND. O Algoritmo 1 detalha o pseudocodigo do VNS com VND seguindo o trabalho citado neste parágrafo. Os valores k, k_M são os parâmetro que controlam as vizinhanças do veículo e k'_M é uma variável que controla as vizinhanças do drone.

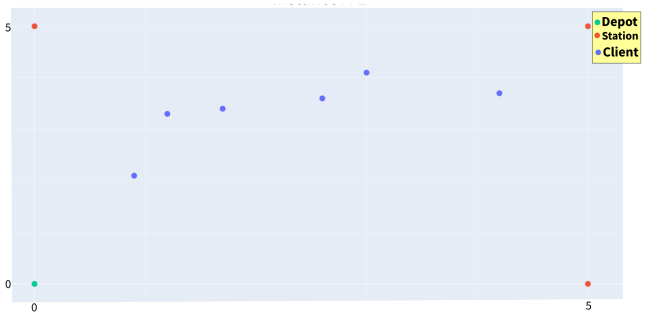


(a) Instância A-1.
Fonte: originalmente publicada em [Zudio et al. 2021].

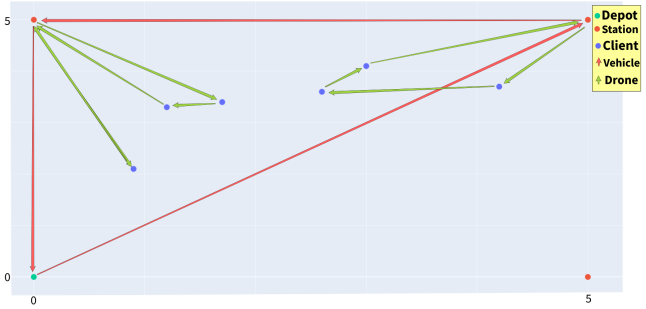


(b) Exemplo de solução para A-1.
Fonte: originalmente publicada em [Zudio et al. 2021].

Figura 1. Instância A-1 do HVDRP com uma solução exemplo.



(a) Instância A-2. Fonte: originalmente publicada em [Zudio et al. 2021].



(b) Exemplo de solução para A-2.
Fonte: originalmente publicada em [Zudio et al. 2021].

Figura 2. Instância A-2 do HVDRP com uma solução exemplo.

Algoritmo 1 Pseudcodigo do VNS com VND.

1: **function** GVNS(**Solução inicial:** x, **Inteiros:** k_{MAX}, k'_{MAX})
2: $x^* \leftarrow$ x ▷ Melhor solução.
3: **while** Critério de parada não atingido **do**
4: $k \leftarrow 1$ ▷ Controla a vizinhança do veículo.
5: **while** $k \leq k_{MAX}$ **do**
6: $x \leftarrow$ SHAKE(x^*, k)
7: $x \leftarrow$ VND(x, k'_M) ▷ A variável k'_M controla as vizinhanças do drone.
8: NEIGHBORHOODCHANGE(x, x^*, k, k_M)
return Melhor solução encontrada x^*

A ideia principal desta aplicação do VNS é realizar iterações que alteram a rota do veículo até que um critério de parada seja atingido, retornando a melhor solução encontrada com boas configurações de voo para os drones. O método de *shake* altera a melhor solução encontrada em uma variável local usando estruturas de vizinhanças que mudam a rota do veículo. Como o veículo pode opcionalmente visitar estações no HVDRP, a ideia é adicionar ou remover estações para despachar os drones enquanto alterna-se a ordem de visita delas. Já na etapa do VND, os voos são submetidos a diversas estruturas de vizinhanças que alteram a ordem de visita dos clientes, o drone que é utilizado e as estações usadas. Dessa forma, a solução é amplamente explorada em relação a sequência e a disponibilidade das estações.

2.1. Solução Inicial

Antes de iniciar a busca com o VNS, uma solução inicial deve ser construída. Uma vez que o HVDRP tem limitações de voo e capacidade de carga nos drones, soluções inviáveis podem ser formuladas através de voos irregulares que ultrapassam os limites de algum drone. Dessa forma, a heurística construtiva proposta é um algoritmo guloso que forma soluções iniciais para o VNS com configurações viáveis. A ideia principal deste algoritmo guloso é visitar as estações e os clientes baseados na ideia de vizinho mais próximo.

Uma iteração do algoritmo guloso para o HVDRP inicia com um movimento do veículo e posteriormente move os drones. O movimento do veículo é um processo que guia o mesmo para a estação com menor custo de visita partindo da localização atual. Então, a primeira iteração do algoritmo guloso move o veículo do depósito para a estação mais próxima e as demais verificam qual é o vizinho mais próximo. Uma vez que o veículo estaciona em uma estação, a segunda parte do processo iterativo é servir os clientes que estão no alcance dos drones.

Um drone é enviado para um cliente a cada iteração da segunda parte baseado no menor custo. Somente movimentos viáveis de drones são considerados nesta etapa. O custo de visita para o próximo cliente é calculado de acordo com a localização atual de cada drone. No fim de cada iteração, se um drone não consegue realizar uma visita, ele é retornado para o veículo. Este processo iterativo para quando todos os clientes são visitados ou quando não for possível movimentar qualquer drone. Assim, resumi-se o processo de iteração principal que realiza um novo movimento para o veículo.

Sabendo que não há mais como enviar drones para os clientes disponíveis, o veículo se movimenta para a próxima estação. Esse processo iterativo termina quando

todos os clientes são servidos. Finalmente, o veículo retorna para o depósito. O Algoritmo 2 sumariza a heurística construtiva proposta para gerar as soluções iniciais usadas no VNS deste trabalho. A função de *menor_custoV* retorna o próximo movimento do veículo baseado no vizinho mais próximo, o *movimenta_drone* realiza um movimento de drone enquanto atualiza sua localização, o *menor_custoD* retorna o próximo movimento de drone considerando todos os drones e *retorna_drones* envia todos os drones que estão em voo de volta para o veículo.

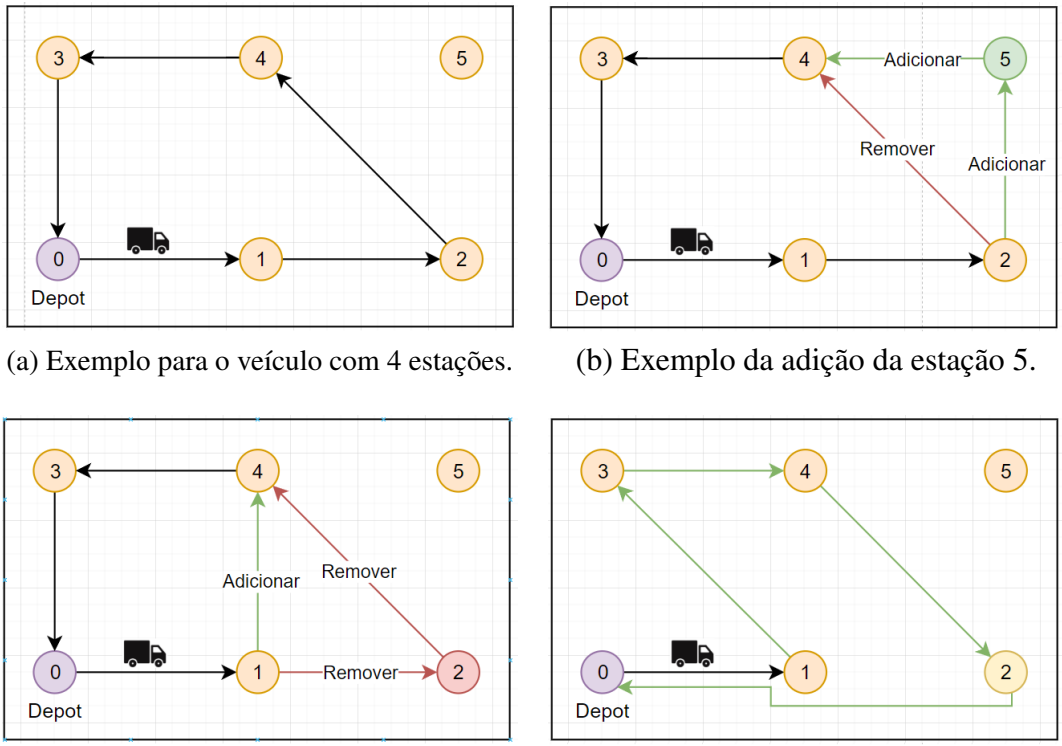
Algoritmo 2 Algoritmo guloso para o HVDRP.

function GULOSO(**Grafo:** $G(N, A)$, **Estações:** N_v, **Depósito:** n_d, **Clientes:** N_c)
  $S \leftarrow N_v$ ▷ Estações disponíveis.
  $V_{path} \leftarrow n_d$ ▷ Começa do depósito.
  $C \leftarrow N_c$ ▷ Clientes disponíveis.
  while S ou C não estão vazios **do** ▷ Ainda tem clientes para servir.
    $s \leftarrow$ MENOR_CUSTOV(G, S, posição atual do veículo)
    $V_{path} \leftarrow V_{path} \cup \{s\}$ ▷ Constrói a solução.
    $S \leftarrow S - s$ ▷ A estação s foi visitada.
    $Moves \leftarrow \emptyset$ ▷ Lista de movimentos.
    while C não está vazio e há pelo menos um movimento de drone **do**
      $d, c \leftarrow$ MENOR_CUSTOD(G, $Move$, C) ▷ Elege o drone d para o cliente c.
      $m \leftarrow$ MOVIMENTA_DRONE(G, d, c) ▷ Realiza o movimento do drone.
      $Moves \leftarrow Moves \cup \{m\}$ ▷ Registra o movimento.
      $C \leftarrow C - c$ ▷ O cliente está indisponível.
    RETORNA_DRONES(G, $Moves$) ▷ Retorna os drones que estão voando.
    $D_{path} \leftarrow D_{path} \cup Moves$ ▷ Constrói a solução.
  $V_{path} \leftarrow n_d$ ▷ Envia o veículo de volta.
  return V_{path} e D_{path}

2.2. Estruturas de vizinhança para a fase pertubação

A fase de pertubação, denotada pela função *Shake* do Algoritmo 1, objetiva mudar a solução de entrada através de adições, remoções e troca de estações. O parâmetro k_M que controla o número máximo de vizinhanças é fixado em 5. O *Shake* aplica uma quantidade de movimentos de acordo com a quantidade $v = |V_{path}|$ de estações que o veículo visita na solução que está sofrendo alteração. As seguintes estruturas de vizinhanças são aplicadas:

- Para $k = 1$, o movimento aplicado é selecionar uma estação aleatoriamente. Caso seja uma que já está presente na rota do veículo, a estação é removida. Caso contrário, ela é adicionada no caminho do veículo em uma posição aleatória. Esse movimento é aplicado $\frac{v}{2}$ vezes.
  - Quando uma estação é adicionada, os voos configurados não são alterados.
  - Quando uma estação é removida, os voos configurados nessa estação são alterados para operar na estação imediatamente posterior. Caso a estação removida seja a final, então a imediatamente anterior é utilizada.
- Para $2 \leq k \leq 5$, o movimento aplicado consiste em trocar duas estações no caminho do veículo. Esse movimento é aplicado v vezes.
  - Todos os voos configurados para as estações trocadas são mantidos.



(a) Exemplo para o veículo com 4 estações.

(b) Exemplo da adição da estação 5.

(c) Exemplo de remoção de uma estação.

(d) Exemplo de troca de duas estações.

Figura 3. Exemplo de operações com o veículo.

A Figura 3 exemplifica as operações aplicadas no caminho do veículo. A primeira figura mostra um caminho exemplo inicial, a segunda figura mostra a operação de adição de uma estação, a terceira figura apresenta um exemplo de remoção de uma estação e a última figura apresenta uma troca de estações no caminho original. Os arcos alterados são destacados nas com as cores verde e vermelha nas figuras. Vale a pena destacar que as vizinhanças podem construir soluções inviáveis. Nestes casos, a avaliação da solução recebe uma grande penalidade dependendo do número de voos que são inviáveis.

2.3. Estruturas de vizinhança para a fase de exploração

O principal objetivo desta fase é obter configurações boas de voo a partir da nova sequência de estações provenientes do *Shake*. Para isso, o VND clássico é utilizado com as estruturas de vizinhança que alteram as configurações dos voos dos drones. O parâmetro k'_M é fixado em 6, de modo que todas as vizinhanças são aplicadas sequencialmente. As seguintes estruturas de vizinhanças são utilizadas em sequência aleatória decidida antes de iniciar o VND em cada iteração:

- *Adicionar cliente* - O movimento consiste em adicionar um cliente em uma determinada rota. A rota que visitava o cliente originalmente é reconfigurada para ignorar o respectivo cliente. Qualquer rota sem clientes é removida da solução.
- *Mudar drone* - Sabendo que a configuração dos drones do HVDRP é heterogênea, o movimento desta estrutura é mudar o drone que vai realizar o voo.
- *Unir voos* - O movimento desta estrutura é unir dois voos que podem partir de estações distintas. A escolha de estação inicial, final e do drone é feita de acordo com a configuração que tiver menor custo.

- *Mudar final* - O movimento aplicado é mudar a estação final para outra no caminho do veículo.
- *Mudar começo* - O movimento muda a estação inicial do voo para outra no caminho do veículo.
- *Trocar clientes* - O movimento é trocar dois clientes de sequência dentro de um mesmo voo.

Assim como especificado na seção anterior, vale notar que estas vizinhanças também podem gerar configurações inviáveis. Nestes casos, a avaliação da solução sofre penalidades baseado na quantidade de voos que não podem acontecer devido às limitações dos drones. Cada voo inviável é penalizado uma vez.

3. Resultados Computacionais

Essa seção mostra os resultados computacionais obtidos com o VNS proposto na Seção 1. Os autores do trabalho original [Karak and Abdelghany 2019] não providenciam as instâncias utilizadas para o HVDRP e também não disponibilizam as implementações dos métodos propostos [1]. Dessa forma, o VNS foi submetido a um teste computacional com as instâncias usadas em [Zudio et al. 2021][2], as quais são classificadas em sete categorias identificadas com as letras de A até G com dez instâncias cada. O teste empírico deste trabalho usou as cinco primeiras instâncias das categorias de A até D.

A configuração destas instâncias é baseada no trabalho de [Karak and Abdelghany 2019], que organiza as estações em forma de um *grid* de tamanho 5 em um plano euclideano. O depósito é localizado na origem e cada estação do veículo é posicionada nas interseções deste *grid*. Os clientes são posicionados aleatoriamente em volta das estações com demandas de entrega e coleta de pesos aleatórios no intervalo $[0, 5]$. Cada instância possui dois drones para servir os clientes com capacidade máxima de carga de 10 unidades e um alcance máximo de 7 unidades. O custo para o veículo atravessar cada arco é a própria distância e o custo para voo do drone é duas vezes o valor da distância. A Tabela 1 sumariza as categorias usadas.

Identificação	Número de clientes	Número de estações	Área
A-1 até A-5	6	3	25
B-1 até B-5	50	8	100
C-1 até C-5	50	15	300
D-1 até D-5	50	24	400

Tabela 1. Configuração das instâncias utilizadas no teste computacional.

Os resultados do teste empírico deste trabalho obtido com o VNS proposto foram comparados com os resultados do BRKGA proposto em [Zudio et al. 2021]. Ambos os algoritmos foram implementados com a linguagem de programação *C++17* e compilada com o *GCC* do *GNU Compiler collection* usando a *flag -O3*. Ambas as aplicações utilizam o *MersenneTwister*[Matsumoto and Nishimura 1998] para gerar números pseudoaleatórios. Além disso, uma implementação do modelo de programação linear inteira

[1]Os autores deste trabalho contactaram os autores de [Karak and Abdelghany 2019] para obter as instâncias sem sucesso

[2]As instâncias e o gerador estão disponíveis em `https://github.com/AndersonZM/hvdrp`.

mista (MILP) de [Karak and Abdelghany 2019] foi utilizada para obter o ótimo para as instâncias da categoria A. A implementação do MILP foi feita através da API de *C++* do Gurobi versão 9.1.2. Todos os resultados obtidos com todas as implementações usaram o mesmo recurso computacional munido com uma CPU Intel Core i7-10700f @2.9 GHz, 32 GB de RAM e sistema operacional Ubuntu 20.4 LTC (x64). Portanto, este trabalho compara os algoritmos somente através da qualidade da solução obtida para o conjunto de instâncias estudado.

As implementações do VNS e do BRKGA foram executadas 30 vezes para cada instância de forma independente usando sementes distintas no intervalo de $[1, 30]$ para controlar o âmbito dos testes. Os parâmetros usados no VNS estão especificados na Seção 1. Os valores para o BRKGA seguem os mesmos detalhados em [Zudio et al. 2021]: população máxima $P_{max} = 100$, TOP $\epsilon = 0{,}2$, BOT $\omega = 0{,}15$ e taxa de elitismo $\rho = 0{,}7$. O critério de parada para ambos os algoritmos foi de 5 segundos. O MILP foi executado somente uma vez para cada instância da categoria A até encontrar o ótimo. Essa categoria é composta por instâncias pequenas em relação as demais.

A Tabela 2 mostra os resultados do teste computacional para as instâncias da categoria A. A primeira coluna mostra o nome da instância, a segunda coluna o custo ótimo e as demais colunas mostram os resultados para o VNS e o BRKGA. A média reportada na tabela é o custo médio obtido das 30 execuções. A Tabela 3 mostra o resultado para as demais instâncias seguindo o mesmo padrão da tabela anterior, porém sem o resultado do MILP. Finalmente, o MILP usou 16 *threads* do recurso computacional mencionado, o BRKGA e o VNS são implementações sequenciais que usam uma única *thread*.

Tabela 2. Resultados para as instâncias de categoria A.

Instância	MILP	VNS		BRKGA	
	Ótimo	Melhor Solução	Média	Melhor Solução	Média
A-1	53,6	53,6	53,6	53,6	53,6
A-2	49,2	49,2	49,2	49,2	49,2
A-3	35,3	35,3	35,3	35,3	35,3
A-4	46,3	46,3	46,3	46,3	46,3
A-5	37,5	37,5	37,5	37,5	37,5

Os resultados da Tabela 2 mostram que o VNS e o BRKGA são capazes de obter soluções equivalentes ao ótimo para o conjunto de instâncias usado com consistência. Vale enfatizar que todas as soluções para a categoria A foram obtidas com menos de 1 segundo de execução no recurso computacional deste trabalho com ambas implementações. Este resultado mostra que os métodos estudados neste trabalho são capazes de obter soluções de alta qualidade para instâncias com poucos clientes. A Tabela 3 mostra que o VNS obteve um custo menor que a melhor solução obtida pelo BRKGA em 10 instâncias do conjunto, tal que as demais são equivalentes ao melhor encontrado pelo BRKGA. Além disso, os resultados também mostram que a qualidade média obtida pelo VNS proposto são consistentemente melhores que os obtidos pelo BRKGA para este grupo de instâncias. Vale notar que o custo médio de ambos os algoritmos estão próximos à melhor solução, indicando que os métodos encontram soluções de boa qualidade com consistência.

Tabela 3. Resultados para as instâncias das categoria B até D.

Instância	VNS		BRKGA	
	Melhor Solução	Média	Melhor Solução	Média
B-1	**199.0**	201.7	201.7	202.0
B-2	198.5	198.5	198.5	199.0
B-3	**177.0**	179.2	179.1	181.4
B-4	**202.9**	203.9	204.0	204.8
B-5	231.8	231.9	231.8	233.1
C-1	**219.0**	221.2	221.1	223.5
C-2	**255.0**	255.1	258.8	260.0
C-3	237.5	238.0	237.5	238.1
C-4	237.7	237.7	237.7	237.7
C-5	**240.7**	243.3	243.9	245.8
D-1	**347.5**	352.3	351.3	360.8
D-2	**397.1**	405.3	400.1	410.5
D-3	**330.2**	349.1	335.4	352.2
D-4	237.7	238.9	237.7	240.1
D-5	**329.2**	331.2	330.4	335.0

Os resultados em **negrito** são os valores que foram melhorados pelo VNS para a melhor solução.

4. Conclusão

Este trabalho apresentou um algoritmo baseado na meta-heurística VNS com VND [Hansen et al. 2019] para o problema de roteamento híbrido com veículo-drone para serviço de entrega e coleta (HVDRP). No contexto de cidades inteligentes, o HVDRP é um problema de logística com diversas aplicações industriais, como transporte de pacotes e monitoramento. O problema modela casos onde um único veículo é usado como um depósito móvel para múltiplos drones servirem clientes com demandas de entrega e coleta. O VNS proposto utiliza uma heurística gulosa para obter as soluções iniciais e a ideia principal é utilizar a fase de pertubação da meta-heurística para obter sequências de visitas de estações para o veículo enquanto a etapa de busca com VND configura os voos dos drones.

O algoritmo proposto foi submetido a um teste computacional com 20 instâncias e comparado com uma abordagem que utiliza BRKGA originalmente introduzida em [Zudio et al. 2021]. Além disso, uma modelagem de programação linear inteira mista foi usada para obter soluções ótimas para as instâncias pequenas do conjunto usado. Os resultados deste teste empírico mostraram que o VNS consegue obter a solução ótima para o conjunto de instâncias pequenas em pouco tempo de execução. Além disso, os resultados também mostram que o VNS obtém com consistência soluções melhores ou equivalentes ao BRKGA nas demais instâncias. Das 15 instâncias utilizadas com um número alto de estações, o VNS melhorou 10 resultados quando comparado ao melhor encontrado pelo BRKGA. Todas as implementações do VNS e do BRKGA foram testadas em um mesmo recurso computacional através de parâmetros iguais para o tempo de execução e quantidade de execuções para cada instância, a fim de manter a comparação justa.

Como trabalho futuro, objetivamos aumentar a quantidade de instâncias e utilizar

diferente topologias para testar as heurísticas propostas. Também desejamos aprimorar o VNS proposto e utilizar recursos computacionais híbridos de CPU-GPU para obter resultados com a versão paralela do algoritmo.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Os resultados computacionais apresentados neste trabalho foram obtidos usando os recursos do Laboratório de Inteligência Computacional (LABIC) do Instituto de Computação da Universidade Federal Fluminense (IC-UFF). Os conceitos usados em cada implementação foram baseados no OptFrame[3] [Coelho et al. 2011].

Referências

[Chowdhury et al. 2017] Chowdhury, S., Emelogu, A., Marufuzzaman, M., Nurre, S. G., and Bian, L. (2017). Drones for disaster response and relief operations: A continuous approximation model. *International Journal of Production Economics*, 188:167–184.

[Coelho et al. 2011] Coelho, I. M., Munhoz, P. L. A., Haddad, M. N., Coelho, V. N., de Melo Silva, M., Souza, M. J. F., and Ochi, L. S. (2011). Optframe: a computational framework for combinatorial optimization problems. In *Proceedings of VII ALIO/EURO*, pages 51–54, Porto. Workshop on Applied Combinatorial Optimization, VII, 2011, Porto, ALIO-EURO.

[Da Silva et al. 2017] Da Silva, L. C. B., Bernardo, R. M., De Oliveira, H. A., and Rosa, P. F. (2017). Multi-uav agent-based coordination for persistent surveillance with dynamic priorities. In *2017 International Conference on Military Technologies (ICMT)*, pages 765–771. IEEE.

[Epstein and Stee 2007] Epstein, L. and Stee, R. v. (2007). Multidimensional packing problems. In Gonzalez, T., editor, *Handbook of Approximation Algorithms and Metaheuristics*, chapter 35, pages 1–15. Chapman & Hall/CRC, New York, 1 edition.

[Gendreau et al. 2010] Gendreau, M., Potvin, J.-Y., et al. (2010). *Handbook of metaheuristics*, volume 2. Springer.

[Getzin et al. 2012] Getzin, S., Wiegand, K., and Schöning, I. (2012). Assessing biodiversity in forests using very high-resolution images and unmanned aerial vehicles. *Methods in ecology and evolution*, 3(2):397–404.

[Glover and Kochenberger 2006] Glover, F. W. and Kochenberger, G. A. (2006). *Handbook of metaheuristics*, volume 57. Springer, New York.

[Golden et al. 2008] Golden, B. L., Raghavan, S., and Wasil, E. A. (2008). *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media.

[Hansen et al. 2019] Hansen, P., Mladenović, N., Brimberg, J., and Pérez, J. A. M. (2019). Variable neighborhood search. In *Handbook of metaheuristics*, pages 57–97. Springer.

[3]OptFrame é um framework para otimização com código aberto e está disponível em `https://github.com/optframe/optframe`

[Karak 2020] Karak, A. (2020). *Hybrid Vehicle-drone Routing Problem For Pick-up And Delivery Services Mathematical Formulation And Solution Methodology*. PhD thesis, Southern Methodist University, Dallas.

[Karak and Abdelghany 2019] Karak, A. and Abdelghany, K. (2019). The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transportation Research Part C: Emerging Technologies*, 102:427–449.

[Kim et al. 2017] Kim, S. J., Lim, G. J., Cho, J., and Côté, M. J. (2017). Drone-aided healthcare services for patients with chronic diseases in rural areas. *Journal of Intelligent & Robotic Systems*, 88(1):163–180.

[Kitjacharoenchai et al. 2020] Kitjacharoenchai, P., Min, B.-C., and Lee, S. (2020). Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics*, 225:107598.

[Liu et al. 2020] Liu, Y., Liu, Z., Shi, J., Wu, G., and Pedrycz, W. (2020). Two-echelon routing problem for parcel delivery by cooperated truck and drone. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

[Luo et al. 2017] Luo, Z., Liu, Z., and Shi, J. (2017). A two-echelon cooperated routing problem for a ground vehicle and its carried unmanned aerial vehicle. *Sensors*, 17(5):1144.

[Matsumoto and Nishimura 1998] Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30.

[Menouar et al. 2017] Menouar, H., Guvenc, I., Akkaya, K., Uluagac, A. S., Kadri, A., and Tuncer, A. (2017). Uav-enabled intelligent transportation systems for the smart city: Applications and challenges. *IEEE Communications Magazine*, 55(3):22–28.

[Moshref-Javadi et al. 2020] Moshref-Javadi, M., Lee, S., and Winkenbach, M. (2020). Design and evaluation of a multi-trip delivery model with truck and drones. *Transportation Research Part E: Logistics and Transportation Review*, 136:101887.

[Murray and Chu 2015] Murray, C. C. and Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109.

[Murray and Raj 2020] Murray, C. C. and Raj, R. (2020). The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies*, 110:368–398.

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

[Taillard 2016] Taillard, E. (2016). Tabu search. In *Metaheuristics*, pages 51–76. Springer.

[Taniguchi et al. 2020] Taniguchi, E., Thompson, R. G., and Qureshi, A. G. (2020). Modelling city logistics using recent innovative technologies. *Transportation Research Procedia*, 46:3–12.

[Zudio et al. 2021] Zudio, A., Coelho, I. M., and Ochi, L. S. (2021). Biased random key genetic algorithm for the hybrid vehicle-drone routing problem for pick-up and delivery. In *Anais do 15 Congresso Brasileiro de Inteligência Computacional*, pages 1–6, Joinville, SC. XV Brazilian Congress on Computational Intelligence, 2021, SBIC.