Um Algoritmo Híbrido para o Problema de Roteamento de Caminhões e Drones com Janela de Tempo

Eugênio Belizário Ribeiro Faria, Lorenza Leão Oliveira Moreno,
Luciana Brugiolo Gonçalves, Stênio Sã Rosário F. Soares,
Universidade Federal de Juiz de Fora
Rua José Lourenço Kelmer - Campus Universitário, Juiz de Fora - MG
{eugenio.faria, lorenza, lbrugiolo, ssoares}@ice.ufjf.br

RESUMO

O Problema de Roteamento de Veículos com Drones e Janelas de Tempo é uma variação do problema de roteamento de veículos onde alguns clientes podem ser atendidos por caminhões ou drones e outros clientes apenas por caminhão. O objetivo é minimizar o deslocamento total necessário para visitar cada cliente respeitando as restrições de janelas de tempo, capacidade de carga de drones e caminhões, além da autonomia e o alcance dos drones. O uso de drones pode melhorar a qualidade do serviço de entrega, já que permite atender um maior número de clientes simultaneamente e reduz o uso de caminhões e suas rotas, resultando na redução das emissões de gases poluentes. Neste trabalho é apresentada uma abordagem híbrida que combina GRASP e RVND a um modelo de programação matemática. O algoritmo foi comparado com a literatura e os resultados destacam a qualidade da solução obtida pela estratégia híbrida.

PALAVRAS CHAVE. Problema de Roteamento Verde, Roteamento de drones, GRASP — RVND — MIP.

L&T — Logística e Transportes, MH — Metaheuristicas, IC — Inteligência Computacional

ABSTRACT

The Vehicle Routing Problem with Drones and Time Windows is a variation of the vehicle routing problem where some customers can be served by trucks or drones and other customers only by trucks. The goal is to minimize the total cost required to visit each customer while respecting their time windows, drone and truck carrying capacities, drone autonomy and range. The use of drones may improve the quality of the service, since it allows a larger number of customers to be served simultaneously. It also reduces the use of trucks and their routes, reducing the emission of pollutant gases. This work presents a hybrid approach that combines GRASP and RVND to a mathematical programming model. The algorithm was compared with the literature and the results highlight the quality of the solution obtained by this hybrid strategy.

KEYWORDS. Green Routing Problem, drone routing, GRASP - RVND - MIP.

L&T — Logistics and Transport, MH — Metaheuristics, IC — Computational Intelligence

1. Introdução

Com o grande crescimento de compras online, um dos aspectos mais importantes para a escolha de qual *e-commerce* utilizar é a velocidade da entrega dos pedidos [Bertão, 2021]. Assim, realizar entregas de forma rápida é um dos maiores desafios de grandes varejistas. Por isto, diversos pesquisadores estão trabalhando para encontrar formas mais rápidas e mais eficientes para realizar o processo de entrega de mercadorias [Ranieri et al., 2018].

O problema de roteamento de veículos (VRP) é um problema clássico de otimização combinatória amplamente estudado na literatura devido às suas diversas aplicações, sendo uma das principais a redução de custos e de tempo de entrega de mercadorias. O seu objetivo é realizar o atendimento de consumidores por uma frota de veículos, que partem e retornam de um ou mais pontos denominados depósitos. Uma de suas variações mais promissoras visando realizar o processo de entrega de mercadorias de forma mais rápida e eficiente é o problema do roteamento de veículos com drones (VRP-D) [Wang e Sheu, 2019].

No VRP-D, são utilizados caminhões e drones como veículos, sendo a frota de drones distribuída entre os diversos veículos da frota de caminhões. Cada drone só pode ser lançado do depósito ou de um caminhão parado em um cliente, tendo que retornar ao mesmo caminhão, não necessariamente no mesmo ponto de partida.

Já o problema de roteamento de veículos com drones e janelas de tempo (VRP-DTW), tratado inicialmente por Di Puglia Pugliese e Guerriero [2017] e estudado neste trabalho, mantém todas as características do VRP-D, acrescentando janelas de tempo para atendimento dos clientes.

Além da questão da velocidade de entrega, utilizar drones em complemento a caminhões no processo de entrega de mercadorias também traz uma redução nas emissões de gases causadores do efeito estufa, dado que drones utilizam energia elétrica como força motriz. O trabalho de Solaymani [2019] mostra que o setor de transporte era responsável por 23.96% das emissões de CO_2 no mundo em 2015, demonstrando ainda mais a necessidade de buscar formas alternativas de se realizar a logística das entregas de mercadorias.

Na literatura, além de Di Puglia Pugliese e Guerriero [2017], que propõe um modelo de programação linear inteira, existe o artigo de Di Puglia Pugliese et al. [2021], que propõe metaheurísticas de duas fases embutidas em uma estrutura *multi-start* seguidas de buscas locais para resolver o VRP-DTW, sendo este o objeto de comparação deste artigo.

Assim, o objetivo deste trabalho é resolver o VRP-DTW, realizando com o menor custo possível o atendimento dos clientes e observando as janelas de atendimento dos mesmos. Na solução, as capacidades e a autonomia dos veículos precisam ser respeitadas e, de cada rota de caminhão, podem partir um ou vários drones.

O restante do trabalho está assim organizado: a Seção 2 apresenta a descrição formal do problema; Os métodos computacionais são apresentados na Seção 3, enquanto os resultados são apresentados na Seção 4. Por último, as conclusões do trabalho são apresentados na Seção 5.

2. Descrição do Problema

O problema do roteamento de veículos com drones e janelas de tempo é uma variação do problema clássico do roteamento de veículos com janela de tempo. Nesta variação, a frota de veículos é formada por drones e caminhões, que devem operar de forma sincronizada. O objetivo é realizar, com o menor custo, uma visita a cada cliente para entrega de sua demanda, respeitando as janelas de tempo dos mesmos. Uma solução indica as rotas dos caminhões, sendo que cada caminhão pode possuir vários drones associados a ele.

Sejam os grafos completos $G^c(V, A^c)$ e $G^d(V, A^d)$, com V = $\{v_0, v_1, ..., v_n\}$, onde v_0 representa o depósito, v_i, $\forall\, i \geq 1$, representa cada cliente a ser atendido, A^c e A^d são, respectivamente, os conjuntos de arestas ponderadas pela distância a ser percorrida por caminhão e por

drone. Dessa forma, denota-se por $A^c{}_{ij}$ a distância percorrida por um caminhão que sai do cliente v_i diretamente para o cliente v_j, e, similarmente, denota-se por $A^d{}_{ij}$ a distância percorrida por um drone que vai de v_i a v_j.

A entrega para cada cliente v_i deve ser realizada dentro de uma janela de tempo pré-definida $[a_i, b_i]$. É utilizada a definição clássica de janelas de tempo, ou seja, o tempo t_i em que a entrega é realizada ao cliente v_i deve ser obrigatoriamente realizada entre a_i e b_i, ou seja, $a_i \leq t_i \leq b_i$. Caso o veículo responsável pela entrega chegue ao cliente v_i antes do instante de tempo a_i, ele deve aguardar para realizar a entrega. Caminhões podem realizar esta operação de espera sem nenhum problema, enquanto drones podem esperar por no máximo T instantes de tempo.

Seja $K = \{k_0, k_1, ..., k_m\}$ um conjunto contendo m caminhões idênticos, todos equipados com drones. Cada um desses caminhões possui uma capacidade de carga W_c e se desloca com velocidade constante s_c. Dessa forma, o tempo de viagem de um caminhão entre o nó v_i e v_j é dado por $A^c{}_{ij}/s_c$. Além disso, cada unidade de distância percorrida por um caminhão tem um custo de c_t.

Cada drone decola e pousa no seu próprio caminhão e pode ser utilizado na entrega de encomendas ao longo da rota do caminhão. O drone deve iniciar e finalizar uma rota apenas enquanto o caminhão está parado em um cliente, realizando um atendimento, ou no depósito, enquanto o caminhão ainda não iniciou ou já finalizou sua rota.

O processo de entrega de drone pode ser descrito como decolar, atender um cliente e retornar. Dessa forma, drones possuem capacidade unitária, sendo obrigados a voltar ao caminhão para recarregar a bateria e buscar a nova encomenda a cada entrega. Assim, uma rota de drone pode ser caracterizada pela tupla (x, y, z), com $x, y, z \in V$, sendo x o nó da decolagem do caminhão, y o cliente atendido pelo drone, e z o nó de retorno ao caminhão.

Cada drone tem uma capacidade de carga W_d, podendo fazer entregas apenas de mercadorias mais leves que este valor, voa a uma velocidade constante s_d e, dessa forma, o tempo de viagem utilizando drones entre o nó v_i e v_j é dado por $A^d{}_{ij}/s_d$. Como drones possuem bateria limitada, é imposta uma distância máxima E que cada drone pode percorrer para realizar cada entrega, assumindo que o gasto de energia é proporcional à distância percorrida. Assim, a entrega de drone (x, y, z) só é viável se $A^d{}_{xy} + A^d{}_{yz} \leq E$. Essa generalização só pode ser feita porque é considerado que a bateria do drone está totalmente carregada antes de iniciar cada entrega. Esta afirmação é plausível, dado que basta trocar a bateria do drone por outra com carga completa ao fim de cada entrega. Pelo mesmo motivo, o tempo de recarga dos drones é ignorado, não sendo necessário aguardar a recarga das baterias. Também têm-se um custo c_d para cada unidade de distância percorrida por um drone.

Para cada cliente v_i, há uma demanda associada, de peso q_i. Para que o cliente v_i possa ser atendido por um drone, deve ter um peso $q_i \leq W_d$. Vale ressaltar que as entregas de todos os clientes podem ser feitas por caminhões, mas apenas os que atendem a esta restrição são passíveis de entrega de drone. Entretanto, o somatório dos pesos de todas as entregas efetuadas por um caminhão e por todos os drones associados a esse caminhão deve ser menor que W_c, capacidade do caminhão.

O modelo matemático para o VRP-DTW foi proposto por Di Puglia Pugliese et al. [2021], onde é possível consultar o modelo completo. A função objetivo é apresentada na Equação 1. As variáveis binárias x^k_{ij} assumem valor 1, se a aresta (v_i, v_j) está sendo usada na rota do caminhão k, ou valor 0, caso contrário. As variáveis y^k_{abc}, também binária, apresentam o valor 1, se as arestas (v_a, v_b) e (v_b, v_c) são utilizadas na rota de algum drone associado ao k-ésimo caminhão, e o valor 0, caso contrário.

$$min\ Z = c_t \cdot (\sum_{i \in V} \sum_{j \in V} \sum_{k \in K} {A^c}_{ij} \cdot x^k_{ij}) + c_d \cdot (\sum_{a \in V} \sum_{b \in V} \sum_{c \in V} \sum_{k \in K} ({A^d}_{ab} + {A^d}_{bc}) \cdot y^k_{abc}) \quad (1)$$

Vale ressaltar a flexibilidade desta função objetivo. Como basicamente se tem uma constante multiplicando a distância percorrida por drone e outra constante multiplicando a distância percorrida por caminhão, pode-se, variando c_t e c_d, utilizá-la tanto para prevermos questões monetárias, dado que o custo tende a ser maior quanto maior é a distância percorrida por drones e caminhões, quanto questões ambientais, dado que as emissões de CO_2 são diretamente proporcionais à distância percorrida. Além disso, ela deixa explicita a diferença de "custo" entre utilizar drones e caminhões para realizar uma dada entrega.

Assim, para resolver VRP-DTW deseja-se definir um conjunto de rotas e entregas de caminhões e drones de forma que todos os clientes $v_i \in V$ sejam atendidos exatamente uma vez dentro de sua janela de tempo com o custo mínimo de transporte.

3. Métodos Computacionais

Nesta seção, são descritas as abordagens propostas para resolver o VRP-DTW: um GRASP (do inglês *Greedy Randomized Adaptive Search Procedure*) reativo e um algoritmo baseado em um modelo de Programação Linear Inteira — PLI, que combina rotas de diferentes soluções. O GRASP é uma metaheurística iterativa que combina etapas de construção e de busca local. Nas próximas seções, são descritos o algoritmo construtivo proposto e um algoritmo RVND (*Random Variable Neighborhood Descendent*), utilizado na etapa de busca local.

3.1. Algoritmo construtivo

Com objetivo de construir diferentes soluções para posterior refinamento por um algoritmo de busca local, neste trabalho, é proposto um algoritmo construtivo randomizado. A partir de uma solução inicialmente vazia, a cada iteração um cliente é inserido na solução. O Algoritmo 1 descreve este procedimento. Como parâmetros de entrada, têm-se a instância do problema e um parâmetro α, que indica o grau de aleatoriedade do construtivo.

Como pode ser visto na linha 2 e 3 do Algoritmo 1, a solução S começa vazia e todos os clientes a serem visitados são incluídos no conjunto V. A cada iteração (linhas de 4 a 8), um cliente v é selecionado para ser inserido em S segundo um dado tipo de inserção chamado de *movimento*. Ou seja, o *movimento* indica qual tipo de veículo atenderá o cliente e em que posição o cliente será inserido em uma rota. Após a inserção, o conjunto de clientes V é atualizado. O algoritmo termina quando o conjunto V estiver vazio.

Três diferentes estratégias de seleção (linha 5) foram adotadas neste trabalho:

- **C1:** Nesta estratégia, para seleção dos clientes foi utilizado como critério o maior peso para entrega. Dessa forma, o algoritmo tende a inserir primeiro na solução os clientes a serem atendidos por caminhões, favorecendo assim a formação de rotas básicas de caminhões antes das primeiras tentativas de entrega via drone. O cliente a ser inserido é selecionado entre os $\alpha\%$ clientes não atendidos que possuem entrega de maior peso.

  Neste algoritmo construtivo, o cliente selecionado v pode ser inserido em qualquer ponto da rota, tanto em rotas de drone quanto em rotas de caminhão. Todas estas alternativas são avaliadas e verifica-se também a possibilidade de criação de uma nova rota para este cliente. Avaliadas todas as possibilidades, seleciona-se uma entre as $\alpha\%$ inserções mais baratas. Assim, é dada prioridade a inserções com menor impacto no custo atual da solução.

Algoritmo 1: Algoritmo Construtivo

Entrada: $\alpha, instancia$
Saída: $Solucao\ S$

```
início
    S = ∅;
    V = {v_1, v_2, ..., v_n};
    while V ≠ ∅ do
        v, movimento = selecionaInsercao(S, V, α);
        S = inserirClienteNaSolucao(S, v, movimento);
        V = V − {v};
    end while
    return S
fim
```

- **C2:** Esta estratégia é bastante similar a utilizada do Construtivo C1, diferindo no critério de seleção dos clientes. Este construtivo considera o tempo de fechamento da janela de tempo. Dessa forma, os vértices que tendem a ser inseridos primeiro na solução são aqueles que têm o fechamento da janela de tempo mais cedo.

- **C3:** Esta estratégia utiliza como critério de seleção de clientes o menor tempo de fechamento da janela de tempo. Diferentemente dos outros construtivos, neste caso, a seleção do cliente é determinística, sem que haja seleção de um entre os $\alpha\%$ melhores candidatos.

  Outra diferença fundamental nesta estratégia é que o cliente selecionado v é, necessariamente, inserido no final de alguma rota pré-existente ou em uma rota nova via caminhão. A partir da seleção do cliente a ser atendido via caminhão, é feita uma busca nos clientes restantes para verificar quais deles podem ser atendidos via drone durante a movimentação do caminhão entre o antigo último cliente atendido na rota e o vértice v. Assim, a randomização deste guloso fica por conta de escolher uma das $\alpha\%$ melhores opções dentre as opções levantadas, utilizando como critério o custo médio de movimentação por cliente atendido. Dessa forma, pode-se realizar apenas uma movimentação de caminhão, uma entrega via caminhão e uma via drone, uma entrega via caminhão e duas via drone, e assim sucessivamente.

3.2. Random Variable Neighborhood Descendent

Como estratégia para realizar melhorias nas soluções obtidas pelos construtivos, optou-se por utilizar o *Random Variable Neighborhood Descendent* (RVND), uma variação do VND (do inglês *Variable Neighborhood Descendent*) proposta por Mladenović e Hansen [1997], em que se considera um conjunto de estruturas de vizinhança alternadas durante o processo de busca.

No RVND proposto, foram utilizados 10 movimentos de busca local, sendo os oito primeiros selecionados randomicamente, enquanto os dois últimos foram fixados. Na Figura 1, estão representados os oito movimentos selecionados de forma randômica no RVND. As setas com linhas contínuas representam rotas de caminhões, enquanto as setas com linhas tracejadas representam rotas de drones. O vértice em destaque em cada uma das figuras indica o nó em que o movimento foi realizado. Vale ressaltar que o depósito é representado pelo nó 0. A Figura 1(a) representa a solução base antes de qualquer movimento ser realizado.

As Figuras 1(b), 1(c), 1(d), 1(e) exemplificam os movimentos intra-rota utilizados. O movimento *or-opt1-truck-drone* tenta remover um nó atendido por caminhão e passar a atendê-lo

por meio de um drone. Já o movimento *or-opt1-truck-truck* modifica a posição na rota onde o cliente está sendo atendido por caminhão. O movimento *or-opt1-drone-truck* tenta remover um cliente atendido por drone e passar a atendê-lo via caminhão. Por fim, o movimento *or-opt1-drone-drone* tenta trocar a posição de partida e/ou de chegada do drone no caminhão ao atender o cliente.

Já as Figuras 1(f), 1(g), 1(h), 1(i) exemplificam os movimentos inter-rota utilizados. Dessa forma, o movimento *shift-truck-truck* tenta remover um cliente de caminhão de uma rota e inseri-lo como entrega de caminhão em outra rota. O movimento *shift-truck-drone* remove um cliente de caminhão de uma rota e o atende via drone em outra rota. Já o movimento *shift-drone-truck* remove um cliente atendido via drone em uma rota e o insere como entrega de caminhão em outra rota. Por fim, o movimento *shift-drone-drone* remove um cliente via drone de uma rota e tenta realizar a entrega em outra rota também por um drone.

Os dois movimentos restantes fixados no final do RVND trabalham com a remoção e reinserção de k nós aleatórios da solução atual. No clientReinsert(2), utiliza-se $k = 2$ e, caso em sua execução não seja encontrada nenhuma solução melhor que a inicial, tenta-se utilizar o movimento clientReinsert(3), em que $k = 3$. Estes dois movimentos finais do RVND não são exaustivos, ou seja, não são testadas todas as possibilidades de movimento, sendo o número de tentativas definido por um parâmetro do algoritmo.

A estratégia de seleção do vizinho também é um parâmetro do algoritmo. O RVND pode adotar a estratégia primeiro aprimorante ou melhor aprimorante.

3.3. GRASP Reativo

No GRASP Reativo, proposto por Prais e Ribeiro [2000], o grau de aleatoriedade α do algoritmo construtivo é ajustado durante a execução do GRASP, utilizando-se como referência a qualidade das soluções alcançadas.

O GRASP Reativo proposto para o VRP-DTW é descrito no Algoritmo 2. O algoritmo recebe como parâmetro uma lista com possíveis valores de aleatoriedade α e associa uma mesma probabilidade de ocorrência para cada elemento da lista.

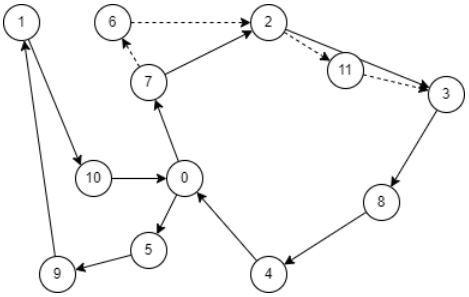
Em resumo, a cada iteração do algoritmo: é selecionado um valor de α da lista, levando-se em conta suas respectivas probabilidades (linha 5); uma solução é obtida através da combinação de uma execução do algoritmo construtivo da Seção 3.1 e do RVND apresentado na Seção 3.2 (linhas 6 e 7); um histórico de qualidade das soluções obtidas por cada valor de α é atualizado (linha 8); e, se for o caso, a melhor solução alcançada é armazenada (linha 10). A cada bloco de iterações, as probabilidades de seleção de cada opção de α são atualizadas, levando-se em conta o histórico de qualidade de cada execução. Desta forma, os valores de α que historicamente alcançam os melhores resultados são priorizados em relação aos outros (linha 13).

O algoritmo é executado até que o número de iterações recebido por parâmetro seja atingido. Por fim, a solução de menor custo obtida durante todo o processo é retornada.

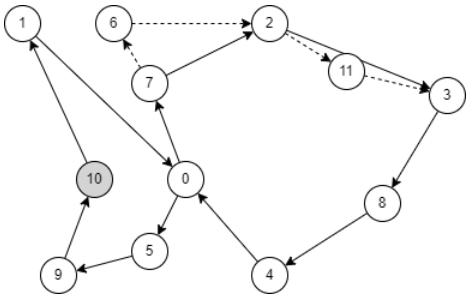
3.4. Estratégia de combinação de rotas através de Programação Linear Inteira

Como etapa final do processo de otimização, foi proposto um modelo de Programação Linear Inteira (PLI). Dada a grande variedade de soluções geradas durante a execução do GRASP, o uso da PLI permite explorar a recombinação de rotas de diferentes soluções, na tentativa de formar uma nova solução de menor custo.

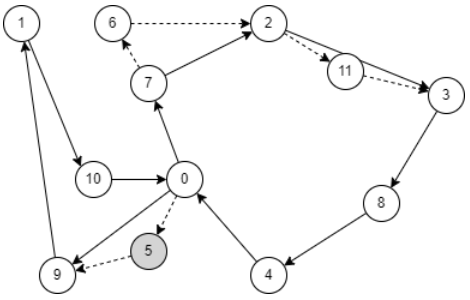
De cada solução previamente gerada, uma variável é criada para representar cada uma de suas rotas e o modelo vai selecionar um subconjunto de rotas compatíveis. Assim, a variável r_i indica se a i-ésima rota foi selecionada ($r_i = 1$) ou não ($r_i = 0$). O custo da rota i é indicado por $custo_i$ e $atend_{i,v}$ é uma constante binária que representa se a rota i atende o cliente v.



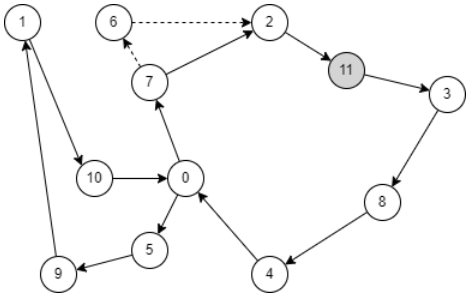
(a) Rota Original



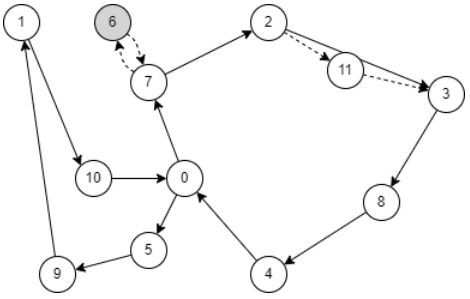
(b) Movimento Or-opt1-truck-truck



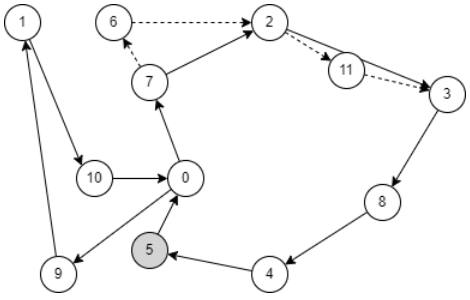
(c) Movimento Or-opt1-truck-drone



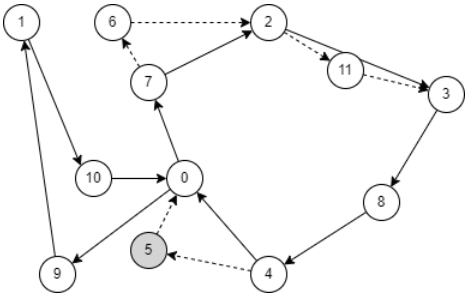
(d) Movimento Or-opt1-Drone-truck



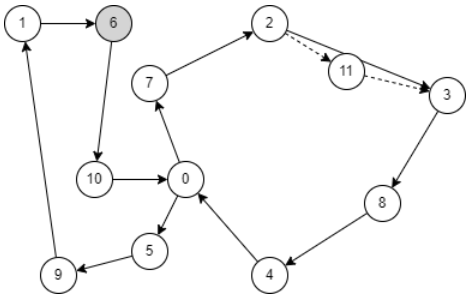
(e) Movimento Or-opt1-drone-drone



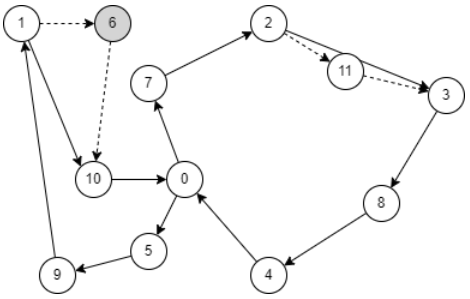
(f) Movimento Shift-truck-truck



(g) Movimento Shift-truck-drone



(h) Movimento Shift-drone-truck



(i) Movimento Shift-drone-drone

Figura 1: Movimentos utilizados no RVND

Algoritmo 2: GRASP Reativo

Entrada: $listaα, numeroIteracoes, tamanhoBloco, instancia$
Saída: $Solucao\ S^*$

```
início
    probabilidades = iniciaProbabilidades(listaα);
    S* = ∅;
    for (i = 0; i < numeroIteracoes; i++) do
        α = escolheα(probabilidades, listaα);
        S = AlgoritmoConstrutivo(α);
        S = RVND(S);
        atualizaHistorico(S, α);
        se custo(S) < custo(S*) então
            S* = S;
        fim se
        se i mod tamanhoBloco == 0 então
            probabilidades = atualizaProbabilidades(probabilidades, listaα);
        fim se
    end for
    return S*
fim
```

As restrições do modelo garantem que, para as rotas selecionadas, cada cliente será visitado exatamente uma vez (Equação 3). Vale ressaltar que as rotas são viáveis e, por isso, as demais restrições do problema já foram consideradas durante sua criação: as demandas dos clientes de uma mesma rota não ultrapassam a capacidade do caminhão e seus drones, a autonomia dos drones é respeitada e os clientes atendidos na rota são visitados no intervalo indicado de sua janela de atendimento. O objetivo do modelo é minimizar o somatório dos custos das rotas escolhidas (Equação 2).

$$\text{Minimizar } z = \sum_{i \in R} custo_i \times r_i \tag{2}$$

$$\sum_{i \in R} atend_{i,v} \times r_i = 1 \quad \forall v \in V \tag{3}$$

$$r_i \in \{0,1\}, \quad \forall i \in R \tag{4}$$

Como conjunto R de rotas consideradas pelo PLI, foram utilizadas as rotas de todas as soluções geradas pelo GRASP descrito na Seção 3.3, utilizando-se dos construtivos, **C1**, **C2** ou **C3**, e das estratégias primeiro ou melhor aprimorante no RVND descrito na Seção 3.2. Também foram inseridas no conjunto R rotas unitárias, ou seja, percursos contendo apenas um cliente, construídas separadamente. Este método, que combina as soluções obtidas via GRASP através de um modelo de programação linear inteira, é nomeada GRASP-PLI.

4. Experimentos Computacionais

Nessa seção são apresentadas as instâncias utilizadas, os parâmetros adotados e os resultados obtidos nos testes realizados.

4.1. Instâncias Utilizadas

Os experimentos foram realizados sobre um conjunto de instâncias geradas a partir das instâncias de Solomon [1987], divididas em 3 grupos: randômicas (R2), clusterizadas (C2) e randômicas clusterizadas (RC2). O processo de adaptação das instâncias é semelhante ao utilizado por Di

Puglia Pugliese et al. [2021]. São mantidas as posições originais de todos os clientes e do depósito. Também são mantidas as demandas e as janelas de tempo.

É assumido que 80% dos clientes podem ser atendidos via drone, sendo os 80% com menor demanda. O tempo de serviço para atendimento via caminhão é igual ao tempo de serviço da instância original, enquanto o tempo de serviço para atendimento via drone é definido como metade do tempo original. O tempo máximo que um drone pode ficar parado esperando a janela de tempo de um cliente abrir é determinado conforme o tipo da instância, sendo 10 unidades de tempo para instâncias randômicas e randômicas clusterizadas e 90 unidades de tempo para as clusterizadas. O custo de transporte por unidade de distância via caminhão é 25, enquanto o custo por unidade de distância via drone é 1. Além disso, a capacidade de carga de cada caminhão é a mesma da instância original e a métrica para medir distâncias entre clientes utilizada para caminhões é a métrica de Manhattan, simulando quarteirões, enquanto a métrica utilizada para drones é a euclidiana, simulando entregas em linha reta. As instâncias obtidas estão disponíveis no repositório[1].

4.2. Ajuste de Parâmetros

Os ajustes de parâmetros para todos os algoritmos apresentados foram feitos a partir de testes empíricos e do framework *iterated race* (IRACE) de López-Ibáñez et al. [2016].

Para o RVND foi calibrado o número de vizinhos explorados pelas vizinhanças clientReinsert(2) e do clientReinsert(3), onde definiu-se empiricamente utilizar $n/2,5$ tentativas para o clientReinsert(2) e $n/1,25$ para o clientReinsert(3), onde n indica o número de vértices da instância.

Para o GRASP, optou-se atualizar as probabilidades de cada α ser selecionado a cada 50 iterações.

Para a combinação GRASP-PLI, seis diferentes versões do GRASP foram combinadas para fornecerem soluções distintas a serem combinadas pelo modelo, como listadas na Tabela 1. Variou-se o construtivo utilizado, o tipo de busca no RVND e o conjunto de valores para α.

Para definir o número de iterações com cada algoritmo foi utilizado o IRACE. Cada um dos algoritmos tinha a possibilidade de ser executado entre 0 a 900 iterações. Além disso, o somatório de todas as iterações foi limitado a 1200. A coluna n_iterações da Tabela 1 indica o número de iterações definidos pelo IRACE para cada uma das versões do algoritmo.

Construtivo	RVND	α's	n_iterações
C1	Primeiro Aprimorante	{0.1, 0.2, 0.3}	500
C1	Melhor Aprimorante	{0.1, 0.2, 0.3}	450
C2	Primeiro Aprimorante	{0.1, 0.2, 0.3}	50
C2	Melhor Aprimorante	{0.1, 0.2, 0.3}	80
C3	Primeiro Aprimorante	{0.2}	100
C3	Melhor Aprimorante	{0.2}	0

Tabela 1: Parâmetros ajustados para a combinação de rotas através de PLI

4.3. Resultados

Todos os algoritmos descritos neste trabalho foram desenvolvidos na linguagem *C++*, com os modelos matemáticos sendo resolvidos através do resolvedor Gurobi Optimization, LLC [2021], na versão 9.1.2. Os testes foram executados em uma máquina com processador core i5 7200U com um clock base de 2.50 GHz, 8 GB de memória RAM com 2400 MHz, sistema operacional Mint 20.2 com kernel Linux 5.4.0-96-generic. O compilador utilizado foi o g++ (Ubuntu

[1] https://github.com/eugeniobrf/VRP-DTW

9.3.0-17ubuntu1 20.04) 9.3.0. Cada instância foi executada 30 vezes, sendo aferidos o tempo de execução e o custo da solução encontrada.

Dado que o código desenvolvido em Di Puglia Pugliese et al. [2021] não foi disponibilizado e considerando que o processo de geração das instâncias não é determinístico, foi necessário implementar os algoritmos da literatura para comparação. A implementação foi a mais fiel possível e utilizou os mesmos elementos, ambiente de desenvolvimento e testes de cada estratégia proposta.

Nas tabelas com resumos de resultados, as linhas denotadas por **Todas** apresentam a média dos resultados para todas as instâncias, as linhas denotadas por **25**, **50** e **100** representam, respectivamente, a média dos resultados das instâncias conforme o número de clientes (25, 50 ou 100), enquanto as linhas denotadas por **C**, **R** e **RC** apresentam as médias dos resultados das instâncias conforme o tipo (clusterizadas, randomizadas ou randomizadas e clusterizadas).

As colunas nomeadas como **MSTPHFLS*** apresentam os resultados obtidos pela implementação do algoritmo MSTPHFLS, de Di Puglia Pugliese et al. [2021], desenvolvido neste trabalho. Já as colunas denotadas por **MSPTPH2*** mostram os resultados obtidos a partir da implementação do algoritmo MSPTPH2, dos mesmos autores. As colunas **GRASP-PLI** mostram os resultados obtidos pelas estratégias propostas neste trabalho.

A Tabela 2 apresenta os resultados no que se refere à qualidade das soluções obtidas, melhor resultado e média, agrupados por número de clientes e por tipo de instância. As colunas indicam a média da diferença percentual para cada grupo de instância em relação ao melhor valor obtido dentre todas as implementações.

Tabela 2: Resumo dos resultados quanto ao custo médio e melhor custo por algoritmo

	Custo Médio				**Melhor Custo**			
	Melhor	**MSPTH2***	**MSTPHFLS***	**GRASP-PLI**	**Melhor**	**MSPTH2***	**MSTPHFLS***	**GRASP-PLI**
R	10709,4	5,55%	21,87%	**1,69%**	9907,2	**2,20%**	13,45%	3,67%
C	9414,2	3,90%	32,19%	**2,58%**	8749,8	2,19%	22,53%	**2,17%**
RC	13932,8	9,86%	40,92%	**0,15%**	13056,5	2,74%	25,26%	**1,16%**
25	6335,8	8,89%	33,29%	**0,14%**	6047,3	3,00%	17,61%	**0,84%**
50	10694,5	6,14%	31,09%	**0,32%**	10066,2	2,14%	19,65%	**1,61%**
100	16811,9	**3,98%**	27,34%	4,05%	15378,7	**1,92%**	21,65%	5,00%
Todas	11280,7	6,34%	30,57%	**1,50%**	10497,4	**2,36%**	19,64%	2,48%

A partir da Tabela 2 é possível observar que, dentre as abordagens da literatura, MSPTH2* possui os melhores resultados em relação ao custo médio e ao melhor custo, sendo significativamente melhor que MSTPHFLS*. Por este motivo, as principais análises comparativas feitas serão entre o GRASP-PLI e o MSPTH2*.

Considerando todas as estratégias e em termos de resultados médios, o GRASP-PLI consegue os melhores resultados gerais independente do tipo de instância. Quanto ao número de clientes, apenas para o grupo de 100 clientes não obteve a melhor média. Ao se considerar o custo médio para todas as instâncias agrupadas, a abordagem proposta obteve resultado com desvio percentual médio mais de quatro vezes menor que o apresentado pela estratégia MSPTH2*, melhor abordagem da literatura. Já com relação ao melhor custo, o MSPTH2* consegue os melhores resultados para as instâncias do tipo R e instâncias de 100 clientes, enquanto a abordagem GRASP-PLI a supera nos demais tipos e tamanhos de instância. Ao se analisar os resultados para o conjunto de todas as instâncias sobre o melhor custo, a abordagem da literatura supera a bordagem proposta em apenas 0.12 ponto percentual.

Em relação tempo médio de execução, a Tabela 3 apresenta o resultado dos algoritmos em segundos conforme o número de clientes e o tipo de instância. Os tempos apresentados para as implementações das heurísticas da literatura são significativamente menores do que aqueles apresentados pelos autores no artigo de referência. Acredita-se que isso seja motivado pela linguagem de programação ou ambiente de teste.

Tabela 3: Resumo do tempo médio por algoritmo

	MSPTH2*	**MSTPHFLS***	**GRASP-PLI**
R	0,849	0,002	17,230
C	2,053	0,002	16,577
RC	1,063	0,002	34,447
25	0,030	0,001	1,192
50	0,301	0,001	5,786
100	3,477	0,004	59,436
Todas	1,269	0,002	22,138

Pode-se perceber que o algoritmo GRASP-PLI tem um grande impacto em seu tempo quando submetido às instâncias do tipo RC ou de tamanho 100, o que acaba elevando bastante sua média de tempo, que foi de quase 18 vezes maior na média que o apresentado pelo MSPTH2*. Convém destacar que a abordagem GRASP-PLI, tendo apresentado resultados competitivos em termos de qualidade, mesmo com tempos superiores ao da abordagem da literatura, tem aplicação prática considerando as características do problema, já que a média geral de tempo de execução não ultrapassa 30 segundos.

5. Conclusão e Trabalhos Futuros

Este trabalho propôs um algoritmo baseado em um modelo de programação linear inteira que combina rotas provenientes de diferentes versões da heurística GRASP para resolver o problema de roteamento de veículos com drones e janelas de tempo (GRASP-PLI). Além disso, foi feita a implementação dos algoritmos propostos por Di Puglia Pugliese et al. [2021], utilizados como base de comparação com o algoritmo proposto. O algoritmo baseado em modelo de programação linear inteira trabalha com um conjunto de soluções de entrada, desmembrando-as e formando novas soluções. Em suma, o algoritmo combina rotas de diferentes soluções, garantindo que todos os clientes sejam atendidos exatamente uma vez.

Como conjunto de soluções de entrada, utilizou-se rotas unitárias formadas artificialmente e rotas provenientes de um algoritmo GRASP, que pode utilizar três heurísticas gulosas diferentes e, na fase de busca local foi utilizado um algoritmo RVND modificado, onde foram utilizados os movimentos Or-opt1-truck-truck, Or-opt1-truck-drone, Or-opt1-Drone-truck, Or-opt1-drone-drone, Shift-truck-truck, Shift-truck-drone, Shift-drone-truck e Shift-drone-drone, além dos movimentos clientReinsert(2) e clientReinsert(3) caso nenhum os anteriores melhore a solução.

Além disso, foram feitas implementações de dois algoritmos da literatura, o MSTPH2 e o MSTPHFLS, de forma a tornar possível comparações entre com os algoritmos propostos neste trabalho. Todos os algoritmos foram submetidos a um conjunto de 81 instâncias da literatura. O GRASP-PL proposto encontrou resultados até quatro vezes melhor que as abordagens da literatura quanto ao custo médio e ficando em média a 0.12% quanto às melhores soluções.

Como trabalhos futuros, sugere-se uma análise das contribuições de cada movimento do RVND no sentido de otimizá-lo e reduzir os tempos de processamento, além de um estudo mais

aprofundado sobre as instâncias onde se obteve os piores resultados, de forma a aplicar movimentos de busca local que melhoram os mesmos. Além disso, sugere-se utilizar fontes mais variadas de soluções para o MIP, dando para o mesmo um conjunto de soluções mais variado para exploração.

Referências

Bertão, N. (2021). Velocidade de entrega de produtos está no topo da lista de expectativa do consumidor. *Valor Investe*. URL https://valorinveste.globo.com/objetivo/gastar-bem/noticia/2021/09/28/velocidade-de-entrega-de-produtos-esta-no-topo-da-lista-de-expectativa-do-consumidor.ghtml.

Di Puglia Pugliese, L. e Guerriero, F. (2017). Last-mile deliveries by using drones and classical vehicles. In Sforza, A. e Sterle, C., editors, *Optimization and Decision Science: Methodologies and Applications*, p. 557–565, Cham. Springer International Publishing.

Di Puglia Pugliese, L., Macrina, G., e Guerriero, F. (2021). Trucks and drones cooperation in the last-mile delivery process. *Networks*, 78(4):371–399. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.22015.

Gurobi Optimization, LLC (2021). Gurobi Optimizer Reference Manual. URL https://www.gurobi.com.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58. ISSN 2214-7160. URL https://www.sciencedirect.com/science/article/pii/S2214716015300270.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100. ISSN 0305-0548. URL https://www.sciencedirect.com/science/article/pii/S0305054897000312.

Prais, M. e Ribeiro, C. (2000). Parameter variation in grasp procedures. *Investigación Operativa*, 9:1–20.

Ranieri, L., Digiesi, S., Silvestri, B., e Roccotelli, M. (2018). A review of last mile logistics innovations in an externalities cost reduction vision. *Sustainability*, 10(3). ISSN 2071-1050. URL https://www.mdpi.com/2071-1050/10/3/782.

Solaymani, S. (2019). Co2 emissions patterns in 7 top carbon emitter economies: The case of transport sector. *Energy*, 168:989–1001. ISSN 0360-5442. URL https://www.sciencedirect.com/science/article/pii/S0360544218323569.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265. URL https://doi.org/10.1287/opre.35.2.254.

Wang, Z. e Sheu, J.-B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 122:350–364. ISSN 0191-2615. URL https://www.sciencedirect.com/science/article/pii/S0191261518307884.