LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

PLANEJAMENTO DE CAMINHO DE COBERTURA PARA DRONES MINIMIZANDO O CONSUMO DE ENERGIA

Bárbara Cristina F. de Souza, José Elias C. Arroyo
Departamento de Informática - Universidade Federal de Viçosa
Campus Universitário, 36570-900, Viçosa, MG – Brasil
barbaracsouza@ufv.br, jarroyo@ufv.br

RESUMO

Neste estudo, aborda-se o problema do Planejamento de Caminho de Cobertura (PCC), que visa encontrar o melhor caminho para um drone de forma que uma determinada área seja completamente coberta. Consideram-se áreas arbitrárias que contêm obstáculos (por exemplo, edifícios, torres, árvores ou morros). A energia da bateria é o principal recurso que limita o tempo de voo de um drone. Para minimizar o consumo de energia do drone consideram-se a minimização da distância total percorrida, as variações de altitude e as mudanças de direção (curvas) feitas durante seu percurso. O PCC é modelado como um Problema do Caixeiro Viajante, onde o drone deve visitar todos os pontos que não sejam obstáculos. Como o problema PCC é NP-difícil, neste trabalho propõe-se uma heurística baseada na meta-heurística GRASP. As soluções determinadas pela heurística proposta são comparadas com soluções geradas pelo modelo de Programação Linear Inteiro Misto (resolvido pelo solver GUROBI). Os resultados mostram que o algoritmo GRASP determina soluções de excelente qualidade gastando tempos de CPU significativamente menores que o GUROBI.

PALAVRAS CHAVE. Planejamento de caminho de cobertura, Meta-heurísticas, Otimização.

Tópicos (Meta-heurísticas, Otimização)

ABSTRACT

In this study, the Coverage Path Planning (CPP) problem is addressed, which aims to find the best path for a drone so that a given area is completely covered. Arbitrary areas containing obstacles (e.g. buildings, towers, trees or hills) are considered. Battery power is the main resource that limits a drone's flight time. To minimize the drone's energy consumption, we consider minimizing the total distance traveled, altitude variations and direction changes (turns) made during its route. The CPP is modeled as a Traveling Salesman Problem, where the drone must visit all points that are not obstacles. As the PCC problem is NP-hard, in this work we propose a heuristic based on the GRASP meta-heuristic. The solutions determined by the proposed heuristic are compared with solutions generated by the Mixed-Integer Linear Programming model (solved by the optimization solver GUROBI). The results show that the GRASP algorithm determines excellent quality solutions using significantly less CPU time than GUROBI.

KEYWORDS. Coverage Path Planning, Drones, Metaheuristics, Optimization.

Paper topics (Metaheuristics, Optimization)

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br

1. Introdução

Veículos aéreos não tripulados (UAVs), também conhecidos como drones, começaram a ser usados nos últimos anos no setor privado e nas indústrias comerciais. Muitos segmentos de aplicação de drones abordam a cobertura de terrenos, como vigilância dentro das cidades, agricultura inteligente e de precisão, fotogrametria, detecção de desastres, rastreamento de incêndios florestais e outros.

Imagens aéreas são ferramentas úteis para trabalhadores que praticam agricultura de precisão. A dificuldade em obter imagens aéreas de alta resolução em uma janela de tempo estreita, considerando as restrições climáticas e o alto custo dos serviços comerciais, são as principais desvantagens dessas técnicas. Neste contexto, drones desempenham um papel importante na obtenção de imagens aéreas. O sistema proposto por [Valente et al., 2013] permite aos agricultores definir e executar facilmente uma missão de cobertura de imagens aéreas, utilizando ferramentas de sistemas de informação geográfica para obter mosaicos de imagens de alta resolução.

A expansão dos campos de cultivo, em decorrência da revolução industrial, tem gerado um desafio significativo no que diz respeito à eficiência na aplicação de fertilizantes e pesticidas em áreas extensas [Pimentel, 2018]. Neste contexto, os drones emergem como uma alternativa viável e economicamente vantajosa para fazer a cobertura de um campo de cultivo, que necessita ser pulverizado com produtos químicos tais como fertilizantes e pesticidas. Essa abordagem tem sido amplamente estudada e adotada devido à sua capacidade de proporcionar uma cobertura precisa e eficaz, maximizando a utilização dos insumos agrícolas e minimizando desperdícios. A aplicação de tecnologias de mapeamento e planejamento de caminhos permite aos operadores de drones calcular rotas otimizadas, garantindo uma distribuição homogênea dos produtos químicos ao longo da área de cultivo.

Recentemente, têm surgido um grande número de estudos que abordam o Planejamento de Caminho de Cobertura (PCC) para drones. Modares et al. [2017] propõem um algoritmo de cobertura para múltiplos drones utilizando busca em profundidade limitada com backtracking. Cada drone é responsável por cobrir parte da área total. Já Shivgan e Dong [2020] apresentam um algoritmo genético para minimizar o consumo de energia do drone, levando em consideração as curvas feitas durante seu trajeto. Cabreira et al. [2019] propõem um algoritmo de PCC baseado em grade (onde a área a ser coberta é dividida em células) visando minimizar o consumo de energia durante missões de mapeamento em áreas de formato irregular. Inspirado em métodos de clustering baseados em densidade, Chen et al. [2021] propõem um algoritmo baseado em clusterização para classificar regiões e obter caminhos altamente eficientes para drones. Baseado em um modelo de energia derivado de medidas reais, Di Franco e Buttazzo [2015] apresentam um algoritmo que leva em consideração energia disponível, peso, velocidade máxima e resolução do sensor do drone. Usando múltiplos drones, Xie e Chen [2022] utilizam um método baseado em *Branch-and-Bound*, bem como um algoritmo genético, capazes de resolver problemas em grande escala. Vasquez-Gomez et al. [2018] apresentam um método para resolver o PCC para áreas disjuntas. A solução se baseia em dividir o problema em duas etapas: otimização da ordem de visita e otimização da orientação das linhas de voo. O método é validado por meio de várias simulações usando parâmetros reais.

Geralmente, algoritmos para o problema de PCC, são baseados no Problema do Caixeiro Viajante, referido na literatura inglesa como *Traveling Salesman Problem* (TSP) [Osaba et al., 2020]. No entanto, a aplicação direta desses algoritmos revela-se inadequada, uma vez que se limitam a considerar apenas a distância entre os pontos a serem visitados, sem contemplar outras variáveis relevantes para o contexto do PCC. Visando isso, foi formulado um problema de Planejamento de Caminho de Cobertura com Eficiência Energética (PCCEE) que considera não somente

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br

a distância entre dois pontos, mas também a variação de altitude entre eles e as curvas feitas pelo drone ao longo da trajetória.

Como forma de abordar o PCCEE, propõe-se um algoritmo baseado na meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedures*) [Resende e Ribeiro, 2013] que utiliza o 2-OPT [Croes, 1958] na sua fase de busca local. O desempenho do algoritmo é testado usando instâncias de tamanhos variados. As soluções determinadas pelo algoritmo proposto são comparadas com soluções geradas pelo modelo de Programação Linear Inteira que é resolvido pelo solver GUROBI [Gurobi Optimization, 2021].

2. Modelagem do Problema

A modelagem do problema de Planejamento de Caminho de Cobertura com Eficiência Energética (PCCEE) segue uma abordagem semelhante à proposta apresentada por [Modares et al., 2017]. Contudo, é importante ressaltar as distinções entre este estudo e o trabalho de [Modares et al., 2017]. A principal diferença reside no fato de que, neste estudo consideram-se possíveis variações de altitude ao longo do terreno percorrido e a cobertura da área é realizada por um único drone.

A área ou terreno a ser coberta pelo drone é representada por uma grade de células do mesmo tamanho ($a \times b$). Cada célula possui um ponto central i que tem coordenadas (x_i, y_i, z_i), onde z_i representa a componente de altitude do terreno. O drone cobrirá uma célula inteiramente quando passa pelo seu ponto central. Assim, tem-se um conjunto de pontos $P = V \cup O$, onde V é o conjunto de pontos a serem visitados pelo drone, e O é um conjunto de obstáculos.

O terreno a ser coberto pelo drone é representado por um grafo não direcionado $G(V, E)$, onde E é o conjunto de arestas e_{ij}, $\forall i, j \in V$. Assume-se que o drone sempre atravessa células adjacentes. Devido à presença de obstáculos, é possível que o drone passe por uma mesma célula em mais de uma ocasião (por exemplo, quando existe uma área de terreno sem saída). Considera-se que a visita à célula ocorra apenas em uma dessas passagens. Por exemplo, a aplicação de fertilizante em uma área ou célula de cultivo ocorre uma única vez. Se o drone passar novamente por uma célula já visitada, não haverá aplicação do fertilizante. As células já visitadas podem ser utilizadas como pontos de passagens no caminho do drone. Nas Figuras (1a) e (1b), mostram-se dois exemplos de grafos de adjacências que representam áreas a serem cobertos pelo drone. Na Figura (1a), tem-se um terreno com 121 células, das quais 28 são obstáculos (células na cor vermelha). Neste caso, o grafo de adjacências possui $|V| = 93$ células. Na Figura (1b), tem-se um terreno com 144 células, das quais 36 são obstáculos.

No grafo G, para cada par de pontos adjacentes, i e j, determina-se a distância euclidiana usando a Equação (1):

$$d_{ij} = \begin{cases} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} & \text{, se } e_{ij} \in E \\ \infty & \text{, caso contrário.} \end{cases} \tag{1}$$

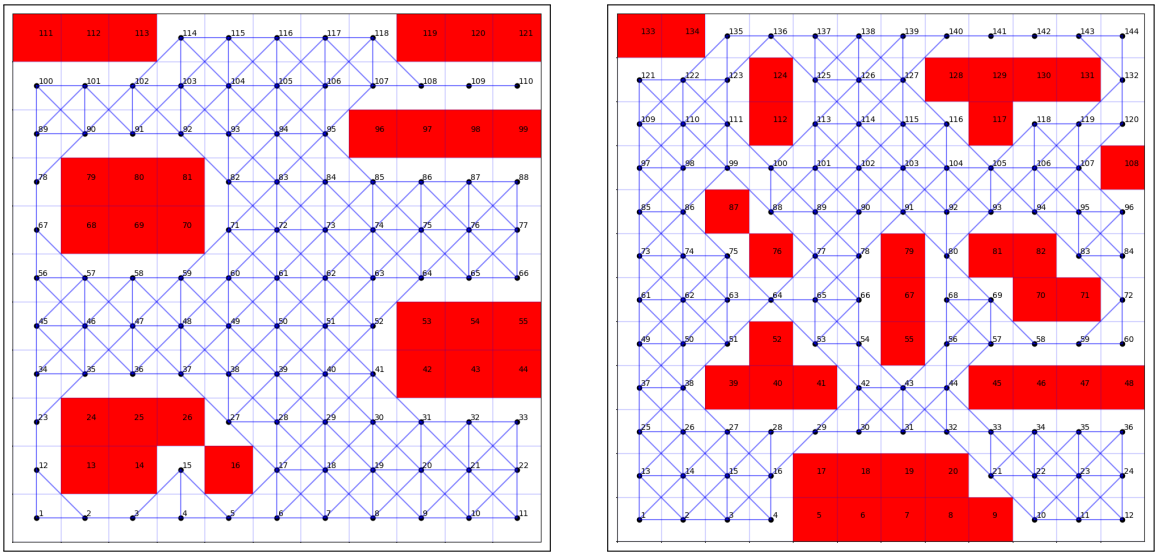
O consumo de energia por um drone é proporcional à distância percorrida, variações de altitude realizadas e ângulos formados pelas curvas feitas durante seu percurso.

A equação (2) calcula o custo da variação de altitude que ocorre durante o trajeto do ponto i para o ponto j (pontos adjacentes), onde k_h é uma penalização pela variação de altitude.

$$h_{ij} = |\frac{k_h(z_j - z_i)}{d(i,j)}| \tag{2}$$

Também são levadas em consideração as curvas feitas durante o percurso do drone, de forma que, quanto maior seja o ângulo da curva realizada, mais energia será gasta. Da forma que

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br



(a) Grafo de adjacências com $|V| = 93$ (b) Grafo de adjacências com $|V| = 108$

Figura 1: Exemplo de grafos de adjacências

o grafo G é construído, é possível a formação de ângulos de 0°, 45°, 90°, 135° e 180°. Tendo θ_{ijk} como o ângulo externo formado pelos pontos $i, j, k \in V$. O comprimento dos lados do triângulo formado pelos pontos i, j, k pode ser determinado pelas equações (3).

$$\begin{aligned} r &= (x_i - x_j)^2 + (y_i - y_j)^2 \\ s &= (x_j - x_k)^2 + (y_j - y_k)^2 \\ t &= (x_k - x_i)^2 + (y_k - y_i)^2 \end{aligned} \tag{3}$$

Sabendo o comprimento dos três lados do triângulo, é possível calcular o ângulo interno usando a Lei dos Cossenos. O ângulo externo formado pelos pontos i, j, k , ilustrado na Figura (2), pode ser determinado pela seguinte equação:

$$\theta_{ijk} = \pi - \arccos\left[\frac{r + s - t}{\sqrt{4rs}}\right] \text{ radianos} \tag{4}$$

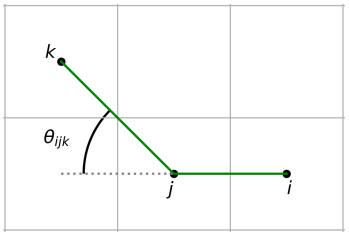


Figura 2: Ângulo externo formado pelos pontos $i, j, k \in V$

Convertendo θ_{ijk} de radianos para graus, tem-se:

$$angulo(i, j, k) = \begin{cases} \frac{180}{\pi}\theta_{ijk}, & \text{se } e_{ij}, e_{jk} \in E \\ \infty, & \text{caso contrário.} \end{cases} \tag{5}$$

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br

O custo q_{ijk} associado a uma curva entre os pontos i, j, k é proporcional ao ângulo da curva. Este custo é determinado pela equação (6), onde k_a representa o peso da penalização que é aplicada a cada curva realizada pelo drone durante seu trajeto.

$$q_{ijk} = k_a \times angulo(i, j, k) \tag{6}$$

Nas equações (2) e (6) são utilizados os parâmetros k_h e k_a que definem o peso das penalizações aplicadas às variações de altitude e curvas realizadas pelo drone, respectivamente. De acordo com [Modares et al., 2017], curvas feitas durante o vôo são responsáveis por cerca de 20% da energia total gasta pelo drone, assim, nesse estudo foi usado $k_a = 0.2$. A penalização por variação de altitude (k_h) deve ser o suficiente para o drone escolher o ponto com altitude mais próxima da sua posição atual. Assim, nesse trabalho foi usado $k_h = 2$.

Devido à presença de obstáculos, no grafo de adjacências pode acontecer situações de impasse, onde a única alternativa viável é deslocar-se para um ponto não adjacente. Esta circunstância é incompatível com a utilização direta da matriz de adjacências d_{ij}, uma vez que a distância entre dois pontos não adjacentes foi definida como infinita (Equação (1)). Por exemplo, no Grafo da Figura (1a), suponha que o drone visita os pontos 106, 107, 108, 109 e 110 (nesta ordem) e o ponto 118 ainda não foi visitado. Como o ponto 110 está em um local sem saída, será necessário determinar o ponto não adjacente (e não visitado) mais próximo do ponto 110. Assim, o drone estando no ponto 110 visitará o ponto 118 através do caminho $110 - 109 - 108 - 118$. Note que o drone passará pelos pontos 109 e 108 mais de uma vez.

A solução consiste em empregar um algoritmo para determinar o caminho mínimo entre dois pontos não adjacentes, sendo que o caminho deve passar exclusivamente por pontos adjacentes. Nesse contexto, nesse trabalho optou-se por utilizar o algoritmo de Dijkstra para calcular os caminhos mínimos entre todos os pares de pontos não adjacentes. Assim, a distância do caminho entre os pontos i e j não adjacentes, determinada pelo algoritmo de Dijkstra, é c_{ij}. Se os pontos i e j são adjacentes, $c_{ij} = d_{ij}$. Durante o cálculo dos caminhos mínimos pelo algoritmo de Dijkstra, são também calculadas as matrizes h_{ij} e q_{ijk} (penalizações por altitude e curvas), onde i, j e k representam pontos intermediários de um caminho.

Após calcular as distâncias c_{ij}, as penalizações por altitude h_{ij} e as penalidade pelas curvas q_{ijk}, apresenta-se um modelo matemático para o problema PCCEE. Para encontrar a rota do drone de menor cosumo de energia, o problema é formulado como um problema de Caixeiro Viajante, onde o drone deve visitar todos os pontos $i \in V$ evitando os obstáculos. As variáveis de decisão do modelo são:

- $x_{ij} = 1$ se o drone viaja do ponto i para o ponto j, $x_{ij} = 0$ caso contrário;

- u_i : índice da posição do ponto i na rota do drone (usado para a eliminação de sub-rotas).

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br

Considerando que $a \in V$ é o ponto de partida do drone, o modelo é:

$$\min \quad \sum_{i \in V} \sum_{j \in V, j \neq i} c_{ij} x_{ij} + \sum_{i \in V} \sum_{j \in V, j \neq i} h_{ij} x_{ij} + \sum_{i \in V} \sum_{j \in V, j \neq i} \sum_{k \in V, k \neq j} q_{ijk} x_{ij} x_{jk} \tag{7}$$

$$s.a: \quad \sum_{i \in V, i \neq j} x_{ij} = 1, \quad \forall j \in V; \tag{8}$$

$$\sum_{j \in V, j \neq i} x_{ij} = 1, \quad \forall i \in V; \tag{9}$$

$$u_i - u_j + |V| x_{ij} \leq |V| - 1, \quad \forall i, j \in V \setminus \{a\}, \quad i \neq j; \tag{10}$$

$$1 \leq u_i \leq |V| - 1, \quad \forall i \in V; \tag{11}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V. \tag{12}$$

A função objetivo (7) é formulada para minimizar a distância total percorrida pelo drone, as variações de altitude e as curvas realizadas ao longo do trajeto. As restrições (8) e (9) garantem que cada ponto é visitado uma única vez. As restrições (10) garantem que não há sub-rotas, forçando uma enumeração dos pontos da rota. As restrições (11) e (12) definem o domínio das variáveis de decisão.

O terceiro termo da função objetivo (7) contém o produto das variáveis x_{ij} e x_{jk}, o que torna uma função não-linear. Para obter um Modelo de Programação Linear Inteiro Misto (PLIM), o produto $x_{ij}x_{jk}$ é substituído pela variável binária y_{ijk} que terá valor 1 caso o drone forme um ângulo entre os pontos $i, j, k \in V$, e valor 0, caso contrário. Com isso, as seguintes restrições são adicionadas ao modelo:

$$y_{ijk} \leq x_{ij}, \quad \forall i, j, k \in V \tag{13}$$

$$y_{ijk} \leq x_{jk}, \quad \forall i, j, k \in V \tag{14}$$

$$y_{ijk} \geq x_{ij} + x_{jk} - 1, \quad \forall i, j, k \in V. \tag{15}$$

3. Heurística GRASP para o PCCEE

Nesse trabalho propõe-se uma heurística simples e eficiente baseada na meta-heurística GRASP [Feo e Resende, 1989] [Resende e Ribeiro, 2013]. GRASP é uma meta-heurística que iterativamente executa dois métodos: um método de construção (que combina uma técnica gulosa com escolhas aleatórias) e um método de melhoria ou busca local. O pseudocódigo da heurística GRASP proposta é apresentado no Algoritmo 1. O algoritmo recebe como entrada dois parâmetros: o parâmetro de aleatoriedade t (tamanho da lista de candidatos restritos) utilizado no método de construção e o parâmetro $nIter$ que define o número de iterações sem melhora (critério de parada). O método de Construção, baseada na Heurística de Inserção Mais Barata (HIMB) [Clarke e Wright, 1964], determina uma solução S' que é melhorada pela busca local 2_OPT. A solução S'', retornada pela busca local é utilizada para atualizar a melhor solução S encontrada pelo algoritmo. Se após $nIter$ iterações consecutivas não ocorrer mais melhorias na solução, o algoritmo retorna a melhor solução encontrada até o momento.

3.1. Heurística Construtiva

Para gerar uma solução inicial, utiliza-se a Heurística de Inserção Mais Barata (HIMB). Na versão da HIMB proposta nesse trabalho, a heurística inicia com uma rota parcial composta pelos pontos da fronteira do terreno a ser coberto (grafo G). Na Figura 3 mostra-se a rota inicial para uma instância com 121 pontos, sendo 28 obstáculos. Na versão convencional da HIMB, a partir da rota inicial, a cada iteração insere-se o ponto mais próximo à rota atual. A heurística finaliza quando

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br

Algorithm 1 : GRASP $(t, nIter)$

```
f(S) = ∞
it = 1
while it ≤ nIter do
    S' = Construção_HIMB(t)   //Construção Gulosa Aleatória
    S'' = Busca_Local_2_Opt(S')
    if f(S'') < f(S) then
        S = S''
        it = 1
    else
        it = it + 1
    end if
end while
Retorna S
```

todos os pontos válidos façam parte da rota. Ou seja, entre dois pontos adjacentes i e j, insere-se um ponto k, de tal forma que $d_{ki} + d_{kj} - d_{ij}$ seja mínimo. Com o intuito de gerar diferentes soluções a cada iteração do algoritmo GRASP, ao invés de selecionar o ponto mais próximo, seleciona-se aleatoriamente um ponto k dentre os t pontos internos mais próximos à rota atual. O parâmetro t é denominado tamanho da lista restrita de pontos candidatos.

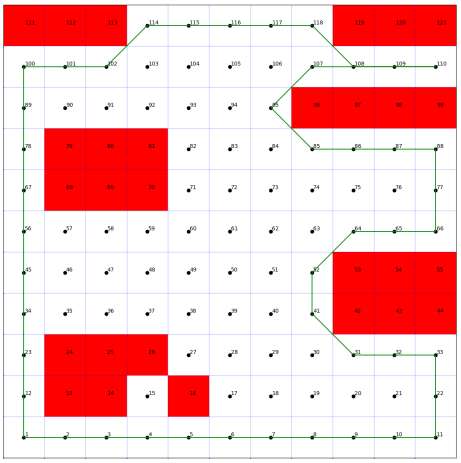


Figura 3: Rota inicial para a HIMB.

3.2. Busca Local 2_Opt

Na segunda etapa do algoritmo GRASP, é executada uma busca local (BL) baseada na vizinhança 2_Opt [Croes, 1958]. A BL é uma técnica iterativa que visa melhorar as rotas construídas através da troca de duas arestas. O algoritmo compara o custo da remoção de quaisquer duas arestas com o custo da adição de duas novas arestas que reconecta os dois segmentos da rota de forma a criar uma nova rota válida. Se o custo da nova rota for menor que o custo da rota original, a nova rota é aceita como a solução atual. Caso contrário, a mudança é descartada e a rota original é mantida. O processo é repetido até que nenhuma melhoria possa ser feita na solução atual.

4. Experimentos Computacionais

Nesta seção, são descritos os experimentos computacionais conduzidos para avaliar o desempenho da heurística GRASP. Nos experimentos também é analisado o desempenho da heurística da inserção mais barata (HIMB) puramente guloso (considerando o parâmetro de aleatoriedade $t = 1$). As soluções geradas pelas heurísticas são comparadas com as soluções do modelo PLIM

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br

(7)-(15) que é resolvido pelo solver GUROBI. Para fazer os testes, foram geradas instâncias com 36, 49, 81, 100, 121, 144, 169, 196, 225, 256 e 289 pontos. Para cada conjunto de pontos, foram criadas 10 instâncias distintas, onde a posição e a quantidade dos obstáculos variam. No total, foram geradas 120 instâncias. Cada célula do terreno a ser coberto possui uma área de $400m^2$, ou seja, o tamanho das células é $a \times b = 20 \times 20$. Como mencionado anteriormente, uma célula é considerada visitada quando o drone passa pelo seu ponto central, logo, as coordenadas (x_i, y_i) do primeiro ponto sempre serão $(10, 10)$. Similar a [Pimentel, 2018], as altitudes dos terrenos (coordenada z_i) variam no intervalo $[200m, 215m]$.

Todos os exeprimentos foram realizados utilizando um computador Ryzen 5 4600G 3.7 GHz com 16GB de RAM, sistema operacional Windows 10, 64 bits. As heurísticas GRASP e HIMB foram implementadas em C++17.

A qualidade das soluções geradas é avaliada pelo desvio percentual relativo (GAP) em relação à melhor solução obtida entre os métodos comparados. Para cada instância resolvida, o GAP é calculado conforme a Equação (16), onde f_M é valor da função objetivo obtido pelo método avaliado, e f_{best} é o melhor (menor) valor conhecido da função objetivo (encontrado entre os métodos comparados)

4.1. Parâmetros do GRASP

O algoritmo GRASP possui dois parâmeros: t (tamanho da lista restrita de candidatos) e $nIter$ (número de iterações sem melhora na solução atual, usado como critério de parada). Nos experimentos realizados, a cada execução da HIMB, o valor para t é escolhido aleatoriamente dentro do intervalo $[2, 20]$ (ou seja, foi usado uma lista restrita de candidatos de tamanho variável). Para escolher o valor do $nIter$, testes empíricos foram realizados utilizando três valores para $nIter$: $50, 100$ e 200. Durante as execuções dos casos de teste, observou-se que 50 iterações foram insuficientes para instâncias maiores, enquanto 200 iterações se revelaram desnecessárias na maioria dos casos. Por outro lado, 100 iterações se mostraram suficientes para encontrar uma melhoria significante em relação à solução atual.

4.2. Comparação de Resultados

Nesta seção são comparados os resultados das heurísticas GRASP e HIMB com os resultados do modelo PLIM (7)-(15) resolvido pelo solver GUROBI. Este solver foi aplicado para resolver o modelo PLIM com um tempo limite de CPU de 3600 segundos para cada instância. Ou seja, se após o tempo estabelecido nenhuma solução ótima for obtida, a melhor solução atual (limite superior) é retornada pelo GUROBI. O desempenho de cada método é avaliado pelo GAP em relação à melhor solução conhecida (obtida entre os três métodos). O GAP é calculado para cada uma das 120 instâncias utilizando a Equação (16).

$$GAP = 100 \times \frac{f_M - f_{best}}{f_{best}}\% \tag{16}$$

Na Tabela 1 são apresentados os resultados do GUROBI e das heurísticas HIMB e GRASP. A primeira coluna da tabela indica o número total de pontos para cada instância. As colunas subsequentes apresentam o número de solucoes ótimas obtidas pelo GUROBI e GRASP, as médias do GAP e as médias dos tempos de execução. Essas médias são obtidas para cada grupo de 10 instâncias. Valores em negrito representam as menores médias (melhores resultados).

Observa-se que, tanto o GUROBI quanto a heurística GRASP conseguiram encontrar as soluções ótimas para todos os conjuntos de instâncias com 36, 49 e 64 pontos. A diferença reside no tempo médio de execução. Para estes conjuntos de instâncias, a heurística GRASP foi aproximadamente 307, 962 e 3040 vezes mais rápida em relação ao GUROBI, respectivamente. Como a HIMB é uma heurística construtiva mais simples, ela apresenta os menores tempos médios entre

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br

Tabela 1: GAP(%) médio e tempo de execução dos métodos propostos

Conjunto de instâncias: $\|P\|$	n^o de Soluções ótimas		GAP (%) médio			Tempo médio em segundos		
	GUROBI	**GRASP**	**GUROBI**	**HIMB**	**GRASP**	**GUROBI**	**HIMB**	**GRASP**
36	10	10	**0,00**	7,83	**0,00**	6,15	**0,01**	0,02
49	10	10	**0,00**	8,55	**0,00**	53,74	**0,02**	0,06
64	10	10	**0,00**	12,50	**0,00**	364,90	**0,03**	0,12
81	9	9	0,05	12,83	**0,00**	1764,48	**0,06**	0,31
100	3	3	0,21	11,44	**0,05**	2650,86	**0,10**	1,31
121	1	1	1,11	10,87	**0,25**	3289,00	**0,17**	3,12
144	1	0	0,80	5,10	**0,63**	3519,50	**0,25**	5,16
169	0	0	2,84	6,63	**0,00**	3600,00	**0,45**	7,75
196	0	0	2,72	4,40	**0,13**	3600,00	**0,56**	9,27
225	0	0	15,70	4,65	**0,00**	3600,00	**1,36**	11,84
256	0	0	12,44	3,99	**0,00**	3600,00	**1,67**	14,46
289	-	0	-	2,07	**0,00**	-	**4,38**	27,59
Média	-	-	3,26	7,57	**0,09**	2368,07	**0,75**	6,75

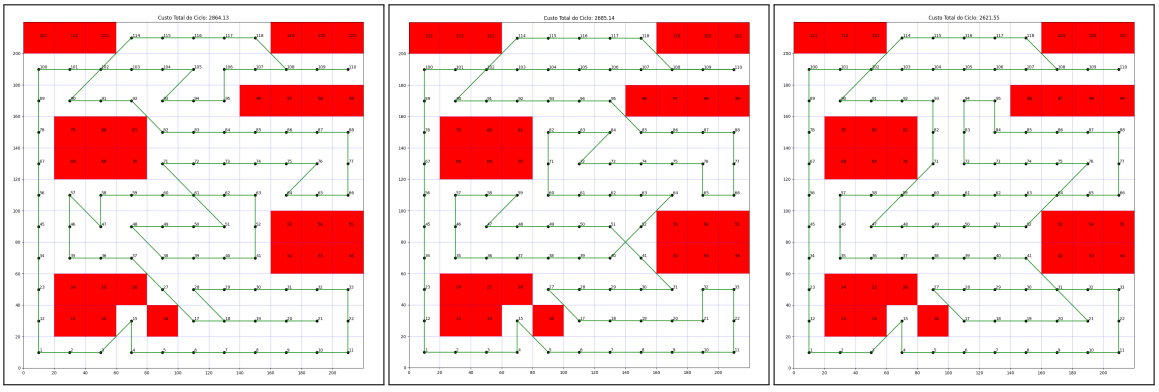
os métodos comparados. No entanto, a heurística HIMB não encontrou a solução ótima para nenhuma das instâncias testadas. Através dos resultados da HIMB é possível observar o percentual de melhoria obtida pela heurística GRASP (sabe-se que a cada iteração da GRASP, a solução a ser melhorada é construída usando a HIMB).

As médias do GAP, considerando todos os conjuntos de instâncias, foram 3,26%, 7,57% e 0,09%, respectivamente, para o GUROBI, HIMB e GRASP. Isto evidencia que o desempenho da heurística GRASP é superior que os outros métodos. A partir das instâncias com 81 pontos, o GUROBI começa a atingir o limite de tempo de execução em algumas instâncias, resultando em um aumento considerável do GAP médio. Para as instâncias com 169, 196, 225 e 256 pontos, o GUROBI não garante encontrar a solução ótima para nenhuma instância dentro do limite de tempo. Das 110 instâncias resolvidas pelo GUROBI, foram obtidas 44 soluções ótimas dentro do tempo de execução estipulado. O GRASP foi capaz de encontrar 43 dessas soluções ótimas. Para as instâncias com 289 pontos, o GUROBI não foi capaz de encontrar nenhuma solução viável dentro do limite de tempo, pois excedeu o limite de memória RAM disponível. Em contrapartida, a heurística GRASP foi capaz de encontrar soluções viáveis em tempos significantemente reduzidos.

Nas Figuras 4 e 5 são mostradas as rotas obtidas pelos métodos HIMB, GUROBI e GRASP, para instâncias com 121 e 144 pontos (grafos de adjacências das Figuras 1a e 1b), respectivamente. Nestas Figuras é possível analisar as diferenças entre as rotas encontradas pelos métodos. Para as duas instâncias, a heurística GRASP determinou as rotas de menor custo total, e a heurística HIMB gerou as rotas com pior custo total. Para a instância com 121 pontos (Figura 4), a rota determinada pelo GUROBI (Figura 4b) possui 34 curvas, das quais 14 têm ângulos externos de 90°, 9 de 135°, 10 de 45° e 1 de 180°. Em contrapartida, a rota encontrada pela heurística GRASP (Figura 4c) apresenta um total de 29 curvas, sendo 13 com ângulos externos de 90°, 6 de 135°, 9 de 45° e 1 de 180°. De acordo com a Equação (6), o parâmetro k_a é multiplicado pelo valor do ângulo externo formado pela curva, de modo que não há penalização quando o drone segue em linha reta, o que resulta em um ângulo de 0°. Ângulos de 45°, 90°, 135° e 180° recebem penalizações de 9, 18, 27 e 36, respectivamente. Somando as penalizações recebidas, as 34 curvas realizadas na rota do GUROBI gera-se um custo de 621, o que representa cerca de 23,13% do custo total da rota. Enquanto as 29 curvas da rota encontrada pela heurística GRASP gera custo de 513, representando aproximadamente 19,57% do custo total da rota. Neste exemplo mostra-se que a rota obtida pelo GRASP possui um menor número de curvas, principalmente curvas com ângulos externos de 135°,

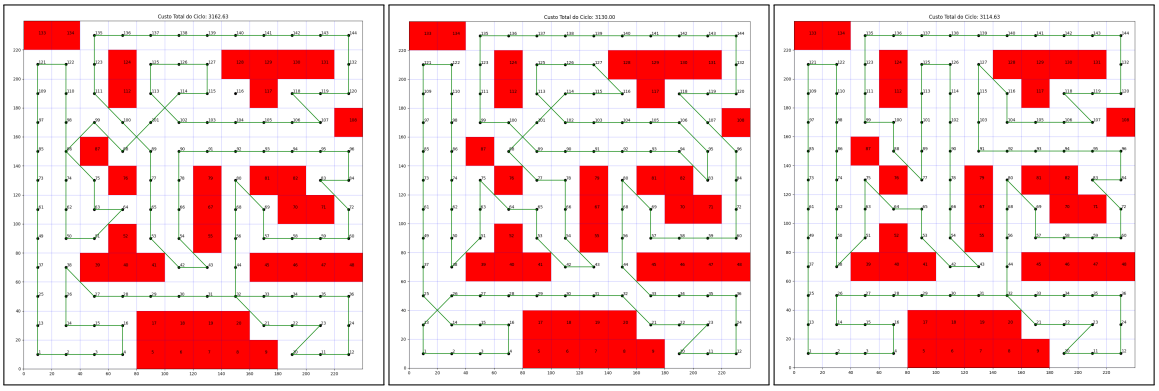
DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br

ou seja, a heurística GRASP consegue gerenciar as curvas realizadas pelo drone de maneira mais eficiente, minimizando o custo total (consumo de energia).



(a) Solução HIMB (b) Solução GUROBI (c) Solução GRASP

Figura 4: Soluções encontradas para uma instância de 121 pontos



(a) Solução HIMB (b) Solução GUROBI (c) Solução GRASP

Figura 5: Soluções encontradas para uma instância de 144 pontos

Como forma de validar os resultados obtidos, aplica-se o teste estatístico não paramétrico de Kruskal-Wallis utilizando o GAP como variável de resposta. Este teste é utilizado para verificar se há diferenças significativas entre dois ou mais grupos de dados independentes. Na Figura 6 mostra o gráfico de médias e os intervalos de Tukey da diferença honestamente significativa (Honestly Significant Difference - HSD), com 95% de confiança. Constata-se que, a heurística HIMB apresenta a maior média do GAP dentre os métodos propostos, enquanto a heurística GRASP demonstra o melhor desempenho, contando com um GAP consideravelmente menor, nunca ultrapassando 1% nos testes realizados. Mais uma vez, o melhor desempenho do algoritmo GRASP é claro. Seu intervalo de confiança não se sobrepõe aos intervalos dos outros algoritmos. Isso significa que o GRASP é estatisticamente melhor do que os outros métodos e por uma margem significativa.

5. Conclusões

Nesse trabalho foi abordado o problema do Planejamento de Caminho de Cobertura para drones que visa determinar a rota ou plano de voo para um drone cobrir uma área desejada minimizando o consumo de energia total (determinadas pela distância total, variações de altitude e

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br

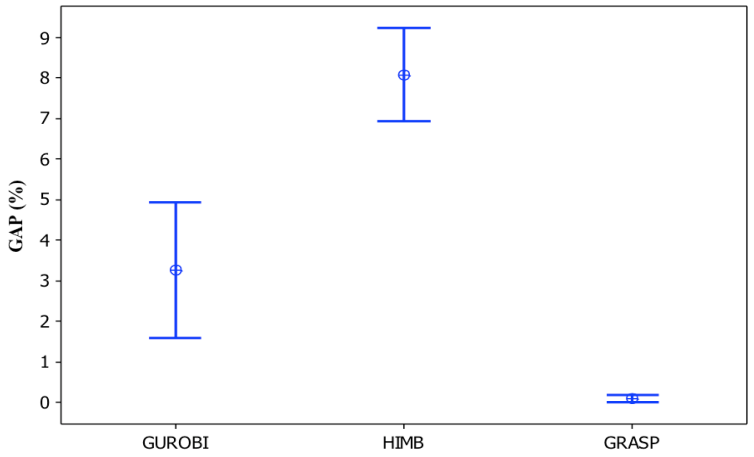


Figura 6: Gráfico de médias do GAP (%) e intervalos HSD.

mudanças de direção). O problema é classificado como um problema NP-Difícil, e para resolvê-lo foi proposto um algoritmo baseado na meta-heurística GRASP. Durante a fase de construção foi utilizada a heurística HIMB. Na etapa da busca local, é usado o algoritmo 2-Opt com o intuito de melhorar as rotas obtidas pela heurística construtiva. O modelo matemático do problema foi resolvido pelo solver GUROBI. Um conjunto de 120 instâncias foi utilizado para analisar o desempenho do algoritmo proposto. Os resultados, analisados estatisticamente, demonstram a alta eficácia do algoritmo GRASP para resolver o problema.

Como trabalho futuro poderá ser considerada uma frota de drones para cobrir a área desejada, obtendo um modelo similar ao do problema de roteamento de veículos.

Agradecimentos

Os autores agradecem o apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq.

Referências

Cabreira, T. M., Ferreira, P. R., Di Franco, C., e Buttazzo, G. C. (2019). Grid-based coverage path planning with minimum energy over irregular-shaped areas with uavs. In *2019 international conference on unmanned aircraft systems (ICUAS)*, p. 758–767. IEEE.

Chen, J., Du, C., Zhang, Y., Han, P., e Wei, W. (2021). A clustering-based coverage path planning method for autonomous heterogeneous uavs. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):25546–25556.

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

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

Di Franco, C. e Buttazzo, G. (2015). Energy-aware coverage path planning of uavs. In *2015 IEEE international conference on autonomous robot systems and competitions*, p. 111–117. IEEE.

Feo, T. A. e Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71.

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br

Gurobi Optimization, L. (2021). Gurobi optimizer reference manual.

Modares, J., Ghanei, F., Mastronarde, N., e Dantu, K. (2017). Ub-anc planner: Energy efficient coverage path planning with multiple drones. In *2017 IEEE international conference on robotics and automation (ICRA)*, p. 6182–6189. IEEE.

Osaba, E., Yang, X.-S., e Del Ser, J. (2020). Traveling salesman problem: a perspective review of recent research and new results with bio-inspired metaheuristics. *Nature-Inspired Computation and Swarm Intelligence*, p. 135–164.

Pimentel, S. B. (2018). *Drone Route Optimization using Constrained Based Local Search*. PhD thesis, Universidade Nova de Lisboa.

Resende, M. G. e Ribeiro, C. C. (2013). Grasp: Greedy randomized adaptive search procedures. In *Search methodologies: introductory tutorials in optimization and decision support techniques*, p. 287–312. Springer.

Shivgan, R. e Dong, Z. (2020). Energy-efficient drone coverage path planning using genetic algorithm. In *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*, p. 1–6. IEEE.

Valente, J., Sanz, D., Del Cerro, J., Barrientos, A., e de Frutos, M. Á. (2013). Near-optimal coverage trajectories for image mosaicing using a mini quad-rotor over irregular-shaped fields. *Precision agriculture*, 14:115–132.

Vasquez-Gomez, J. I., Herrera-Lozada, J.-C., e Olguin-Carbajal, M. (2018). Coverage path planning for surveying disjoint areas. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, p. 899–904. IEEE.

Xie, J. e Chen, J. (2022). Multiregional coverage path planning for multiple energy constrained uavs. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):17366–17381.

DOI: 10.59254/sbpo-2024-193533
https://proceedings.science/p/193533?lang=pt-br