UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO

Lorenzo Correia Maia

Estudo e Comparação de Algoritmos para Seleção de Nós Sobre o Problema de Escalonamento de Drone Paralelos com o Caixeiro Viajante

São Carlos - SP

2024

Lorenzo Correia Maia

Estudo e Comparação de Algoritmos para Seleção de Nós Sobre o Problema de Escalonamento de Drone Paralelos com o Caixeiro Viajante

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação da Universidade Federal de São Carlos, como requisito para a obtenção do título de Bacharel em Ciência da Computação.

Orientação Prof. Dr. Alan Demétrius Baria Valejo

São Carlos - SP

2024

Dedico este trabalho a minha família, amigos, padrinhos e Deus

Agradecimentos

Agradeço a todas as pessoas que fazem ou fizeram parte da minha jornada até aqui, tudo que sou é devido aos aprendizados e auxílios que recebi de todas elas.

Minha família e amigos, dos quais sempre tive boas conversas e um local seguro para descansar quando necessário. Em especial, minha mãe, Cleia, a pessoa mais paciente do mundo. Aos meus irmãos, Caio e Júlia, que são presentes a cada momento da minha vida. Ao meu grande amor, Mariany, que sem sombra de dúvidas é a pessoa que mais ouviu eu falar de grafos e sonhos. Aos meus amigos que são como irmãos, Nilto, que foi minha primeira grande amizade quando entrei na graduação, ao Richard, que faz o tempo passar mais rápido quando jogamos juntos e ao Gabriel, que tem as conversas mais engraçadas e aleatórias. Sem eles seriam muito mais árduos os dias.

Aos meus professores, que sem eles, não conseguiria avançar tanto. Em especial agradeço a quatro. Ao Valter e Renato, que enquanto estive no Programa de Educação Tutorial (PET), foram ótimos tutores, cada um ao seu modo de entender e guiar o grupo. Ao Mário que me aceitou como seu orientando de pesquisa, e me fez conhecer melhor estruturas tão belas como grafos. E ao Alan, meu orientador neste trabalho, que me acolheu na minha empreitada de fazer pesquisa, e que tem sido muito compreensivo.

Agradeço ao Osvaldo e meu tio, Domingos, que sempre me auxiliaram, e compreenderam todas minhas dificuldades nesses últimos cinco anos.

E claro, agradeço a Deus que me guia todos os dias.

“Eu sou porque nós somos“
(Líderes Ubuntu)

Resumo

O famoso problema do caixeiro viajante (do inglês, *Traveling Salesman Problem* TSP), é uma questão que sempre ronda grandes empresas que realizam entregas, e uma nova variável vem sendo adicionada nesta questão, que são os drones, pequenos veículos voadores não tripulados. O uso deles contribuiu com uma nova forma de se pensar o clássico problema do caixeiro viajante. Agora é possível realizar entregas, com um entregador e um drone, trabalhando para poderem ajudar mutualmente. Neste trabalho será abordada uma variante do problema, chamada Escalonamento de Drones Paralelos com o Caixeiro Viajante (do inglês *Parallel Drone Schedelingna* TSP, PDSTSP). O objetivo deste novo problema, é integrar em uma rota de entregas, que antes era atendida somente por um entregador humano, o uso do drone, de forma que enquanto a pessoa realiza o trajeto desenhado pelo caixeiro viajante, ele não precise visitar determinados clientes, pois o drone irá cobrir eles. Objetivamente, neste trabalho será comparada a questão, qual modelo de pré-seleção melhor escolhe os clientes que o drone deve visitar e retornar diretamente ao ponto de origem. Para realização de tal comparação, serão utilizados dois algoritmos, um guloso e um aleatório, que dada uma solução TSP, irá selecionar os clientes a serem atendidos pelo drone. Por sua vez, será utilizado de bibliotecas do Python para solucionar o TSP. Por fim, teremos um valor que corresponda ao tempo que o último veículo retorna à origem, após seus trabalhos. Os resultados indicaram que o algoritmo guloso utilizado teve um melhor desempenho na pré-seleção de nós.

Palavras-chave: Problema de Roteamento de Veículos; Grafos; Algoritmos;

Abstract

The famous Traveling Salesman Problem (TSP) is a challenge that often arises in large companies that handle deliveries, and a new variable is being added to this issue: drones, small unmanned aerial vehicles. Their use has contributed to a new way of thinking about the classic Traveling Salesman Problem. It is now possible to make deliveries with a delivery person and a drone working together to mutually assist each other. This work will address a variant of the problem, called Parallel Drone Scheduling with the Traveling Salesman Problem (PDSTSP). The goal of this new problem is to integrate the use of a drone into a delivery route that was previously served only by a human delivery person. While the person follows the route designed by the Traveling Salesman, they no longer need to visit certain customers because the drone will cover them. Specifically, this work will explore the question of which pre-selection model better chooses the customers that the drone should visit and return directly to the origin. To perform this comparison, two algorithms will be developed: a greedy algorithm and a random algorithm. Python libraries will be used to solve the TSP. Finally, we will obtain a value corresponding to the time it takes for the last vehicle to return to the origin after completing its tasks. The results indicated that the greedy algorithm performed better in the pre-selection of nodes.

Keywords: Vehicle Routing Problem; Graphs; Algorithm;

Lista de ilustrações

Sumário

1 Introdução

Nos últimos anos, empresas que possuem uma grande carga de entregas de mercadorias, vem desenvolvendo ferramentas para auxiliar nesta tarefa, entre elas, é a utilização de veículos aéreos não tripulados, ou como ficaram popularmente conhecidos, os drones. Com potencial de revolucionar a área de roteamento, sendo capazes de realizar entregas, de devido a isso, algumas novas problemáticas na área de roteamento de veículos surgiram (MURRAY; CHU, 2015).

Um famoso desafio de roteamento é o Problema do Caixeiro Viajante, ou do inglês, *Traveling Salesman Problem* (TSP), que se baseia no seguinte cenário. Existe um vendedor de objetos quaisquer, que deve sair de sua cidade e passar por todas as vizinhas para tentar vender seus produtos, de forma que ele não repita nenhuma cidade e, ao fim, ele retorne à sua origem.

Com o desenvolvimento dos drones, novas variantes do clássico TSP surgiram, entre elas o Escalonamento de Drones Paralelos com TSP, do inglês *Parallel Drone Scheduling TSP* (PDSTSP), que foi apresentado por Murray e Chu Murray e Chu (2015). Nesta configuração, enquanto o caminhão percorre a rota principal, os drones são usados para realizar entregas em clientes específicos, retornando ao ponto de origem após completar suas tarefas. Esta abordagem visa aumentar a eficiência do processo de entrega, reduzindo o tempo total para que o último veículo retorne ao depósito, essa variável é chamada de *makespan*.

1.1 Objetivo

Este Trabalho de Conclusão de Curso visa executar algoritmos que irão selecionar nós, de uma dada solução TSP, assim formando uma solução PDSTSP, e comparando diferentes abordagens ao escolher os destinos dos drones. Sendo elas o uso de um algoritmo guloso que fará escolhas ótimas locais em busca de minimizar o tempo de retorno de todos os veículos, e um aleatório que almeja selecionar aleatoriamente os nós visitados pelos drones. Mediante experimentos computacionais, é avaliado o desempenho do *makespan* após os vértices serem selecionados em ambos os algoritmos, além de comparar o tempo de execução entre eles.

Os algoritmos mencionados serão chamados de pré-seleção, em que eles irão decidir os vértices para o drone visitar, assim os retirando da rota do caminhão. Para uma comparação mais robusta, os algoritmos de (MURRAY; CHU, 2015) para o PDSTSP serão replicados, para ser possível observar os valores que o *makespan* irá retornar, além

de servir de limiar para os algoritmos de pré-seleção.

1.2 Objetivo específico

Como citado, o objetivo final do trabalho é comparar o resultado do tempo de seleção e o tempo total do retorno do último veículo. E temos a motivação para a escolha destes parâmetros.

O tempo que os algoritmos levam para selecionar os nós que serão atendidos pelo drone, é importante devido à relevância de como cada algoritmo consegue entregar os resultados, e como o objetivo intrínseco do trabalho é mostrar qual se sairá melhor, usar o tempo como fator de comparação é importante. Principalmente quando varia o tamanho do grafo que será trabalhado.

O tempo de chegada do último veículo ao depósito é importante, pois com ele é possível buscar um equilíbrio entre as visitas do drone e o caminho do caminhão, ou seja, fazendo com que nenhum dos dois precise de um tempo muito maior que o outro para cumprir os objetivos. Para mais, é importante notar que não é uma minimização apenas do caminhão, pois caso fosse, utilizar somente o drone resolveria o problema, mas como será futuramente abordado, tal escolha não é ótima para o problema.

2 Fundamentação Teórica

Este capítulo contém os conceitos que são abordados neste trabalho. Além disso, serve como um guia para entender os assuntos abordados.

2.1 Grafos

Um grafo G é uma tupla composta por dois outros conjuntos, V e E, assim sendo $G(V, E)$. O conjunto finito V é definido da seguinte maneira, $V = \{v_1, v_2, \dots v_n\}$, em que cada n elemento são os vértices. Por sua vez o conjunto finito E é representado como, $E = \{e_1, e_2, ..., e_m\}$, onde os m elementos são pares ordenados ou não, que representam as arestas entre os nós. Importante ressaltar que, dois vértices v_i e v_j quaisquer, estão ligados por uma aresta e_k, e pode ser representado como $e_{i,j}$.

A importância dos grafos vem das inúmeras possibilidades de se representar relacionamentos e estruturas, respectivamente, redes sociais e redes de computadores. Para esse trabalho, essa estrutura é imensamente importante, pois com ela conseguimos representar destinos de entregas e caminhos que podem ser feitos para cumprir tal objetivo.

Existem tipos de grafos diferentes, a seguir alguns que são mais comuns:

- Direcionados: as arestas possuem direção, tendo uma orientação específica, em que indica uma origem e um destino;
- Não direcionados: os nós da relação podem ir e vir pela mesma aresta;
- Ponderados: as arestas possuem pesos, o que pode indicar, por exemplo, distância entre nós;
- Não ponderados: todas as arestas possuem o mesmo peso.

Em um grafo, existe a possibilidade de percorrê-lo de forma e com objetivos diferentes, entre elas, o caminho é um conceito muito útil. O caminho é uma sequência de nós conectados por arestas, em que seja possível ir de um a outro. Formalmente, se existe um caminho entre dois nós v_1 e v_k, podemos representá-lo como uma sequência de vértices e arestas, onde cada par de vértices consecutivos está conectado por uma aresta.

Por exemplo, um caminho P que conecta os nós $v_1, v_2, \dots, v_k$ pode ser descrito como:

$$P = v_1 \to v_2 \to v_3 \to \cdots \to v_k$$

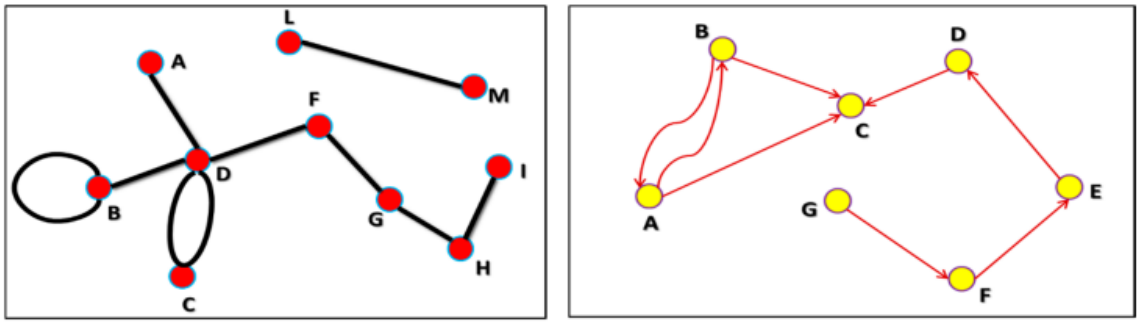
Ou seja, para todo $i = 1, 2, \ldots, k-1$, existe uma aresta $e_{i,i+1}$ que conecta v_i com v_{i+1}.

A seguir alguns exemplos de caminhos famosos da literatura.

- Caminho euleriano: um caminho que visita todas as arestas do grafo exatamente uma vez;
- Caminho hamiltoniano: um caminho que visita todos os nós, exatamente uma vez;
- Ciclo: um caminho que começa e termina no mesmo nó.

Na Figura 1 a seguir, temos dois exemplos de grafos, o (a) é um grafo não direcionado, enquanto o item (b) é um grafo direcionado, notem as setas que tem em uma das pontas das arestas, isso indica a direção da relação.

Figura 1 – Exemplo de grafos direcionados e não direcionados.



(a) Example of an undirected graph. **(b)** Example of the directed graph.

Fonte: (MAJEED; RAUF, 2020).

2.2 NP-dificuldade

Em 2000 foi publicado pelo *Clay Mathematics Institute* (CMI), sete problemas matemáticos extremamente desafiadores, na qual se alguém conseguisse resolver um deles, seria premiado com um montante em dinheiro (LOPES, 2017). Entre os problemas, um deles é interessante para a computação, sendo a questão P versus NP.

O problema P versus NP, consiste em provar que a classe P (do inglês, *Polynomial-time*) é igual ou diferente a classe NP (do inglês, *Nondeterministic Polynomial-time*). Os problemas que moram em P são possíveis de serem resolvidos em tempo polinomial, enquanto os que estão em NP, são verificáveis em tempo polinomial, se suas respostas estão certas.

Além do primeiro contato com P versus NP, existem outros conjuntos de problemas, entre eles. NP-Difíceis são importantes, pois são difíceis como os que estão em NP, porém

não obrigatoriamente moram em NP, ou seja, problemas NP-Difíceis podem ou não serem solucionados em tempo polinomial.

2.3 Problemas clássicos

Alguns problemas famosos que moram em NP-Dificuldade, são o problema da mochila, árvore de Steiner, escalonamento de máquinas paralelas e o TSP. O problema da mochila se propõem em enfrentar o seguinte cenário, existem itens com pesos e valores individuais, e o objetivo é colocar o maior número deles em uma mochila que tem uma capacidade mor, de forma que maximize o valor total dessa mochila.

A árvore de Steiner tem como alvo final, obter uma árvore mínima que tenha todos os nós terminais de um grafo, podendo ter nós intermediários. O interessante desse problema é a aplicabilidade dele, sendo possível em áreas como redes de comunicação e planejamento de rotas.

O problema de escalonamento de máquinas consiste em dividir um conjunto de tarefas para serem realizadas de forma simultânea por várias máquinas, para otimizar um critério, sendo o tempo total de execução ou o balanceamento do tempo que cada máquina irá executar a atividade. Este problema é particularmente relevante para este trabalho, pois, juntamente com o TSP, serve como base para o modelo de (MURRAY; CHU, 2015).

2.3.1 Construção de Grafo Aleatório

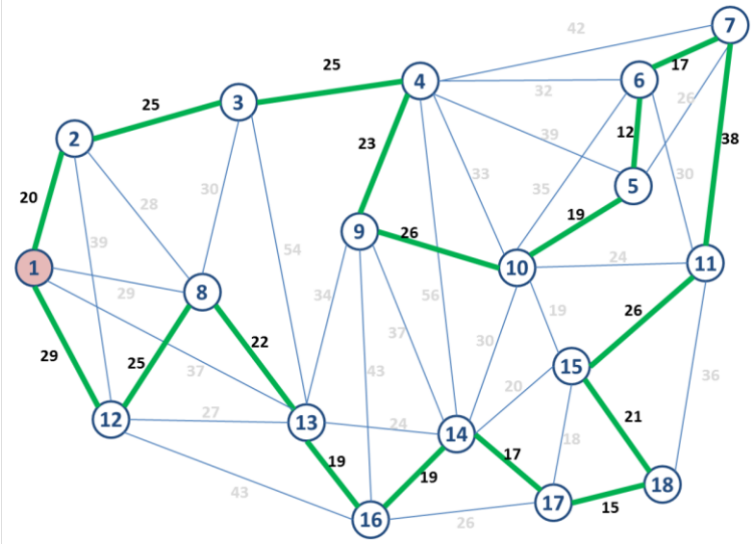
Construção de grafos aleatórios é útil para diversas áreas, desempenhando um papel importante neste trabalho. Tal problema proposto por (ERDŐS; RÉNYI, 1959) diz que dado um conjunto com n vértices é possível criar um grafo com N arestas que são selecionadas aleatoriamente. Neste modelo, cada par de vértices tem a mesma probabilidade de serem conectados por uma aresta. A seleção das arestas é feita de forma independente, repetindo-se o processo até que o grafo esteja completamente conectado, ou seja, sem nenhum vértice isolado. Esse método garante que cada grafo possível, dado o conjunto inicial de vértices, tenha a mesma probabilidade de ser formado. Este processo permite estudar as propriedades estatísticas e estruturais dos grafos aleatórios de forma rigorosa.

2.3.2 TSP

O problema do caixeiro viajante consiste em um vendedor que terá a seguinte missão, partir de sua cidade de origem, e percorrer diversas outras, até que retorne ao ponto inicial, de forma que ele não repita nenhuma cidade no caminho. O TSP é um problema que pode ser aplicado em várias situações, como circuitos eletrônicos e roteamento de veículos.

Para o TSP, uma abordagem que é encontrada, é utilizar grafos, pois com ela é possível representar as cidades como nós e as estradas como arestas. A seguir na Figura 2 um exemplo de uma solução para o TSP.

Figura 2 – Solução para o TSP. Imagem retirada do artigo (PARDINI, 2015).



Na Figura 2, podemos notar que os nós estão numerados de 1 a 18, e existem diversas arestas com pesos, assim sendo um grafo ponderado. Para a solução do TSP, temos que o nó 1 destaco em vermelho é o ponto de origem do ciclo, e as arestas que estão na coloração verde, é o caminho em si.

2.4 Programação Linear Inteira

Os problemas citados anteriormente podem ser resolvidos utilizando Programação Linear Inteira (PLI), que consiste em definir uma função objetivo a ser otimizada e um conjunto de restrições e variáveis que irão convergir para uma solução do problema. A PLI deve assumir suas variáveis com valores inteiros.

2.4.1 TSP resolvido com PLI

Para uma fundamentação ampla e consistente do conceito do TSP, vamos solucionar o problema utilizando PLI.

Uma solução para o TSP é um ciclo Hamiltoniano. A seguir a formulção em PLI do TSP baseada no trabalho de (DANTZIG; FULKERSON; JOHNSON, 1954).

$$\min \quad \sum_{e \in E} w_e x_e \tag{2.1}$$

$$\sum_{e \in \delta(v)} x_e = 2, \qquad \forall v \in V \tag{2.2}$$

$$\sum_{e \in \delta(S)} x_e \geq 2, \qquad \forall S : 0 \neq S \subseteq V \tag{2.3}$$

$$x_e \in \{0, 1\}, \qquad \forall e \in E \tag{2.4}$$

Dado o problema, precisamos de uma entrada que é um grafo $G = (V, E)$. É necessário definir uma função que nos dá o peso das arestas, podemos chamar de w, onde, w_e representa a e-ésima aresta. A função objetivo está definida em (2.1) que é a soma de todas as arestas que fazem parte do ciclo final.

A restrição (2.2) indica, que ao final da iteração todos os vértices terão grau 2, ou seja, teremos uma aresta entrando e outra saindo. A segunda restrição (2.3) é um subconjunto S que representa o corte atual do circuito, sendo necessário ser atravessado por duas ou mais arestas. A restrição ajuda na construção da solução, pois com ela vamos formando gradualmente o circuito. Por fim, x é uma variável que pode assumir o valor de 0 ou 1,e representa se uma aresta não faz parte da solução ou faz, respectivamente,

2.5 Categorias/Variedades de Algoritmos

Os algoritmos são uma ferramenta muito importante na área da computação, para esse trabalho se buscou conhecer e usar de tipos variados de algoritmos. Essa subseção pretende explicar de forma geral como essas categorias funcionam e por qual motivo foram selecionadas.

2.5.1 Algoritmos Gulosos

Os algoritmos gulosos ou míopes, fazem a melhor escolha localmente, na esperança de que essa sequência de escolhas conduza à solução ótima global (CORMEN et al., 2012). Imagine um vale com várias colinas, e o seu objetivo é encontrar a melhor colina entre elas, que nesse caso será a mais alta. Você parte de um ponto e deve começar seu trajeto, no qual já existe uma escolha para se fazer, a direita tem uma escada que leva para baixo e à esquerda, um pequeno morro. Um algoritmo guloso, escolheria a esquerda, pois com isso teria um ganho de altura, e a cada novo ponto, essa escolha se repete até chegar em um ponto que não dê para melhorar.

2.5.2 Algoritmos Aleatórios

Os algoritmos aleatórios são interessantes, primeiramente por sua natureza, pois ao tentar aplicar uma certa aleatoriedade em um computador, nos enfrentamos com um dilema, tal máquina é determinística, ou seja, diferentemente de uma pessoa, quando pedida para falar um número, esse que será aleatório, o computador não conseguirá fazer o mesmo. Para tentar gerar um número aleatório ou próximo disso, é utilizado uma premissa conhecida como geração de números pseudoaleatórios.

Os números pseudoaleatórios são gerados a partir de um valor conhecido como semente, ele dá a base para que números sejam gerados em uma certa sequência. Se existirem dois algoritmos diferentes, que se utilizam de modelos aleatórios e ambos usarem a mesma semente, eles obterão o mesmo resultado. Para tentar chegar o mais próximo de uma geração aleatória de números, é utilizado como sementes, por exemplo, o horário que foi executado um algoritmo, casas decimais do número pi, valores de longitude e latitude com grande precisão e entre outras abordagens.

A importância de se utilizar de valores aleatórios em algoritmos pode ser uma escolha de modelagem, e auxilia na construção de códigos não exatos, como heurísticas e meta-heurísticas. Pegamos como exemplo o problema da mochila, nele como podemos começar a explorar o conjunto de itens? Pegamos o que tem maior valor, ou que tem uma melhor razão entre peso e valor? Para qualquer uma dessas opções, encontrar o melhor item a começar, pode ser infrutífero, pois o tamanho do escopo pode fazer com que demore muito. Uma possibilidade é usar um algoritmo aleatório para começar de alguma região e pegar o melhor de lá, e assim replicando até concluir o objetivo do dilema.

3 Revisão da Literatura

Para o avanço deste trabalho, uma etapa fundamental é a revisão da literatura sobre o tema abordado. Essa seção contém os detalhes e resultados encontrados.

3.1 Mapeamento Bibliográfico

Para realizar a pesquisa de artigos, foram utilizadas duas principais ferramentas, o portal de periódicos (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, 2024) (CAPES) e o *Google* Acadêmico (Google, 2024). Nas duas plataformas, foram utilizados descritores para encontrar o material alvo, sendo eles:

- TSP com drones;
- TSP *with* drones;
- PDSTSP;
- TSP *with* drones PDSTSP.

Usando o primeiro descritor, foram encontrados poucos resultados, apenas dois, devido ao mau desempenho, as próximas buscas foram feitas com apenas palavras na língua inglesa, o que gerou um número maior de resultados na CAPES. A segunda e terceira buscas mostraram 81 e 7 resultados, enquanto a última busca não apresentou nenhum resultado. Dos artigos encontrados, alguns foram selecionados para analise. Usando a plataforma do *Google* Acadêmico, os resultados foram mais abrangentes, passando de 3.000 consultas.

Durante o mapeamento dois autores foram citados em todos os artigos, a partir de descobertos, foi feita uma analise mais profunda sobre o trabalho deles, tornando a principal referência.

3.2 Principal Referência

Com o avanço dos drones e o aumento do uso dessa tecnologia para realizar entregas, surgiu a oportunidade do TSP receber uma nova formulação, o TSP com Drones (TSP-D). O novo modelo consiste em inserir um drone ou mais para otimizar o clássico problema.

Buscando na literatura referências para o TSP-D, encontramos a principal fonte que este trabalho se baseia, a formulação proposta por Murray e Chu (MURRAY; CHU,

2015). No trabalho deles, temos dois problemas sendo definidos e tratados. O TSP com Ajudante Voador, do inglês *Flying Sidekick* TSP (FSTSP), é o primeiro a ser mencionado. O segundo problema é o Escalonamento de Drones Paralelos com TSP, em inglês *Parallel Drone Schedelingna* TSP (PDSTSP), que será o principal foco deste trabalho.

O FSTSP consiste na utilização de um único drone para auxiliar o entregador, que é um caminhoneiro, a entregar pedidos de clientes. O drone pode partir junto ao caminhão ou ser lançado do depósito. Quando o drone visita um cliente, ele deve retornar ao caminhoneiro em algum ponto futuro da rota dele. Após a recuperação, o drone é carregado pelo condutor e, caso necessário, lançado novamente para atender a um cliente e retornar ao caminhoneiro. As ações do drone são representadas como uma tripla $< i, j, k >$, sendo i o ponto de lançamento, j um cliente e k um ponto de recuperação do drone que faz parte da rota do caminhão.

O PDSTSP combina dois outros problemas, sendo o primeiro o escalonamento de máquinas paralelas e o TSP. O escalonamento de máquinas paralelas é devido ao uso dos drones funcionando de forma independente, pois eles são lançados do depósito, atendem um cliente e retornam imediatamente ao ponto de origem, sem interagirem diretamente com o caminhão que está realizando as demais entregas seguindo um caminho do TSP. Sendo assim, a sua representação é composta por dois vetores, sendo um deles a sequência de nós que o caminhão irá visitar e o outro os pontos atendidos pelos drones.

Para o PDSTSP o valor a ser otimizado é o *makespan*, que é o tempo que o último veículo retorna ao depósito assim minimizando tanto o tempo dos drones como do caminhão. Para isso é otimizado a locação de clientes para os drones e para o caminhão, criando um balanceamento da carga de trabalho entre eles.

3.3 Análise da Revisão da Literatura

Sobre o problema do TSP-D, existem características em comum e particularidades para serem comentadas. O critério de seleção de um artigo foi com base na semelhança da avaliação de otimização dele com o que será abordado neste trabalho. A minimização do tempo do *makespan*. A seguir uma tabela com as principais informações a cerca dos artigos selecionados.

* No FSTSP foi usado apenas um drone enquanto no PDSTSP se usou múltiplos.

** A quantidade de drones varia de acordo com a demanda, podendo ser usado apenas um ou vários ao mesmo tempo.

As estratégias mais utilizadas foram programação linear inteira mista, em inglês, *Mixed-Integer Linear Programming* (MILP) e programação inteira mista, do inglês, *Mixed Integer Programming* (MIP). As duas abordagens são capazes de modelar problemas

Tabela 1 – Referências de estratégias e tipos de drones

ID	Referência	Estratégia	Tipo
1	(MURRAY; CHU, 2015)	MILP	FSTSP/PDSTSP
2	(VÁSQUEZ; ANGULO; KLAPP, 2021)	MIP	FSTSP
3	(BOUMAN; AGATZ; SCHMIDT, 2018)	PD	FSTSP
4	(CAVANI; IORI; ROBERTI, 2021)	MILP	FSTSP
5	(LU et al., 2022)	K-means	FDTSP
6	(KITJACHAROENCHAI et al., 2019)	MIP	FSTSP
7	(HA et al., 2018)	Alg. Genético	FSTSP
8	(MORANDI et al., 2022)	MILP	FSTSP
9	(DELL'AMICO; MONTEMANNI; NOVELLANI, 2020)	MILP	PDSTSP
10	(MONTEMANNI; DELL'AMICO; CORSINI, 2024)	MILP e PR	PDSTSP
11	(KLOSTER et al., 2023)	MILP	TSP - DS
12	(KIM; MOON, 2018)	MIP	TSP - DS

Tabela 2 – Referências de estratégias e tipos de drones

ID	Dados	Objetivo	N° de Drones
1	Própria	Minimizar o *Makespan*	Somente 1*
2	Bouman et al.[a]	Resolver com Método Exato o TSP-D	Somente 1
3	Agatz et al. [b]	Resolver com PD o TSP-D	Somente 1
4	Derividas de Poikonen et al [c]	Resolver com Método Exato o FSTSP	Multiplos
5	Própria	Minimizar o Tempo de Concluir o FDTSP	Multiplos
6	Própria	Minimizar o Tempo de Concluir o FSTSP	Multiplos
7	Própria	Minimizar o Custo de Operação do FSTSP	Somente 1
8	Própria	Encontrar Benefícios em Repetir Arcos	Multiplos
9	Derivadas de Murray e Chu [d]	Resolver o PDSTSP com Meta-Heurísticas	Multiplos
10	Derivadas de Nguyen et al.[e]	Minimizar o *Makespan*	Multiplos
11	Própria	Explorar Soluções para o TSP-DS	Multiplos**
12	Própria	Minimizar o *Makespan* do TSP-DS	Multiplos**

[a] (BOUMAN; AGATZ; SCHMIDT, 2018)
[b] (AGATZ; BOUMAN; SCHMIDT, 2018)
[c] (POIKONEN; GOLDEN; WASIL, 2019)
[d] (MURRAY; CHU, 2015)
[e] (NGUYEN; HÀ, 2023)

baseados no uso de uma função objetivo, que usa variáveis que assumem valores inteiros ou lineares, a depender da abordagem nomeada e restrições para atingir uma maximização ou minimização da função. A utilização de programação dinâmica (PD) e Algoritmo genético apareceram uma vez, como abordagens para lidar com o FSTSP. PD se baseia na estratégia de dividir um problema em pequenas partes, otimizando-as e encontrando uma solução para a questão original. Algoritmos genéticos se baseiam na teoria da evolução proposta por Darwin (DARWIN, 2009), e utilizam de uma população que cada indivíduo representa uma solução para o problema abordado, a partir é realizado o cruzamento dos indivíduos gerando novas gerações, é repetido essa ação até que se tenha uma solução adequada. Sobre o uso de programação por restrição (PR), foi encontrado um artigo com o uso da técnica, que consiste em satisfazer restrições, visando encontrar valores para o conjunto de variáveis que respeitem as restrições. O uso de *k-means* é encontrado uma vez, essa estratégia é feita para agrupar dados semelhantes no mesmo grupo, no caso do artigo, ele é usado para agrupar os clientes para otimizar o trabalho do caminhão e drones.

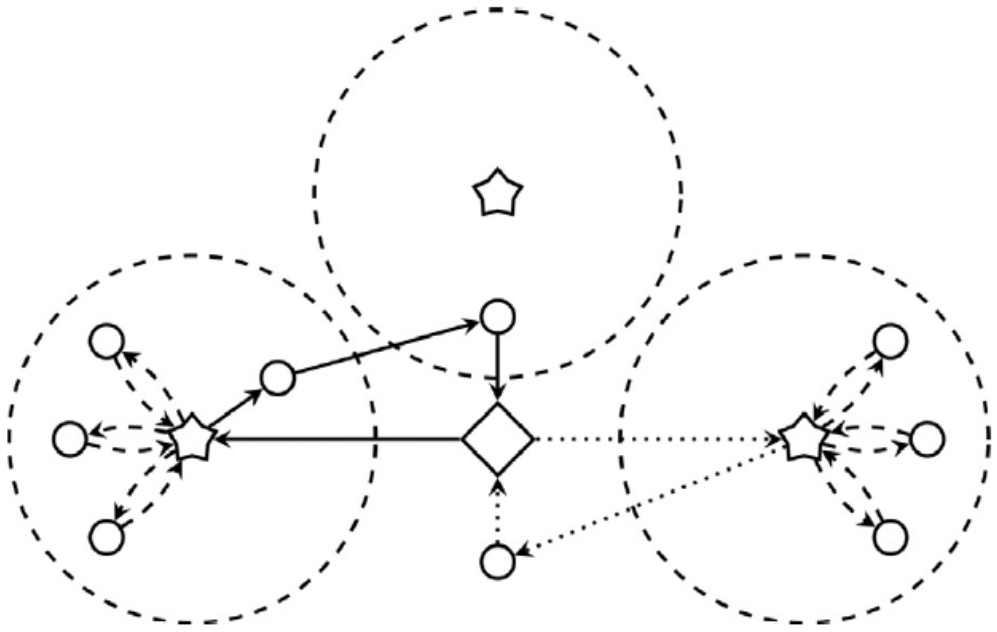
Na maioria dos artigos analisados, a base de dados para testar as hipóteses foram feitas pelos próprios autores, sendo que na construção delas foram consideradas características para o problema, como limitar o número de clientes a um valor fixo. Nesses casos, a geração aleatória do local dos clientes foi presente nos artigos.

O TSP com drones flexíveis, em inglês *Flexible Drones TSP* (FDTSP), é uma variação do FSTSP, que ao invés de utilizado o TSP clássico, é utilizada uma abordagem diferente, sendo ela a utilização de mais de um entregador para realizar as tarefas, de forma que cada caminhoneiro esteja responsável por uma região.

Sobre os objetivos encontrados, existem diferentes formas de abordar um problema TSP-D. A principal foi a minimização do tempo do *makespan* ou de concluir a abordagem atacada. Além disso, temos a resolução do problema sendo atacada utilizando métodos como PD e métodos exatos, e a exploração de características que podem auxiliar nos problemas atacados, como repetir arcos e explorar diferentes soluções para o problema.

Sobre o PDSTSP, foi encontrada uma variação que é o TSP com Estação de Drones, em inglês *TSP with Drone Station* (TSP - DS). É uma adaptação do problema, em que ao invés do drone partir do depósito, ele ficaria em uma estação especifica para ele, e as encomendas respectivas a eles seriam depositadas na estação e depois entregues pelos drones para os clientes. A seguir o exemplo de funcionamento do TSP-DS.

Figura 3 – Exemplo de TSP - DS.



Fonte: (KLOSTER et al., 2023).

A Figura 3 descreve o funcionamento do TSP-DS, sendo o quadrado o depósito onde os caminhões partem, as estrelas as estações de drones e os círculos os clientes a serem

atendidos. As linhas solidas representam uma solução TSP que será feita pelo caminhão. As linhas tracejadas são os drones visitando os clientes, enquanto o círculo tracejado é o alcance que a estação de drones possui.

4 Metodologia

O objetivo desse capítulo é informar sobre a base de dados, algoritmos e métodos utilizados durante o trabalho.

4.1 Ambiente de execução

O ambiente utilizado para desenvolver os algoritmos e realização de testes foi o Google Colab[1], enquanto a linguagem de programação escolhida foi a Python[2], em que a biblioteca mais importante foi a *NetworkX* (nx).

A escolha de tais ferramentas é devido à facilidade de execução que o Google Colab é capaz de proporcionar, além de auxiliar em um desenvolvimento colaborativo. A linguagem de programação Python, por ser de alto nível (SCUDERO, 2017), facilita a escrita de códigos de forma concisa e eficiente. Além disso, a biblioteca *NetworkX*, que está implementada em Python, é especialmente adequada para manipular e exibir grafos, tornando o trabalho com essas estruturas mais intuitivo e acessível.

4.2 Base de dados

A base de dados utilizada veio de uma experiência anterior do orientando, que realizou uma Iniciação Cientifica (IC) também envolvendo grafos. A base consistia em arquivos do tipo texto, em que cada um deles tinha a descrição dos nós, em relação à longitude e latitude, e incluindo o depósito. Os arquivos eram estruturados da seguinte forma:

- Velocidade do caminhão
- Velocidade do drone
- Número de nós
- Destaque para o depósito
- Restantes dos nós

Os dois primeiros itens referentes a velocidade do caminhão e drone não foram utilizados, pois tais valores foram baseados nos utilizados por (MURRAY; CHU, 2015).

[1] <https://colab.google/>
[2] <https://www.python.org/>

O número de nós é importante devido ao auxílio para abstrair os dados de um arquivo para uma estrutura de dados conveniente. Os dois últimos atributos foram de extrema importância, pois com eles foi possível realizar medições. Cada nó sendo representado em longitude e latitude, respectivamente, representando as coordenadas x e y como iriam ser tratadas.

4.3 Algoritmos Utilizados

Os algoritmos utilizados especificamente para este trabalho, que são para a pré-seleção de nós, foram dois: um algoritmo guloso e um aleatório. Importante ressaltar que esses algoritmos estão selecionando nós de uma solução TSP, retirando eles do caminho do caminhão e atribuindo ao drone.

4.3.1 Algoritmo Guloso

A seguir o pseudo-código 1, baseado no algoritmo guloso genérico de Rocha (ROCHA; DORINI, 2004), descreve como foi feita a seleção dos nós de forma gulosa.

Algoritmo 1: guloso_PDSTSP

```
Input: TSP, dist, vel_cam, vel_drone
Output: nos_sel
nos_sel ← {};
list_dist ← {};
aux ← 0.0;
pos ← 0;
tsp_copy ← copy(TSP);
for cada i em TSP do
    ele ← (dist[0][i], i);
    list_dist.append(ele);
list_dist.sort();
min_temp ← cam_vis_tsp(dist, TSP, vel_cam);
for cada i em list_dist do
    if i[1] ≠ 0 then
        pos ← tsp_copy.index(i[1]);
        tsp_copy.remove(i[1]);
        nos_sel.append(i[1]);
        aux ← tempo_gasto_PDSTSP(tsp_copy, nos_sel, dist, vel_drone,
         vel_cam);
        if aux ≤ min_temp then
            min_temp ← aux;
        else
            tsp_copy.insert(pos, i[1]);
            nos_sel.remove(i[1]);
return nos_sel;
```

Esse algoritmo visa achar os nós mais próximos do depósito de maneira a minimizar o tempo do caminhão inicialmente, e atribuindo os nós ao drone até que se tenha um balanceamento entre os veículos. A entrada dele são: um caminho do TSP, uma matriz de distâncias dos nós e a velocidades do caminhão e drone.

São iniciadas as variáveis: *nos_sel* uma lista para guardar os nós que serão selecionados pelo algoritmo; *list_dist* uma lista de tuplas contendo a distância daquele nó ao depósito e o índice dele; *aux* que irá guardar os valores retornados da função *tempo_gasto_PDSTSP*; *pos* que guarda a posição do índice de um valor da lista sequencial do TSP e *tsp_copy* que é a variável que sofrerá as mudanças referentes ao caminho do caminhão.

Um passo fundamental para um algoritmo guloso é a ordenação dos elementos a

serem manipulados, pois é priorizada as escolhas do mais próximo ao mais longínquo, portanto foi feita a ordenação dos nós mais próximos ao depósito. O primeiro laço de repetição é usado para preencher a variável *list_dist*, sendo o primeiro comando dele a criação de uma tupla com a distância do nó ao depósito e o índice do nó, em sequência a atribuição dela a lista *list_dist*. Após finalizado o laço de repetição é usado o comando *sort* do Python para ordenar o *list_dist*, nesse contexto a função ordenanda pelo primeiro elemento da tupla, ou seja, a distância dos nós ao depósito.

Em seguida, é definida e inicializada a variável *min_tempo* que é o tempo que o caminhão leva para finalizar o circuito do TSP. O próximo laço de repetição irá percorrer toda a *list_dist*, é inicialmente verificado se o elemento atual não é o depósito, pois não podemos retira-ló da rota do caminhão e a distância dele para si é 0, o que resultaria em nenhum impacto nos cálculos futuros.

É atribuído ao *pos* o índice do i-ésimo elemento do vetor *tsp_copy*, pois iremos removê-lo de tal vetor e insertá-lo em *nos_sel* para chamar a função $tempo\_gasto\_PDSTSP$, que será explicada futuramente. A variável *aux* recebe o retorno de $tempo\_gasto\_PDSTSP$, para ser comparado com o *min_tempo*, caso seja menor, o valor em *min_tempo* é atualizado. Caso contrário, o nó será reposicionado em *tsp_copy* na mesma posição de onde saiu e será removido de *nos_sel*. O algoritmo tem uma eficiência próxima a $O(n)$, considerando apenas a parte gulosa, por percorrer todos os n vértices do grafo.

De forma abrangente o algoritmo está recebendo uma solução TSP, e testando para cada um dos nós, se é vantajoso retirar ele do caminho do caminhão e atribuir ao drone, de forma a minimizar o *makespan*.

Algoritmo 2: tempo_gasto_PDSTSP

```
Input: TSP, drone, dist, vel_drone, vel_cam
Output: tempo_gasto
t_drone ← 0.0;
t_cam ← 0.0;
if length(drone) > 0 then
    t_drone ← drone_vis_sel(dist, drone, vel_drone);
t_cam ← cam_vis_tsp(dist, TSP, vel_cam);
if t_drone > t_cam then
    return t_drone;
else
    return t_cam;
```

O algoritmo 2 que refere ao cálculo do tempo gasto no PDSTSP tem as seguinte características. Recebe um circuito do TSP, os nós atendidos pelo drone, a matriz de distâncias dos nós e as velocidades do drone e caminhão. São iniciadas as variáveis *t_drone*

e *t_cam*, que são o tempo do drone realizar todas as entregas atribuídas a ele e o tempo do caminhão realizar seu percurso, respectivamente.

O primeiro condicional serve para avaliar se o tamanho do conjunto de clientes atendidos pelo drone é maior que zero, pois se maior é feito o cálculo do tempo gasto pelo drone para terminar a tarefa, caso contrário nada acontecerá. Em sequência é calculado o tempo do caminhão realizar as entregas e armazenado o resultado em *t_cam*. Por fim é retornando o maior valor entre *t_drone* e *t_cam*.

O uso conjunto da função $tempo\_gasto_PDSTSP$ com a $guloso\_PDSTSP$ é importante, pois o algoritmo 2 está calculando o tempo do último veículo retornar ao depósito, que em suma é a definição do *makespan*. Por sua vez o algoritmo $guloso\_PDSTSP$ 1, se beneficia desse calculo, devido a um balanceamento simples entre a distribuição de trabalho para o caminhão e drone. Como será mostrado na seção futura, não é vantajoso utilizar somente o drone para realizar todas as entregas, similar ao uso do caminhão.

4.3.2 Algoritmo Aleatório

Algoritmo 3: Aleatório_ER

Input: V (Número de vértices), A (Número de arestas)
Output: nos (Lista de nós selecionados)

```
prob ← A / (V - 1);
nos ← {};
for v ← 1 to V-1 do
    if random.random() < prob then
        nos.append(v);
return nos;
```

O algoritmo 3 se baseia no algoritmo de construção de grafos aleatórios de (ERDŐS; RÉNYI, 1959). O algoritmo 3 foi baseado na explicação feita em (FEOFILOFF, 2024), algumas alterações foram essenciais para atingir o objetivo desejado, que é escolher aleatoriamente arestas que ligam o depósito a outros nós. Primeiramente, foi retirada a necessidade de permutar por todos os vértices do grafo. Escolhendo aleatoriamente as arestas, que podem partir do nó 0, que é o depósito. Outra mudança significativa foi na probabilidade de a aresta ser selecionada, originalmente ela tomaria a chance de $A/(V * (V - 1))$, sendo A quantas arestas desejadas e V a quantidade de vértices do grafo. No estado atual, se tornou $A/(V - 1)$, que é um valor mais condizente com o que será abordado.

Sobre o funcionamento do algoritmo, ele recebe V que é o número de nós do grafo e A que é o número desejado de arestas. As variáveis *prop* e *nos* são inicializadas. O primeiro laço de repetição irá percorrer do primeiro nó até o último, e para cada um deles, terá

um condicional que avalia se um número gerado aleatoriamente entre 0 e 1 é menor que o valor de *prob* caso seja, a aresta entre o depósito e esse nó será feita. Por fim o algoritmo 3 tem uma eficiência de $0(v)$.

Dado o funcionamento do algoritmo 3 ele está selecionando aleatoriamente as arestas que irão conectar o depósito ao cliente que será atendido pelo drone.

4.4 Medidas de Avaliação

Para avaliar os resultados obtidos com os algoritmos de pré-seleção, as métricas utilizadas foram compostas no tempo de execução dos algoritmos, onde foi marcado o tempo de início e fim de execução em segundos, após registrado foi feita a subtração do tempo final e inicial. A outra métrica utilizada foi o valor do *makespan* que é o tempo do último veículo retornar ao depósito.

A intenção utilizando as medidas citadas é para uma comparação objetiva do desempenho dos algoritmos desenvolvidos no trabalho, assim obtendo gráficos que são fáceis de ler.

4.5 Configuração Experimental

Com o ambiente estabilizado, e os novos algoritmos devidamente preparados, os experimentos serão realizados.

Importante ressaltar que foi considerado o uso de um drone para realizar as entregas. Cada um dos algoritmos de pré-seleção foram executados e o mesmo conjunto de nós foram dados para eles como entradas. Importante ressaltar que foram doze conjuntos de nós testados. A importância de se manter uma mesma entrada para a pré-seleção vem justamente para que na comparação dos resultados se tenha um denominador em comum.

Lido o grafo, ele foi transformado em um objeto da classe nx para facilitar sua manipulação. Primeiramente é usada uma função da classe que retorna um caminho TSP do grafo, para que exista a seleção dos nós. Foi realizada a execução dos algoritmos, em que cada um retornou os vértices selecionados, que são os clientes que serão atendidos pelo drone, é representado como um vetor com os nós que serão atendidos pelo drone. Enquanto o restante dos nós não selecionados serão alocados para o caminhão atender.

Para se realizar uma boa análise, foi feita a composição dos gráficos, referentes ao tempo de pré-seleção que os algoritmos levaram para terminar e a relação do tempo de execução para cada um dos. Resultados esses que serão abordados na futura seção.

5 Análise e discussão dos resultados

Este Capítulo tem como objetivo analisar e discutir os resultados obtidos durante a etapa de experimentos.

Para cada um dos conjuntos de dados usados, foram gerados dois gráficos principais, o de tempo de execução da pré-seleção e um comparativo entre o valor do *makespan* entre o algoritmo guloso, aleatório e o de (MURRAY; CHU, 2015), além do tempo para o caso de apenas o drone realizar as entregas e do caminhão. Os grafos usados tinham vértices que variavam entre 10, 25, 250, 375 e 500.

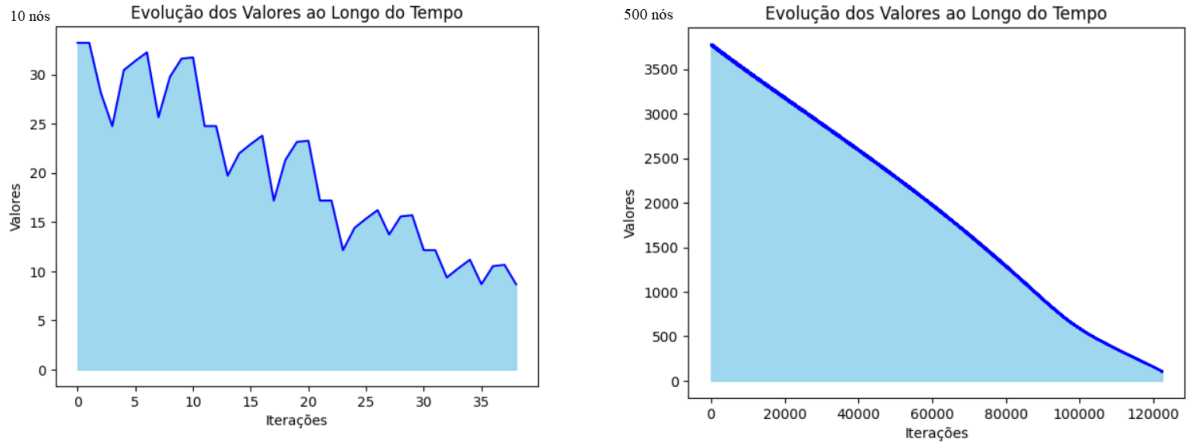
Um ponto que surgiu uma curiosidade foi em relação à evolução do valor do *makespan* relacionado ao algoritmo de (MURRAY; CHU, 2015), para descobrir como era o comportamento, foi realizado o registro do valor do *makespan* durante a execução do código, que resultou em um gráfico que informa uma minimização do valor. O eixo horizontal representa a quantidade de iterações que ocorreram durante a execução do algoritmo que solucionou o PDSTSP, o eixo vertical são os valores que o *makespan* atingiu por toda a solução. Na primeira vez, foi observado atuando em um grafo com 10 nós e tendo uma curva decrescente, porém, quando executado com um grafo com 500 vértices, se mostrou tendo uma queda quase linear em relação ao valor e número de iterações. Os dois gráficos podem ser observados na figura 4, sendo o de cima o histórico do grafo de 10 nós e o de baixo com 500.

Abordando o tempo que cada veículo conseguirá realizar suas tarefas, temos a presença de alguns gráficos de radar na Figura 5, os itens 5a, 5b, 5c e 5d, temos respectivamente os resultados do *makespan* para os grafos com 10, 25, 250 e 500 nós. Esses grafos foram selecionados devido ao tamanho que possuem, sendo os com 10 e 500 nós, o menor e maior grafos, respectivamente, enquanto os grafos com 25 e 250 vértices eram os que tinham o tamanho mais comum. O uso de um gráfico de radar é devido ao agrupamento de informações do mesmo tipo, que são o tempo de chegada do drone, caminhão e o *makespan* dos algoritmos.

Os valores presentes na figura 5 são $temp\_drn\_t$ que é o tempo necessário para o drone atender todos os clientes, sendo que ele parte do depósito, visita o alvo e retorna. O $temp\_cam\_t$ é o tempo que o caminhão demora para atender todos os clientes em um ciclo do TSP. Por sua vez, $temp\_al$, $temp\_gul$ e t_pdstsp_CM, são os *makespan*, respectivamente, do uso do algoritmo aleatório, guloso e da função PDSTSP de (MURRAY; CHU, 2015).

A forma mais usual de interpretar um gráfico de radar é a partir do centro para a borda, sendo os valores mais centrais, aqueles que são menores, enquanto os valores mais

Figura 4 – Comparação do Histórico do *makespan* da execução do algoritmo de (MURRAY; CHU, 2015).



(a) Histórico do *makespan* com 10 nós. **(b)** Histórico do *makespan* com 500 nós.

Legenda: O eixo horizontal representa quantas iterações o algoritmo fez, enquanto o eixo vertical são os valores do *makespan* durante a execução. Os dois gráficos apresentam um comportamento decrescente em relação ao tempo.

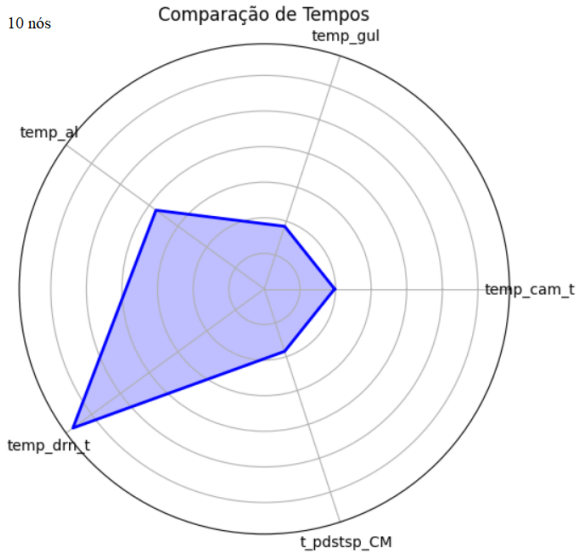
Fonte: Do Autor.

periféricos são maiores. Importante evidenciar que valores que estão no mesmo circulo ou região, tem valores próximos uns dos outros. Como podemos perceber na Figura 5, usar somente o drone como forma de entrega é altamente custoso em comparação com outras abordagens. Se deve ao fato de toda operação dele ter que ir e voltar, para assim seguir para a próxima entrega. Outra conclusão é que, mesmo sendo pouca diferença para os outros valores, principalmente para $temp\_gul$ e t_pdstsp_CM, o tempo do caminhão realizar todas as entregas não são ruins, isso se dá ao ponto que a otimização de soluções para o TSP são consolidadas e geram bons valores. Por fim os valores de $temp\_gul$ e t_pdstsp_CM são os melhores resultados.

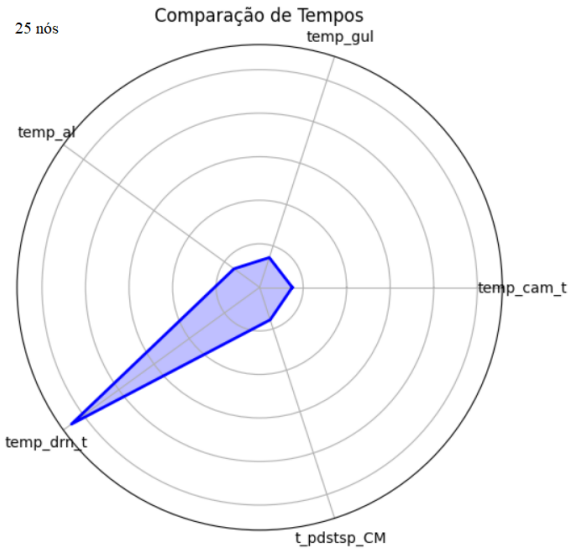
Sobre o *makespan* utilizando os algoritmos guloso e aleatório, temos a Figura 6 com os valores para cada uma das entradas. O eixo horizontal é a quantidade de nós que o grafo tinha, o eixo vertical o valor do *makespan*. Notasse uma pequena vantagem do *makespan* do algoritmo guloso em cima do aleatório. Para a padronização o algoritmo aleatório sempre recebeu como numero de arestas desejadas para seleção, um terço da quantidade de nós que o grafo tinha.

A Figura 7 representa o tempo que cada algoritmo de pré-seleção demorou para realizar sua chamada. Os eixos são iguais ao da Figura 6. O tempo de execução do algoritmo aleatório desempenhou um valor praticamente constante conforme foi evoluindo a quantidade de vértices do grafo. Enquanto o guloso foi aumentando o tempo necessário para retornar uma solução conforme aumentava o tamanho da entrada.

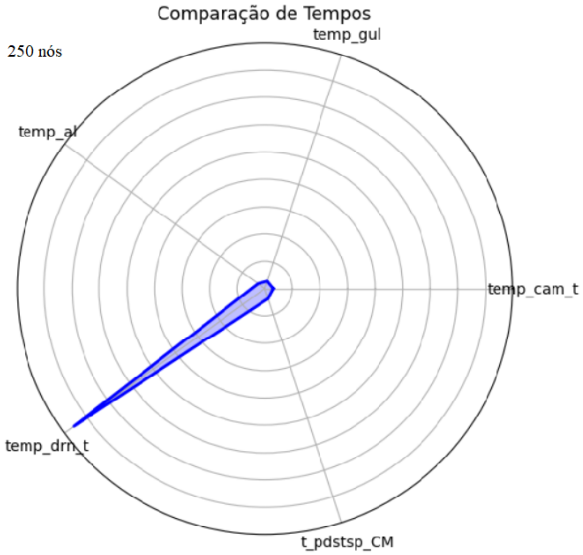
Figura 5 – Gráficos de radar para diferentes quantidades de nós.



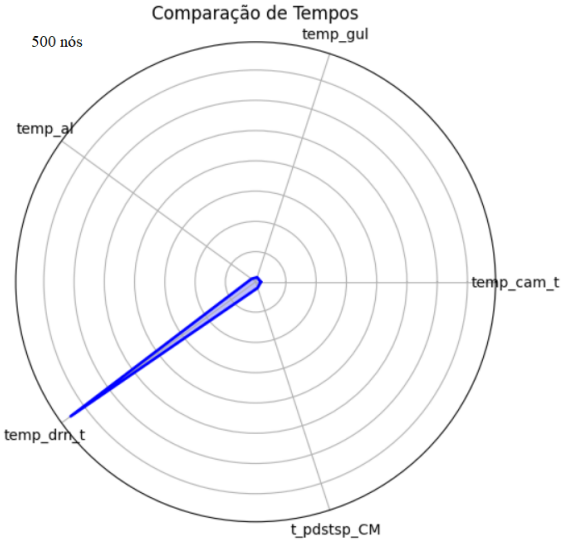
(a) Gráfico de radar para 10 nós.



(b) Gráfico de radar para 25 nós.



(c) Gráfico de radar para 250 nós.



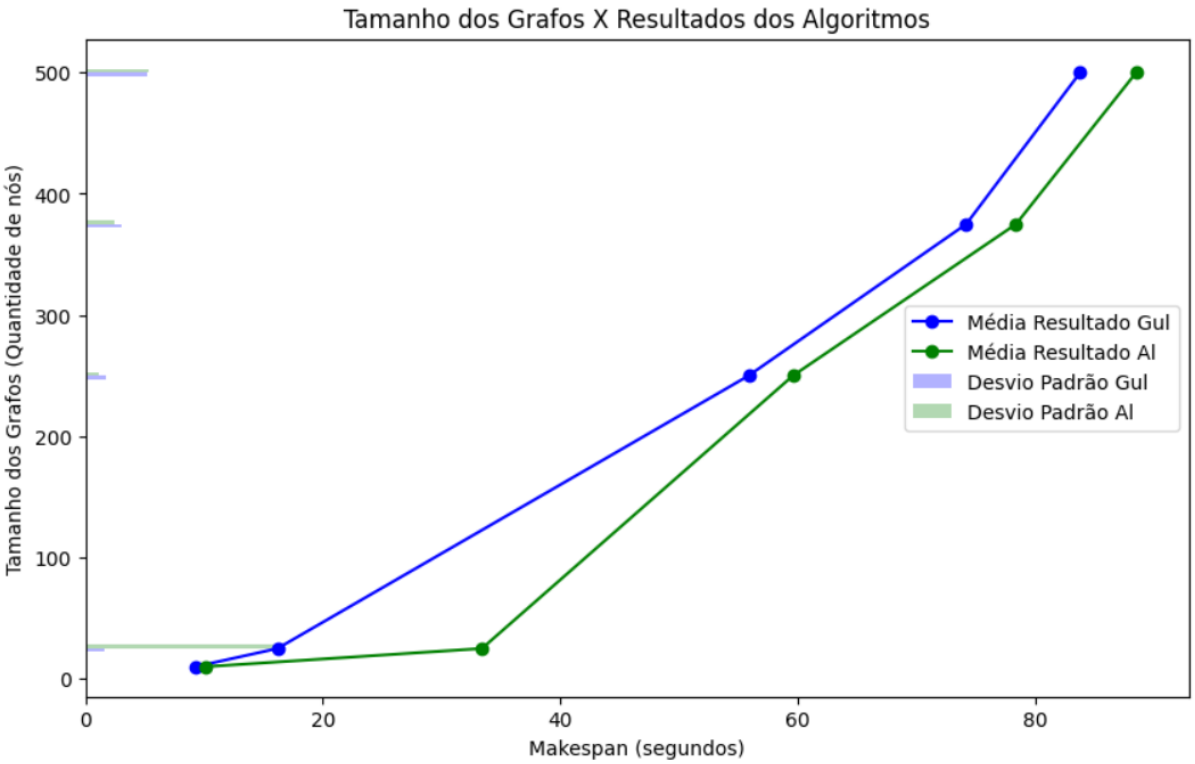
(d) Gráfico de radar para 500 nós.

Legenda: Os quatro gráficos contêm o tempo do drone e caminhão realizarem todas as entregas individualmente, o *makespan* resultante da pré-seleção do aleatório e guloso e por fim o *makespan* do algoritmo vindo de (MURRAY; CHU, 2015).

Fonte: Do Autor.

A Tabela 3 contém as informações gerais sobre cada um dos testes feitos e suas entradas, ao todo foram realizados 12 testes, sendo um deles um grafo com 10 nós, três com 25 e 250 nós, e dois experimentos com 375 e 500 vértices. A Tabela 3 contém as informações sobre o tempo de chamada e do *makespan* para cada um dos grafos dados

Figura 6 – *Makespan* dos algoritmos Aleatório e Guloso

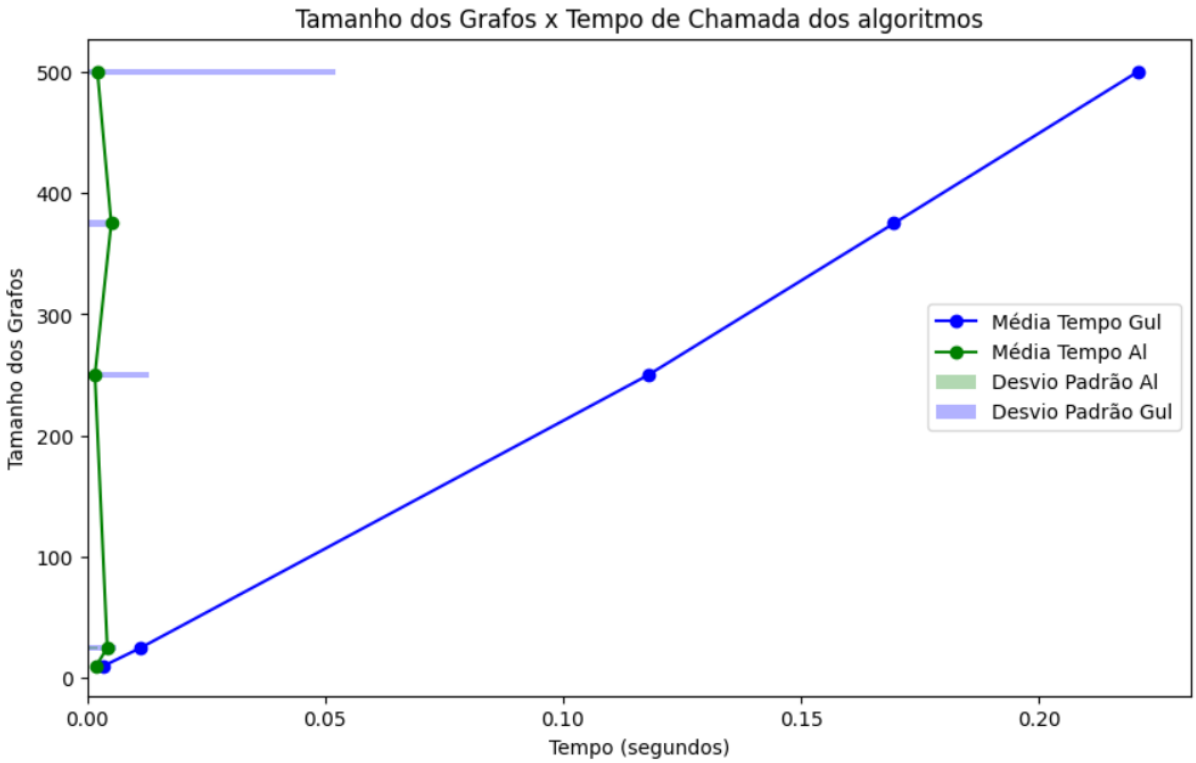


Legenda: O eixo horizontal representa o valor do *makespan* obteve em média para cada tamanho do grafo, enquanto o eixo vertical é o valor que o tamanho do grafo. As barras que se encontram no eixo vertical são o desvio padrão referente ao tamanho do grafo e o valor do *makespan*

Fonte: Do Autor.

como entrada.

Figura 7 – Tempo de Chamada das Funções



Legenda: O eixo horizontal representa o tempo de execução dos algoritmos de pré-seleção em média para cada tamanho do grafo, enquanto o eixo vertical é o valor que o tamanho do grafo. As barras que se encontram no eixo vertical são o desvio padrão referente ao tamanho do grafo e o tempo de execução

Fonte: Do Autor.

Tabela 3 – Resultados e tempos para diferentes arquivos

Nome Arquivo	**Quantidade de nós**	**temp_resul_gul**	**Tempo de chamada do gul**	**temp_resul_al**	**Tempo de chamada de al**
doublecenter-1-n10.txt	10	9.2488	0.00791	9.2873	0.00067
doublecenter-1-n25.txt	25	14.376	0.00607	27.1429	0.00158
doublecenter-1-n250.txt	250	56.8602	0.06707	525.6667	0.00035
doublecenter-2-n25.txt	25	17.1715	0.00385	39.2797	0.00136
doublecenter-2-n250.txt	250	53.6274	0.09411	717.7211	0.00260
doublecenter-3-n25.txt	25	16.9728	0.00167	24.2588	0.00024
doublecenter-3-n250.txt	250	55.3958	0.11982	519.0916	0.00171
doublecenter-3-n375.txt	375	76.3372	0.20090	736.1987	0.00227
doublecenter-4-n250.txt	250	57.5305	0.07462	494.8909	0.00034
doublecenter-4-n375.txt	375	72.0289	0.14371	1308.403	0.00212
doublecenter-8-n500.txt	500	80.0983	0.29763	1173.205	0.00223
doublecenter-9-n500.txt	500	87.4709	0.26957	1496.026	0.00284

6 Conclusão

Este trabalho aborda o PDSTSP e como a pré-seleção de nós contribui para uma solução do problema. Utilizaram-se dois modelos para a tarefa de selecionar os nós que serão atendidos pelo drone, sendo eles um guloso e outro aleatório.

É possível observar algumas evidências com os experimentos realizados. A primeira é que o algoritmo guloso mostra um comportamento crescente tanto no tempo de execução como no *makespan* encontrado, dado o tamanho da entrada dada. Por sua vez, o algoritmo aleatório, embora não chegue a resultados melhores que o guloso em relação ao *makespan*, seu desempenho no tempo de execução é contante, mesmo variando o tamanho da entrada.

Comparando os resultados envolvendo os algoritmos de pré-seleção com o de (MURRAY; CHU, 2015) podemos notar um desempenho próximo do algoritmo guloso para a solução usando MILP do artigo supracitado.

6.1 Limitações

Um aspecto que não foi considerado neste trabalho, é o custo de manutenção do drone e caminhão, que são operações que acontecem no mundo real e impactam de certa forma na atuação dos veículos.

Uma característica em comum aos problemas do TSP-D é utilizar as limitações físicas do drone como um fator de análise, por exemplo, a bateria do drone, carga suportada da encomenda, alcance de voou e condições climáticas. Este trabalho focou exclusivamente na pré-seleção dos nós, e abdicou de considerar os aspectos físicos do drone como um agente importante para a resolução do PDSTSP. Portando os experimentos feitos não contemplam a totalidade do uso de drones.

6.2 Trabalhos Futuros

O tema abordado neste trabalho, envolvendo comparações entre algoritmos de pré-seleção de nós, sendo utilizado como opção para roteamento de veículos e pode ser melhorado em trabalhos seguintes, como listado a seguir.

- Realizar a construção de novos algoritmos para a pré-seleção dos nós. Exemplo: unir as duas abordagens feitas no trabalho;
- Implementar as outras versões do problema do TSP-D encontradas na literatura;

- Deixar mais robusto a pré-seleção, em que seja possível levar características do drone, bateria e carga suportada.

Referências

AGATZ, N.; BOUMAN, P.; SCHMIDT, M. Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, INFORMS, v. 52, n. 4, p. 965–981, 2018. Citado na página 27.

BOUMAN, P.; AGATZ, N.; SCHMIDT, M. Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, Wiley Online Library, v. 72, n. 4, p. 528–542, 2018. Citado na página 27.

CAVANI, S.; IORI, M.; ROBERTI, R. Exact methods for the traveling salesman problem with multiple drones. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 130, p. 103280, 2021. Citado na página 27.

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior. *Portal de Periódicos da CAPES*. 2024. Acesso em: 23 jul. 2024. Disponível em: <https://www.periodicos.capes.gov.br/>. Citado na página 25.

CORMEN, T. H. et al. *Algoritmos: Teoria e Prática*. 3. ed. Rio de Janeiro: Elsevier, 2012. Tradução de: Introduction to Algorithms, 3rd ed. Citado na página 23.

DANTZIG, G. B.; FULKERSON, D. R.; JOHNSON, S. M. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, INFORMS, v. 2, n. 4, p. 393–410, 1954. Citado na página 22.

DARWIN, C. *A origem das espécies*. [S.l.]: Lelo & Irmão, 2009. Citado na página 27.

DELL'AMICO, M.; MONTEMANNI, R.; NOVELLANI, S. Matheuristic algorithms for the parallel drone scheduling traveling salesman problem. *Annals of Operations Research*, Springer, v. 289, p. 211–226, 2020. Citado na página 27.

ERDŐS, P.; RÉNYI, A. On random graphs i. *Publicationes Mathematicae Debrecen*, v. 6, p. 290–297, 1959. Citado 2 vezes nas páginas 21 e 35.

FEOFILOFF, P. *Algoritmos para Grafos - Algoritmos Aleatórios*. 2024. Acesso em: 23 jul. 2024. Disponível em: <https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/random.html>. Citado na página 35.

Google. *Google Acadêmico*. 2024. Acesso em: 23 jul. 2024. Disponível em: <https://scholar.google.com.br/>. Citado na página 25.

HA, Q. M. et al. On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 86, p. 597–621, 2018. Citado na página 27.

KIM, S.; MOON, I. Traveling salesman problem with a drone station. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, IEEE, v. 49, n. 1, p. 42–52, 2018. Citado na página 27.

KITJACHAROENCHAI, P. et al. Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. *Computers & Industrial Engineering*, Elsevier, v. 129, p. 14–30, 2019. Citado na página 27.

KLOSTER, K. et al. The multiple traveling salesman problem in presence of drone-and robot-supported packet stations. *European Journal of Operational Research*, Elsevier, v. 305, n. 2, p. 630–643, 2023. Citado 2 vezes nas páginas 27 e 28.

LOPES, M. $problemas_mil.2017.Disponívelem : <>$. Citado na página 20.

LU, S.-H. et al. Improving the efficiency of last-mile delivery with the flexible drones traveling salesman problem. *Expert Systems with Applications*, Elsevier, v. 209, p. 118351, 2022. Citado na página 27.

MAJEED, A.; RAUF, I. Graph theory: A comprehensive survey about graph theory applications in computer science and social networks. *Inventions*, v. 5, n. 1, 2020. ISSN 2411-5134. Disponível em: <https://www.mdpi.com/2411-5134/5/1/10>. Citado na página 20.

MONTEMANNI, R.; DELL'AMICO, M.; CORSINI, A. Parallel drone scheduling vehicle routing problems with collective drones. *Computers & Operations Research*, Elsevier, v. 163, p. 106514, 2024. Citado na página 27.

MORANDI, N. et al. The tsp with drones: The benefits of retraversing the arcs. *arXiv preprint arXiv:2207.03441*, 2022. Disponível em: <https://arxiv.org/abs/2207.03441>. Citado na página 27.

MURRAY, C. C.; CHU, A. G. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 54, p. 86–109, 2015. Citado 10 vezes nas páginas 13, 17, 21, 26, 27, 31, 37, 38, 39 e 43.

NGUYEN, M. A.; HÀ, M. H. The parallel drone scheduling traveling salesman problem with collective drones. *Transportation Science*, INFORMS, v. 57, n. 4, p. 866–888, 2023. Citado na página 27.

PARDINI, D. *O Problema do Caixeiro Viajante*. 2015. Disponível em: <https://otimizacaonapratica.wordpress.com/2015/11/09/o-problema-do-caixeiro-viajante/>. Citado 2 vezes nas páginas 13 e 22.

POIKONEN, S.; GOLDEN, B.; WASIL, E. A. A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, INFORMS, v. 31, n. 2, p. 335–346, 2019. Citado na página 27.

ROCHA, A.; DORINI, L. B. Algoritmos gulosos: definições e aplicações. *Relatório Técnico, Instituto de Computação, Universidade Estadual de Campinas*, Campinas, SP, v. 53, 2004.

Disponível em: <http://www.ic.unicamp.br/~relatorio/2004/report_53.pdf>. Citado na página 32.

SCUDERO, E. *Linguagens de Alto Nível vs. Baixo Nível: qual é melhor?* 2017. Disponível em: <https://becode.com.br/linguagens-alto-nivel-x-baixo-nivel/>. Citado na página 31.

VÁSQUEZ, S. A.; ANGULO, G.; KLAPP, M. A. An exact solution method for the tsp with drone based on decomposition. *Computers & Operations Research*, Elsevier, v. 127, p. 105127, 2021. Citado na página 27.