

Combinando Técnicas de *Machine Learning* e Otimização para Roteamento Combinado de Veículos Tripulados e Não Tripulados

Guilherme Manso Haddad

Instituto de Computação - Universidade Federal Fluminense

Niterói, RJ – Brasil

gmhaddad@ic.uff.br

Luiz Satoru Ochi

Instituto de Computação - Universidade Federal Fluminense

Niterói, RJ – Brasil

satoru@ic.uff.br

RESUMO

A crescente evolução das tecnologias de *drones* permite que novas abordagens para a entregas de mercadoria sejam realizadas, otimizando recursos, reduzindo emissões e contribuindo no desenvolvimento de cidades inteligentes. Tais tecnologias, como a distribuição de mercadorias utilizando uma frota híbrida caminhão-*drone* depende da otimização do roteiro de entregas para que seu uso seja significativamente relevante. Neste trabalho, propomos o uso de um algoritmo de aprendizado por reforço, conhecido na literatura como Otimização por Colônia de Formigas (Ant Colony Optimization-ACO), que tem resolvido diferentes problemas de roteamento e *scheduling* de forma eficiente. O algoritmo proposto é testado em instâncias baseadas na TSPLib, apresentando bons resultados quando comparado com o melhor valor conhecido, contribuindo para a utilização desta tecnologia nas cidades inteligentes.

PALAVRAS CHAVE. Cidades Inteligentes, Roteamento, ACO.

Tópicos: Cidades Inteligentes, Logística e Transportes, Metaheurísticas

ABSTRACT

The increasing evolution of drone technologies allows new approaches to merchandise parcel delivery, optimizing resources, reducing emissions, and contributing to the development of smart cities. These technologies, such as the distribution of goods using a hybrid fleet: the truck-drone system, depend on optimizing delivery routes to make their use significantly relevant. In this work, we propose the use of a reinforcement learning algorithm, known in the literature as Ant Colony Optimization (ACO), that has solved different routing and scheduling problems efficiently. The proposed algorithm was tested on instances based on TSPLib and presented good results when compared to the best-known solution, contributing to the use of this technology in smart cities.

KEYWORDS. Smart Cities, Routing, ACO.

Paper topics: Smart Cities, Logistics and Transport, Metaheuristics

1. Introdução

A movimentação de mercadorias é um dos setores mais cruciais da economia global, exigindo alta eficiência e precisão para garantir que as mercadorias cheguem ao destino final em perfeitas condições e dentro do prazo previsto. A entrega de produtos é um processo complexo, mas as tecnologias presentes nos segmentos de cidades inteligentes estão revolucionando a forma como a logística é gerenciada, com tecnologia sendo usada para otimizar a infraestrutura urbana e fornecer serviços mais eficientes e sustentáveis para a população.

O uso de veículos com energia renovável, e em particular, o uso de *drones* tem se mostrado uma alternativa promissora, conforme destacado em [Korczak e Kijewska, 2019]. *Drones* são pequenos veículos aéreos não tripulados que podem ser programados para transportar cargas leves, como documentos, medicamentos, órgãos para transplantes, alimentos e outros produtos de pequeno porte, por toda a cidade. Eles voam em linha reta, evitando congestionamentos e reduzindo o tempo de entrega. Além disso, a utilização de energia elétrica torna-os uma opção mais sustentável do que carros e caminhões convencionais movidos a combustão, emitindo menos gases de efeito estufa.

Embora o uso de *drones* para a entrega de mercadorias seja uma solução em destaque, ainda existem desafios a serem superados. Por exemplo, o alcance e a capacidade de carga limitada tornam difícil sua utilização exclusiva para distribuição de mercadorias. Porém, a combinação de *drones* e caminhões surge como uma solução viável e promissora para reduzir essas limitações e permitir a entrega de mercadorias com maior eficiência, principalmente em grandes centros urbanos.

Ao combinar o uso de *drone* com um caminhão, as empresas podem criar uma rede de entrega mais rápida e eficiente, reduzindo o tempo de entrega e custos operacionais, além de aumentar a satisfação do cliente. O *drone* pode também ser uma das poucas alternativas eficientes para entregas em áreas de difícil acesso ou de congestionamento de tráfego, enquanto o caminhão pode lidar com entregas em áreas mais amplas. A combinação dos dois modos de transporte pode reduzir os custos de entrega e melhorar a rentabilidade da empresa.

O roteamento deste par caminhão-*drone* é um problema de natureza complexa, se mostrando como um desafio a ser resolvido para que o uso desta tecnologia seja eficiente. Desta forma, este trabalho propõe um novo algoritmo, baseado em otimização por colônia de formigas, para resolver este desafio. Inicialmente, os trabalhos relacionados são apresentados na seção 2. Após, na seção 3, o problema é apresentado e o algoritmo proposto é detalhado na seção 4. Por fim, os resultados são exibidos na seção 5 e as conclusões e trabalhos futuros são descritos na seção 6.

2. Trabalhos Relacionados

Há diversas publicações que tratam do uso de *drones* para entregas de mercadorias. Em um desses estudos, descrito em [Wang e Sheu, 2019], os autores solucionaram o problema de roteamento de veículos considerando somente o uso de *drones* para as entregas, onde conseguiram bons resultados utilizando um algoritmo misto de programação linear inteira e heurística. Em outros estudos, como em [Othman et al., 2018] e [Mavrovouniotis et al., 2018], os autores utilizaram o algoritmo de otimização por colônia de formigas (ACO) para resolver o problema de roteamento de veículos (VRP), onde o último considerou o uso de veículos elétricos. Em [Narayanan et al., 2022], foi empregado um método de aprendizado por reforço para resolver o VRP com veículos elétricos. Num artigo mais recente, [Marques Junior et al., 2023] propõem o uso de uma frota de *drones* para entregas de mercadorias utilizando um modelo multi-objetivo.

A definição do problema a ser tratado é essencial, pois permite que as formulações e restrições, necessárias para o desenvolvimento do algoritmo, sejam estabelecidas. O *drone* pode ser utilizado em conjunto ou separadamente do caminhão. Em [Mbiadou Saleu et al., 2022], por

exemplo, o *drone* é utilizado de forma independente do caminhão, sem interação direta entre eles. O *drone* realiza suas entregas enquanto o caminhão segue o seu roteiro, simultaneamente. Esse problema é chamado de *The Parallel Drone Scheduling TSP* (PDTSP) em [Murray e Chu, 2015]. Nesse mesmo trabalho, os autores apresentam o *The Flying Sidekick Traveler Salesman Problem* (FSTSP), no qual o *drone* trabalha em conjunto com o caminhão, realizando as entregas e utilizando o caminhão como ponto de apoio.

É importante destacar que o problema TSP-D, definido em [Agatz et al., 2018], é semelhante ao FSTSP, mas há algumas diferenças, como a possibilidade do caminhão permanecer estacionado em um local enquanto o *drone* realiza as entregas. No trabalho de [Murray e Chu, 2015], o autor resolve o problema FSTSP para pequenas instâncias por meio da programação linear, mas recorre a recursos de heurísticas não determinísticas para resolver instâncias maiores.

Diversas abordagens têm sido utilizadas para solucionar o problema de Roteamento de Veículos com Drone (FSTSP). Em [Ponza, 2016], é usado o algoritmo *Simulated Annealing Metaheuristic* para mostrar que o uso de *drones* para entregas pode ser eficiente em áreas urbanas, especialmente para encomendas de pequeno e médio porte. O modelo de otimização proposto foi capaz de reduzir significativamente o tempo de entrega em comparação com outros métodos convencionais. No entanto, o autor destaca que há limitações, como a necessidade de desenvolver tecnologias que permitam *drones* voarem de forma autônoma e segura em áreas urbanas e a necessidade de regulamentação por parte das autoridades aeronáuticas para garantir a segurança e privacidade das pessoas envolvidas no processo de entrega.

A abordagem apresentada em [Crişan e Nechita, 2019] parte da solução inicial do Caixeiro Viajante obtida pelo método Concorde e realiza uma heurística gulosa para associar os vértices ao *drone*. Embora tenha obtido bons resultados em termos de qualidade de valores e tempo de processamento, o algoritmo foi testado em apenas duas instâncias. Em [Freitas e Penna, 2018], os autores apresentaram uma heurística baseada em busca pela vizinhança, juntamente com novas instâncias baseadas na conhecida TSPLib, contribuindo com referências para o estudo do problema.

Há uma grande quantidade de soluções propostas para o roteamento de veículos ou *drones*. Em [Chung et al., 2020], os autores apresentam o estado da arte dos trabalhos de otimização para operações com *drones* e *drones* em conjunto com caminhões, concluindo que são necessários bons métodos de otimização para um melhor uso dos *drones* e que o uso combinado de *drones* com caminhões ainda está em fase inicial, necessitando do desenvolvimento de boas heurísticas, visto que a programação inteira demandaria um grande esforço computacional, dada a natureza do problema.

3. Descrição do Problema

Este trabalho propõe um método para resolver o problema do roteamento de entregas realizadas por um *drone* atuando em conjunto com um caminhão, denominado *The Flying Sidekick Traveler Salesman Problem* (FSTSP) e apresentado em [Murray e Chu, 2015].

O problema FSTSP, tido como um problema NP-Difícil, define o roteamento conjunto do caminhão e do *drone* para realizar a entrega aos clientes localizados nos vértices. Neste problema, o *drone* pode realizar entrega a clientes em que a carga a ser transportada está dentro de sua capacidade e o tempo de voo não ultrapasse sua autonomia. Além disso, o *drone* atua em conjunto com o caminhão, pois pode realizar o trecho caminhão-cliente-caminhão ao realizar uma entrega. Uma vez no caminhão, o *drone* pode ser armazenado ou carregado e direcionado para realizar outra entrega. Assim, o caminhão é direcionado aos seus clientes, realizando seu roteiro de entrega, enquanto o *drone* realiza as entregas em missões, definida em três vértices, a saber:

- Local de partida: vértice que o *drone* decola a partir do caminhão.

- Local de entrega: vértice (cliente) a ser atendido pelo *drone*.
- Local de retorno: vértice em que o *drone* retorna ao caminhão.

Importante destacar que para decolar e ser recebido pelo caminhão, o *drone* necessita que o motorista, presente no caminhão, faça adequações, assim, o tempo de preparo e tempo de recebimento são variáveis importantes. O tempo de serviço, ou de entrega, é contabilizado no momento da entrega realizada pelo *drone* ou pelo caminhão.

A autonomia do *drone* deve ser respeitada para o trecho considerado da missão do *drone*, assim, o tempo total de voo não deve ultrapassar a capacidade da bateria do *drone*. O *drone* decola e retorna apenas em clientes que foram atendidos pelo caminhão. O caminhão pode esperar o *drone* no local definido de encontro ou vice-versa, este tempo de espera deve ser considerado na autonomia necessária do *drone*, pois é assumido que o *drone* deve aguardar o caminhão em voo.

O FSTSP é um problema de minimização do tempo total necessário para o conjunto caminhão-*drone*, partindo do depósito, realizar todas as entregas, ou seja, visitar uma única vez todos os vértices do problema e retornar ao vértice inicial.

4. Descrição do Algoritmo Proposto

A proposta deste trabalho é utilizar um algoritmo de aprendizado de máquina que tem se mostrado muito eficiente na solução de diferentes problemas de Otimização Combinatória. Mais especificamente, o algoritmo proposto para resolver o FSTSP é baseado no método de aprendizado por reforço, conhecido na literatura como Algoritmo de Colônia de Formigas, ou *Ant Colony Optimization* (ACO), apresentado em [Dorigo, 1992]. Este método foi escolhido por mostrar boa capacidade de encontrar excelentes soluções, como exposto em [Bell e McMullen, 2004] que utilizou o ACO para resolver o problema de roteamento de veículos; Boa flexibilidade, podendo ser adaptado para incorporar variações, como veículos heterogêneos ([Zhang et al., 2019]) e entregas e retiradas ([Çatay, 2009]); Boa capacidade de incluir restrições, como limitações de capacidade e duração do tempo de viagem, restrições pertinentes quando se trata de veículos elétricos, como em [Mavrovouniotis et al., 2018]. Além disso, o ACO é uma técnica facilmente implementada em computação paralela, podendo acelerar o processo de otimização, [Mavrovouniotis et al., 2019]. Por fim, podemos destacar que o ACO consegue entregar bons resultados para situações complexas e próximas ao real, como mostra [Rizzoli et al., 2007], onde o autor utilizou este método para resolver o VRP em um ambiente urbano, incluindo situações do mundo real como entrega e coleta e janela temporais, destacando que o ACO se tornou uma ferramenta importante neste tema.

O pseudo-código exposto no Algoritmo 1 mostra a aplicação do ACO para resolver o FSTSP. Neste algoritmo, a cada iteração, o conjunto de formiga é liberado. Este conjunto é composto por duas formigas, a formiga tipo A, que representa o caminhão e a formiga tipo B, representando o *drone*.

O algoritmo inicia com a contabilização dos custos e atratividades das formigas do tipo A e do tipo B. O custo é definido pelo tempo de traslado entre dois vértices. As distâncias para o *drone* são calculadas pela métrica euclidiana, já as distâncias para o trânsito do caminhão, são calculadas pela métrica de *Manhattan*, visto que o caminhão deve seguir as rodovias e trajetos definidos que na maioria das aplicações não são em linhas retas. A atratividade, ou informação heurística, é calculada pelo inverso do custo e definida de forma independente para cada tipo de formiga.

A inicialização de ferormônio utiliza o método do vizinho mais próximo, ou seja, o roteiro de solução é escolhido utilizando a regra de selecionar sempre o ponto mais próximo a partir da saída do depósito. O ferormônio de cada tipo de formiga é tratado de forma independente, ou seja,

Algoritmo 1 ACO para FSTSP

```

1: Inicializa Custos e Atratividades
2: Inicializa Níveis de Ferormônio
3: while critério de parada não for alcançado do
4:   for all ConjuntoFormiga do
5:     Inicia Lista de Locais Não Visitados
6:     while Há locais não visitados do
7:       EscolheDestinoFormigaA
8:       if a formiga B está no caminho then
9:         EscolheDestinoFormigaB
10:      else
11:        SeleccionaLocalRetornoFormigaB
12:      end if
13:      RealizaMovimentações (atualiza roteiros)
14:      Atualiza lista de clientes não visitados
15:      Contabiliza tempos gastos
16:    end while
17:  end for
18:  RealizaBuscaLocal
19:  AtualizaNiveisFerormônio
20:  Atualiza melhor solução
21: end while

```

o depósito e o seu peso na tomada de decisão são exclusivos para cada tipo de formiga. A linha 04 indica que a cada iteração todos os conjuntos de formigas são, um a um, liberados. A cada liberação os pontos são, um a um, escolhidos para serem visitados até que todos os vértices sejam visitados (linha 06 a 16). A escolha do ponto a ser visitado pela formiga tipo A é realizada na linha 07, conforme a distribuição de probabilidades definida na Equação 1.

$$\rho_{ij}^a = \frac{(\tau_{ij}^a)^\alpha (\eta_{ij}^a)^\beta}{\sum_{l \in N_i^a} (\tau_{il}^a)^\alpha (\eta_{il}^a)^\beta}, j \in N_i^a \quad (1)$$

Onde ρ_{ij}^a representa a probabilidade da formiga A escolher ir para j partindo de i , sendo proporcional ao nível de ferormônio τ_{ij}^a e a atratividade η_{ij}^a no trecho ij . N_i^a é o conjunto de possíveis e viáveis destinos da formiga A saindo de i . Ou seja, a probabilidade ρ_{ij}^a é nula para j pertencente a destinos inviáveis (como locais já visitados, por exemplo). Assim, o algoritmo sorteia, dentre os pontos possíveis, o destino escolhido da formiga tipo A, aqui denominado eA . Os valores de α e β representam, respectivamente, a importância do nível de ferormônio e da atratividade, devendo ser ajustados corretamente para um bom resultado do modelo.

O destino da formiga do tipo B depende do seu local durante a iteração, conforme definido na linha 08 e na sua negação na linha 10. Assim, caso a formiga esteja no caminho, antes que a formiga A se movimente para o ponto eA , o algoritmo escolhe o destino da formiga tipo B através da Equação 2.

$$\rho_{ij}^b = \frac{(\tau_{ij}^b)^\alpha (\eta_{ij}^b)^\beta}{\sum_{l \in N_i^b} (\tau_{il}^b)^\alpha (\eta_{il}^b)^\beta}, j \in N_i^b \quad (2)$$

O conjunto N_i^b contém apenas os clientes não visitados cuja carga pode ser entregue por *drone* e os clientes em que o *drone* tem autonomia suficiente para visitar, partindo de i e retornando a eA . O conjunto N_i^b também contém o ponto anteriormente escolhido pela formiga A, pois, caso a formiga B escolha este mesmo ponto, não haverá decolagem do *drone*, ou seja, o *drone* permanecerá no caminhão até o próximo ponto.

Importante ressaltar que o *drone* deve aguardar o caminhão caso chegue em eA antes do caminhão, devendo este tempo ser considerado na sua autonomia necessária. Ou seja, o tempo da formiga A de i até eA deve ser menor ou igual ao autonomia do *drone* para que haja possibilidade de decolagem do *drone* do ponto i , pois o *drone* deve esperar o caminhão utilizando suas baterias. Caso contrário, a formiga B terá apenas a opção de continuar no caminhão. O ponto escolhido para ser visitado pelo *drone* é chamado de eB e a previsão de retorno (local de encontro com o caminhão) é denominado rB , inicialmente igual a eA .

Se a formiga B estiver realizando uma entrega, ou seja, estiver fora do caminhão, o algoritmo realiza a escolha do ponto de encontro da formiga B com o caminhão. Nesta verificação, o algoritmo analisa qual opção tem menor tempo total, conforme descrito na Figura 1. Neste exemplo, durante a escolha do eB , passo 1, as formigas A e B estão no vértice 1 e o algoritmo decide que a formiga A segue para o vértice 2 e a formiga B segue para o vértice 3 com previsão de retorno (ponto rB) o vértice 2.

No momento de escolha do retorno do *drone* ao caminhão, passo 2, o algoritmo considera o novo destino da formiga A, ou seja, o vértice 4 (novo eA). O algoritmo então avalia o melhor local de retorno da formiga B, comparando os tempos gastos nas duas escolhas possíveis: $op1$ e $op2$, ou seja, o algoritmo verifica se o *drone* encontra o caminhão no ponto rB , seguindo juntos ao próximo destino (opção $op1$) ou se o *drone* encontra o caminhão em seu novo destino, realizando o trecho $op2$.

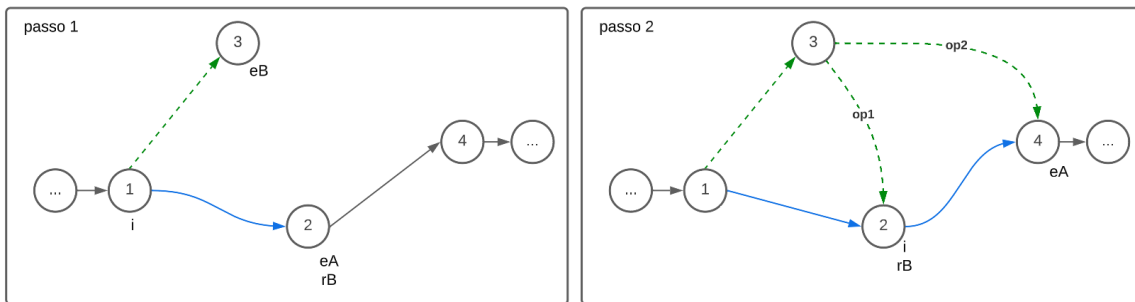


Figura 1: Escolha do retorno do *drone*. Linha contínua: caminhão. linha pontilhada: *drone*

Após as escolhas, o algoritmo realiza as movimentações (linha 13), atualiza os pontos não visitados (linha 14) e contabiliza os tempos gastos pelas formigas (linha 15). O conjunto de formigas deve visitar todos os pontos da instância para que o próximo conjunto seja liberado a visitar todos os pontos (linha 16). Após todos os conjuntos de formigas definirem os seus roteiros (bloco entre linhas 04 e 17), os níveis de ferormônio são atualizados, conforme Equações 3 e 4.

$$\tau_{ij}^a \leftarrow (1 - \rho)\tau_{ij}^a + \sum_{k=1}^K \delta \Delta \tau_{ij}^{a,k} + w \Delta \tau_{ij}^{a,gb} \quad (3)$$

$$\tau_{ij}^b \leftarrow (1 - \rho)\tau_{i,j}^b + \sum_{k=1}^K \delta \Delta \tau_{ij}^{b,k} + w \Delta \tau_{ij}^{b,gb} \quad (4)$$

$$\Delta \tau_{ij}^{a,k} = \frac{Q}{T_k} \quad , \quad \Delta \tau_{ij}^{b,k} = \frac{Q}{T_k} \quad (5)$$

Onde Q é uma constante e T_k é o valor da solução encontrada no conjunto de formigas k . O índice gb indica o conjunto de formiga que apresentou o melhor resultado na rodada.

O lado direito das Equações 3 e 4 indica, em seu primeiro termo, o decaimento do ferormônio na taxa definida ρ . O segundo termo mostra a atualização do ferormônio por todas as formigas, aonde δ é o fator que indica o peso dos ferormônio depositado, quando este peso é zero, apenas a melhor formiga deposita ferormônio. O terceiro termo mostra a parcela de ferormônio depositada pelo melhor conjunto de formigas da rodada considerada, aonde o fator w indica o peso desta parcela.

Na utilização do ACO, o uso de Busca Local durante a execução do algoritmo pode auxiliar na procura por melhores soluções, otimizando as rotas já encontradas, ao verificar novas possibilidades de locais de retorno para o *drone*, reduzindo as limitações assumidas pelo algoritmo (exibidas na Figura 1). A Busca Local desenvolvida é simplificada com o propósito de não sobrecarregar o algoritmo, visto que uma busca local com uma extensa procura na vizinhança poderia impactar negativamente o tempo de processamento.

O procedimento da Busca Local percorre o roteiro realizado pela dupla caminhão e *drone*, realizando verificações de melhoria, conforme detalhado a seguir, as ilustrações utilizam linha sólida preta para representar o trajeto realizado pelo conjunto caminhão-*drone*, linha sólida azul representa o trajeto realizado apenas pelo caminhão e a linha pontilhada verde represente o trajeto realizado apenas pelo *drone*.

Em trechos em que o *drone* segue embarcado no caminhão, o algoritmo de Busca Local verifica se a alternância entre o destino e o próximo ponto da rota resulta em melhoria, conforme representado na Figura 2, onde o tempo necessário para o conjunto caminhão-*drone* percorrer o trajeto $ABCD$ é comparado com o tempo para o trajeto $ACBD$. Destaca-se a importante restrição de todos os subtrechos considerados na verificação acima serem realizados pelo conjunto *drone*-caminhão, ou seja, não há decolagem do *drone* nos vértices considerados.

A Busca Local verifica novos pontos de saída e retorno do *drone* conforme o tipo do trajeto realizado originalmente pelo *drone*, trajeto triangular ou quadrangular, conforme Figuras 3 e 4 respectivamente. Estes trajetos são definidos nas opções *op1* e *op2* no momento da escolha do retorno do *drone* ao caminhão.

O algoritmo de Busca Local, ao reconhecer a viagem triangular do *drone*, verifica se a saída do *drone* no vértice seguinte resulta em redução de tempo, respeitando as restrições existentes, como a autonomia do *drone*. Assim, a viagem do *drone*, representada na sequência ADB da Figura 3, junto com a ida do caminhão no percurso AB é confrontada com a opção representada na Figura 5, ou seja, o *drone* realizando a viagem BDC e o caminhão o trecho BC durante o voo do *drone*.

O algoritmo de Busca Local, ao reconhecer a viagem do *drone* do tipo quadrangular, representada pelos trajetos ABC e ADC , respectivamente, do caminhão e do *drone* na Figura 4, realiza as verificações representadas nas Figuras 6, 7 e 8 buscando uma redução no tempo de viagem total.

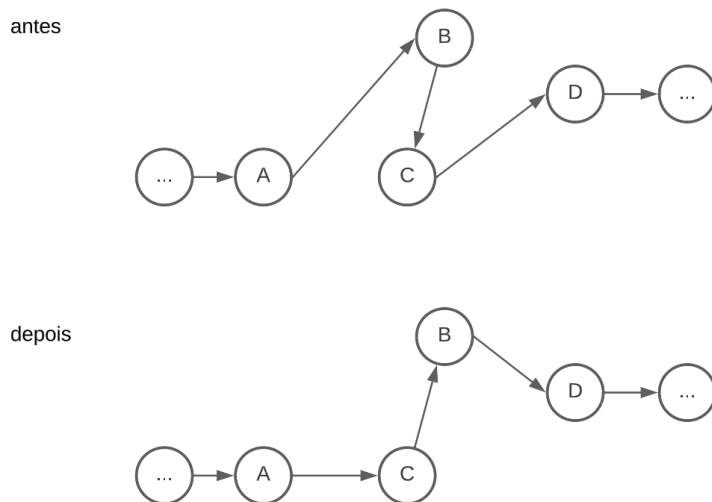


Figura 2: Trajeto de entrega por caminhão. (a) antes: opção original (b) depois: opção de teste

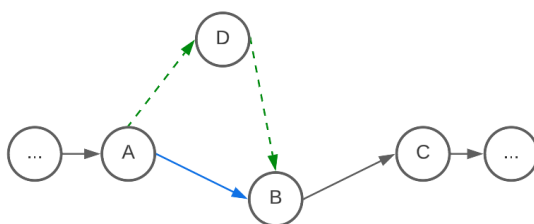


Figura 3: Retorno triangular.

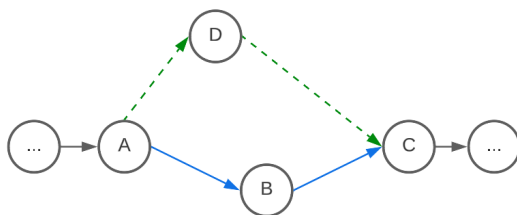


Figura 4: Retorno quadrangular.

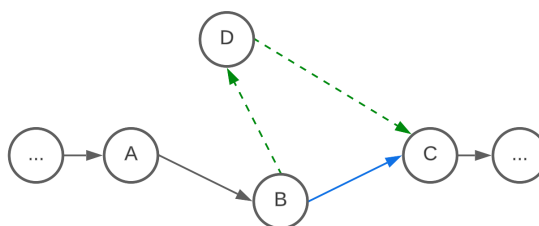


Figura 5: Opção testada para a viagem triangular.

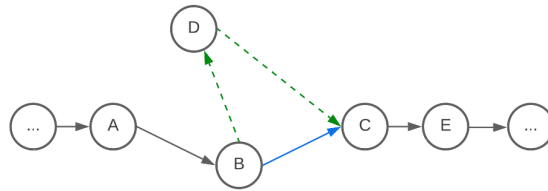


Figura 6: Opção 1 de teste para a viagem quadrangular.

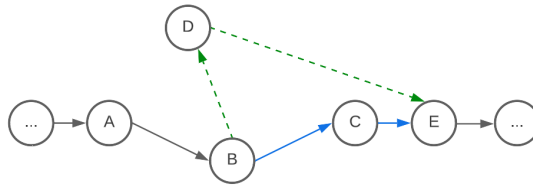


Figura 7: Opção 2 de teste para a viagem quadrangular.

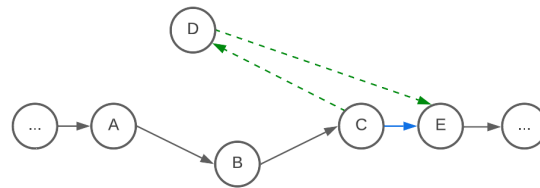


Figura 8: Opção 3 de teste para a viagem quadrangular.

A opção 1 verifica a redução do trajeto para o tipo triangular, com a saída do *drone* do vértice B , a opção 2 mantém a opção do trajeto do tipo quadrangular, mas realiza a saída a partir vértice B e a opção 3 realiza o teste para o *drone* decolando a partir do nó C e retornando ao caminhão nó E , formando um trajeto triangular.

5. Experimentos Computacionais

O algoritmo foi testado utilizando as instâncias geradas a partir da conhecida TSPLib, conforme exposto em [Freitas e Penna, 2018], assim como os valores de distância, velocidades, autonomia, tempo de preparo, tempo de serviço e tempo de recebimento definidos na referência. Os valores dos parâmetros α e β foram ajustados para cada instância ao se verificar a melhor combinação após uma grande quantidade de testes, variando α e β no intervalo $[0.1, 8]$. Estes testes consideraram cerca de 20% da quantidade de vértices como a quantidade do conjunto de formigas liberado, visando a rápida convergência.

A Tabela 1 exibe os valores obtidos para todas as instâncias, onde os valores de α e β selecionados pela calibragem automática são expostos. O valor da quantidade de conjuntos de formigas V adotado é metade da quantidade de vértice (N) da instância. Os resultados apresentados mostram os valores obtidos para 30 testes para cada instância. O algoritmo foi desenvolvido em *python* e executado em ambiente Windows 10, 16GB de RAM e processador AMD Ryzen 5 2600 de 3.40 GHz.

Tabela 1: Resultados obtidos

Instância	N	α	β	S_{bks}	Δbks	$\bar{S}$	ΔS_{max}	t_{bks}	$\bar{t}$
berlin52	52	1	6	172.25	7.99%	194.52	8%	6.50	2.83
bier127	127	1.3	5	3456.8	3.25%	3678.15	10%	23.69	42.58
ch130	130	1.3	6	178.16	-1.14%	182.32	10%	44.13	52.16
d198	198	1.3	5	461.83	1.31%	480.78	11%	27.69	264.32
eil51	51	1	5	13.45	-5.89%	13.25	9%	11.57	1.95
eil76	76	1.3	5	16.35	-8.68%	16.08	15%	27.14	9.03
kroA100	100	1.3	4	587.8	-3.19%	628.32	21%	10.95	20.86
kroA150	150	2	6	729.95	0.92%	793.84	18%	10.95	58.29
kroA200	200	2	6	870.65	-7.44%	880.69	14%	6.53	174.08
kroB150	150	1.3	5	763.15	-4.47%	779.83	18%	20.20	82.07
kroB200	200	1.3	4	804.47	5.88%	892.90	8%	60.00	210.91
kroC100	100	1.3	6	575.3	-6.35%	591.96	17%	20.07	20.41
kroD100	100	1.3	6	606.45	0.96%	636.28	8%	9.15	19.08
kroE100	100	1.3	6	556.3	6.46%	617.63	7%	8.57	34.10
lin105	105	1.3	5	378.25	0.86%	411.92	14%	10.64	25.56
pr107	107	1.3	6	1017.5	-11.10%	929.76	4%	15.48	37.37
pr124	124	1	6	1553.8	-0.40%	1586.49	5%	25.45	53.34
pr136	136	1.3	4	2542	1.06%	2713.99	16%	44.50	79.14
pr144	144	1	5	1666.25	1.96%	1717.11	3%	62.63	74.77
pr152	152	2	5	1919.35	-0.32%	2082.16	21%	70.33	82.69
rat195	195	1.3	6	71.4	0.25%	73.62	6%	9.76	154.91
rat99	99	0.8	6	37.15	-1.08%	37.67	7%	24.13	23.46
rd100	100	1.3	4	240.46	-5.50%	238.48	10%	10.83	25.13
st70	70	1.3	6	20.5	-1.31%	21.11	8%	3.85	6.60

A coluna S_{bks} identifica o melhor valor conhecido (bks), conforme apresentado em [Freitas e Penna, 2018]. A coluna Δbks mostra a diferença percentual do valor encontrado com o bks , valores negativos indicam solução melhor que o bks . A coluna $\bar{S}$ exibe o valor da média dos valores encontrados nos testes realizados. A coluna ΔS_{max} identifica o valor máximo encontrado nos testes, em termos percentuais do melhor valor encontrado. As duas últimas colunas indicam o tempo de processamento para o bks e para o algoritmo desenvolvido, respectivamente.

As primeiras colunas mostram que o método de ajuste dos parâmetros definiu baixos valores de α e altos valores de β , conforme esperado, aumentando a importância do ferormônio na escolha dos destinos das formigas, visto que os valores de atratividade e ferormônio são menores que a unidade. Observa-se que o algoritmo conseguiu entregar bons resultados na maioria das instâncias, se comparado com a melhor solução conhecida, visto a incidência de valores negativos na coluna Δbks . Dentre as 24 instâncias, o resultado apresentado é menor (melhor) que o bks em 13 e próximo do resultado conhecido em 7 instâncias. Porém, o resultado em 4 instâncias está entre 3% e 8% maior (pior) que o melhor valor conhecido. A média da variação do resultado em relação

ao bks é de -1,08%.

O tempo de processamento indicado na coluna $\bar{t}$ mostra que o algoritmo implementado requer maior processamento quanto maior a instância, ao contrário do valor exposto na coluna t_{bks} , onde o tempo de processamento para o bks é exposto. Observa-se que o tempo para o algoritmo desenvolvido é menor ou próximo do algoritmo de [Freitas e Penna, 2018], com exceção para as maiores instâncias, próximas de 200 vértices, onde o algoritmo desenvolvido apresenta um grande custo computacional. Porém, é importante destacar que o tempo de processamento de referência (t_{bks}) foi extraído de [Freitas e Penna, 2018], pois não foi possível executar este algoritmo no mesmo ambiente que o algoritmo proposto.

6. Conclusão

O método ACO tem grande potencial para resolver o problema FSTSP, visto sua capacidade de construir soluções viáveis e explorar variações para a busca de um ótimo global. O tempo de processamento deve ser um ponto de atenção, pois há um grande aumento quando a instância se aproxima de 200 vértices. Deve ser estudado a utilização de recursos computacionais e de programação para otimizar o código visando a redução do custo computacional.

Os resultados apresentados mostraram bons valores em comparação ao bks , porém, em poucos casos, o algoritmo não conseguiu se aproximar do melhor valor conhecido. Tais limitações podem estar associadas a limitações do algoritmo que foram assumidas, como a escolha do vértice de retorno do *drone* após a realização de uma entrega, que se restringe aos dois próximos nós escolhidos pelo caminhão.

Uma melhoria identificada é a necessidade de modificar o algoritmo para que este possa escolher, utilizando os princípios de otimização por colônia de formigas, o nó de retorno ao caminhão dentre todos os nós viáveis no percurso que o caminhão realizou após a sua decolagem. Com isto, espera-se que seja possível explorar novas soluções, que não sejam limitadas ao exposto na Figura 1.

Um algoritmo que seja mais próximo da realidade deve ser capaz de trabalhar com mais de um *drone*, considerando a limitação do caminhão em os receber e preparar, ou seja, deve ser capaz de gerenciar filas de preparação e recebimento do *drones* pelo caminhão. Esta modificação aumenta a complexidade do algoritmo e requer ajustes significativos no procedimento de Busca Local assim como necessita de outras instâncias para que seja possível a análise de seus resultados.

Referências

- Agatz, N., Bouman, P., e Schmidt, M. (2018). Optimization Approaches for the Traveling Salesman Problem with Drone. *Transportation Science*, 52(4):965–981.
- Bell, J. E. e McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18(1):41–48. ISSN 1474-0346.
- Chung, S. H., Sah, B., e Lee, J. (2020). Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers Operations Research*, 123: 105004. ISSN 0305-0548.
- Crişan, G. C. e Nechita, E. (2019). On a cooperative truck-and-drone delivery system. *Procedia Computer Science*, 159:38–47. ISSN 1877-0509. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 23rd International Conference KES2019.

- Dorigo, M. (1992). Optimization, learning and natural algorithms. *Ph.D. Thesis, Politecnico di Milano*.
- Freitas, J. e Penna, P. (2018). A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research*.
- Korczak, J. e Kijewska, K. (2019). Smart logistics in the development of smart cities. *Transportation Research Procedia*, 39:201–211. ISSN 2352-1465. 3rd International Conference "Green Cities – Green Logistics for Greener Cities", Szczecin, 13-14 September 2018.
- Marques Junior, E., Ochi, L. S., Coelho, I. M., Maculan, N., Mladenovic, N., Coelho, V. N., e Coelho, B. N. (2023). A two-phase multi-objective metaheuristic for a green uav grid routing problem. *Accepted for publication in [Optimization Letters - Springer]*.
- Mavrovouniotis, M., Ellinas, G., e Polycarpou, M. (2018). Ant colony optimization for the electric vehicle routing problem.
- Mavrovouniotis, M., Li, C., Ellinas, G., e Polycarpou, M. (2019). Parallel ant colony optimization for the electric vehicle routing problem.
- Mbiadou Saleu, R. G., Deroussi, L., Feillet, D., Grangeon, N., e Quilliot, A. (2022). The parallel drone scheduling problem with multiple drones and vehicles. *European Journal of Operational Research*, 300(2):571–589. ISSN 0377-2217.
- Murray, C. C. e Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54: 86–109. ISSN 0968-090X.
- Narayanan, A., Misra, P., Ojha, A., Bandhu, V., Ghosh, S., e Vasan, A. (2022). A reinforcement learning approach for electric vehicle routing problem with vehicle-to-grid supply.
- Othman, W., Wahab, A., Alhady, S., e Wong, H. (2018). Solving vehicle routing problem using ant colony optimisation (aco) algorithm. *International Journal of Research and Engineering*, 5: 500–507.
- Ponza, A. (2016). *Optimization of drone-assisted parcel delivery*. PhD thesis.
- Rizzoli, A.-E., Montemanni, R., Lucibello, E., e Gambardella, L. M. (2007). Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, 1:135–151.
- Wang, Z. e Sheu, J.-B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 122:350–364.
- Zhang, H., Zhang, Q., Ma, L., Zhang, Z., e Liu, Y. (2019). A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows. *Information Sciences*, 490:166–190. ISSN 0020-0255.
- Çatay, B. (2009). *Ant Colony Optimization and Its Application to the Vehicle Routing Problem with Pickups and Deliveries*, p. 219–244. ISBN 978-3-642-04038-2.