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

EXPLORAÇÃO E OTIMIZAÇÃO DO PROBLEMA DOS MÚLTIPLOS CAIXEIROS VIAJANTES

Helen Vitaline de Castro Santos
Universidade Federal de Viçosa – Campus Rio Paranaíba
Rodovia BR 230 KM 7, Rio Paranaíba - MG, 38810-000
helen.castro@ufv.br

Laís Gonçalves Viana
Universidade Federal de Viçosa – Campus Rio Paranaíba
Rodovia BR 230 KM 7, Rio Paranaíba - MG, 38810-000
Lais.viana@ufv.br

Thiago Henrique Nogueira
Universidade Federal de Viçosa – Campus Rio Paranaíba
Rodovia BR 230 KM 7, Rio Paranaíba - MG, 38810-000
thiagoh.nogueira@ufv.br

Raiane Ribeiro Machado Gomes
Universidade Federal de Viçosa – Campus Rio Paranaíba
Rodovia BR 230 KM 7, Rio Paranaíba - MG, 38810-000
raianemachado@ufv.br

Larissa Sousa Campos
Universidade Federal de Viçosa – Campus Rio Paranaíba
Rodovia BR 230 KM 7, Rio Paranaíba - MG, 38810-000
larissa.sousa@ufv.br

RESUMO

Este artigo explora o problema dos Múltiplos Caixeiros Viajantes (mTSP) através da comparação entre o modelo de otimização usando Colônia de Formigas e um projeto anterior que utilizou cinco heurísticas. O mTSP é uma variação do problema clássico do Caixeiro Viajante, onde n pontos devem ser visitados por um conjunto de m caixeiros, cada um apenas uma vez. Os algoritmos foram implementados em Python neste estudo, com o objetivo de identificar qual das seis metaheurísticas obteve a solução mais eficiente.

PALAVRAS CHAVE. Heurística, Problema do Caixeiro Viajante Múltiplo, Colônia de Formigas (ACO).

ABSTRACT

This article explores the Multiple Traveling Salesman Problem (mTSP) by comparing the optimization model using Ant Colony and a previous project that used five heuristics. mTSP is a variation of the classic Traveling Salesman problem, where n points must be visited by a set of m salesmen, each one only once. The algorithms were implemented in Python in this study, with the aim of identifying which of the six metaheuristics obtained the most efficient solution.

KEYWORDS. Heuristic, Multiple Traveling Salesman Problem, Ant Colony Optimization.

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

1. Introdução

Ter competências e habilidades para solucionar adversidades no ambiente de trabalho e na vida cotidiana é uma aptidão almejada por todos, mas esta não é uma tarefa simples. A sociedade busca inovar através de tecnologias e métodos que auxiliem na solução de obstáculos emergentes. Sendo assim, a todo momento, atribuições referentes a setores econômicos, transporte, logística, produção e entre outros, são resolvidas maximizando e minimizando os critérios concebidos. Onde são simbolizadas matematicamente por modelos de Otimização que reivindicam algoritmos e grafos em sua resolução computacional [MACAMBIRA, 2022].

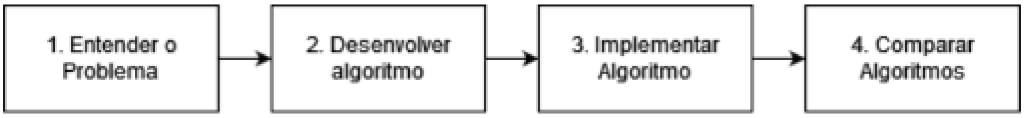
Estudos de modelos de otimização aplicados à logística possibilitam a redução de custos envolvidos nesta área, além de promover uma melhoria significativa do nível de serviço, aumento da capacidade de cumprir prazos, competitividade e credibilidade das organizações [BRAGA 2007]. Segundo Cerqueira e Cravo [2006], tais estudos dizem respeito a problemas de otimização combinatória, envolvendo uma função objetivo a ser otimizada, sujeita a um conjunto de restrições.

Exemplos desses problemas são o Problema do Caixeiro Viajante, o Problema da Mochila, Problema de Roteamento de Veículos e Programação de Horários. De acordo com Goldbarg e Luna [2000], o problema do Caixeiro Viajante (PCV), em inglês chamado de travelling salesman problem (TSP), é um dos problemas mais conhecidos e tradicionais da programação matemática, que lida com passeios em pontos de demanda, como cidades, postos de trabalho e depósitos. Apesar de ser facilmente compreendido, é um problema de difícil solução, já que é classificado como NP-Difíceis [KARP 1975].

O problema do Caixeiro Viajante (PCV) é reconhecido por apresentar uma elevada complexidade de solução devido ao grande número de soluções existentes em instâncias de médio e grande porte, segundo Garey e Johnson [1979]. Os autores também afirmam que um problema NP-difícil é comumente solucionado por meio de heurísticas, procedimentos estes que possibilitam a obtenção de uma ou mais soluções através de uma abordagem intuitiva que considera a estrutura específica do problema. Whitley et al. [1991] afirmam que o objetivo destes procedimentos é encontrar a melhor solução possível dentre as soluções viáveis, buscando maximizar ou minimizar uma função objetivo, enquanto satisfaz a todas as restrições estabelecidas pelo problema.

O objetivo do estudo é utilizar uma heurística eficiente para o problema de mTSP e comparar seus resultados com algoritmos já existentes na literatura Sendo assim, este estudo terá como base de dados as informações do artigo publicado "Estudo Comparativo de Metaheurísticas Aplicadas ao Problema do Caixeiro Viajante Múltiplo" que envolve a mesma problemática. Sendo assim, o modelo utilizado será o Modelo de Otimização por Colônia de Formigas, para diferenciar dos demais aplicados e assim comparar os resultados das diferentes heurísticas manipuladas para a solução deste problema.

Figura 1: Desenvolvimento do trabalho



Fonte: autores [2023]

2. Referencial Teórico

O Problema do Caixeiro Viajante Múltiplo representa vendedores que tem como propósito visitar n cidades por uma única vez, dando início em uma localização e retornando para o mesmo local de origem. Sendo assim, o objetivo do mTSP é encontrar o caminho mais curto com o menor custo, ou seja, minimizar o trajeto percorrido pelo viajante ou os custos envolvidos [DANTZIG 1954].

O mTSP é um problema NP-difícil, assim como o TSP clássico, e não existe um algoritmo conhecido que possa resolver o problema, ou seja, os métodos de solução aplicados a instâncias reais são, em geral, heurísticos, isto é, não asseguram a obtenção da solução ótima do

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

ponto de vista matemático [DA CUNHA 2002]. Sendo assim, de acordo com Helsgaun [2000] existem dois grupos de procedimentos heurísticos para solucionar o problema do caixeiro viajante múltiplo: os métodos de construção de roteiros e os métodos de melhoria de roteiros.

Para o método de construção de roteiros as cidades serão alocadas gradualmente, seguindo regras de construção, em outros termos, as sequências parciais de cidades são definidas ao longo do algoritmo e, posteriormente, não são modificadas, seguindo um processo iterativo [DA CUNHA 2002]. Já o método de melhoria busca testar novas rotas refazendo conexões para melhorar o roteiro pré-estabelecido, com o intuito de diminuir a distância total percorrida [LIN e KERNIGHAN 1973].

Para a construção dos roteiros Laporte [2000] ressalta o método vizinho mais próximo e o método de inserção, já Reinelt [1994] exemplifica heurísticas baseadas em árvores de cobertura e o método do economista. Para a melhoria das rotas, Helsgaun [2000] traz a heurística de Lin e Kernighan [1973] sobre as melhorias do tipo k-opt, e Laporte [2000] aponta as metaheurísticas simulated annealing e busca tabu. Dessa forma, é entendível que existem diversas possibilidades para a resolução do problema do caixeiro viajante múltiplo.

Dessa forma, a quantidade de operações necessárias para solucionar o problema e analisar todas as possibilidades é o que dita o nível de complexidade do algoritmo que será usado. Assim como no caso do PCVM, onde múltiplos caixeiros querem visitar n cidades, muitos outros problemas não podem ser solucionados através de algoritmos em tempo polinomial, sendo denominados como problemas computacionalmente intratáveis. Tais problemas são conhecidos como NP-difícil [GAREY e JOHNSON 1979]. Assim, pode-se afirmar que o PCVM também pertence à classe NP-Difícil, onde o uso de métodos exatos não se mostra uma opção viável. Neste contexto, a resolução para o PCVM proposta neste trabalho utiliza conceitos de meta-heurísticas [MESTREIA 2010].

Para Silva Drummond et al. [2000], uma excelente opção para a solução de problemas NP-Difíceis e NP-completos, na área de otimização combinatória, são as chamadas heurísticas inteligentes ou Meta-Heurísticas. Essas metodologias podem ser explicadas como sendo uma estrutura algorítmica geral, aplicável a variados problemas de otimização com poucas modificações que possam adaptá-las a um problema específico, [BRAGA 2007]. Kureuchicj, MiagKikh et al. [2001] ressaltam que a principal característica das Meta-heurísticas é a de possuírem mecanismos que buscam, em problemas de otimização, não ficar presos a ótimos locais, quando ainda estamos longe de um ótimo global.

Segundo Kanda [2014], A Otimização por Colônia de Formigas (Ant Colony Optmization, ACO) está entre as principais meta-heurísticas usadas para o TSP. Ela tem alcançado resultados consistentes para problemas de otimização combinatória, recebendo esta denominação por ser baseada na maneira como colônias de formigas reais encontram rotas mais curtas até suas fontes de alimento, como mostram Dorigo et. Al. [1996]. A partir do primeiro algoritmo ACO, denominado Ant System, que obteve resultados encorajadores para o TSP, a ACO tem sido aplicada com sucesso para vários problemas NP-difíceis de otimização combinatória.

Stutzle [2004] afirma que na ACO, as soluções são geradas num processo iterativo no qual agentes inteligentes, as formigas, percorrem todas as cidades de uma instância de TSP, por exemplo, escolhendo a cada iteração a próxima cidade de sua rota de maneira estocástica através da Regra de Transição de Estado (RTE).

Para Rodrigues [2007], a solução para o problema tratado é um caminho de menor custo entre os estados do problema, respeitando suas restrições. A complexidade de cada formiga é tal que uma formiga sozinha poderia resolver o problema, mas, provavelmente, de forma ruim. Cormen et al. [2002], diz que as soluções de boa qualidade são obtidas somente como o resultado da cooperação global entre todos os agentes da colônia que criam soluções diferentes ao mesmo tempo.

Os trabalhos relacionados mostram que o PRV ainda é um problema difícil de resolver na prática, pois os métodos existentes são muito complexos e pouco eficientes. Assim, existe uma necessidade de abordagens mais simples e mais eficazes para esse problema. Na literatura,

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

algoritmos que usam Colônia de Formigas (CF) [Dorigo et al. 1996] têm obtido bons resultados com baixo custo computacional para resolver variações do PRV [Bell; Mcmullen 2004; Tan, Zhuo e Zhang 2006; Yu, Yang e Yao 2009]. Logo, um algoritmo que usa Colônia de Formigas (CF) pode ser uma solução adequada para o PRV.

Segundo Gonçalves [2008], o trabalho apresenta um modelo que usa a Otimização por Colônias de Formigas, que foi introduzida por Colorni et. al [1991]. Um dos principais desafios em usar os algoritmos ACO é a necessidade de transformar o problema em um problema de encontrar o menor caminho em grafos. Isso é necessário porque o ACO é baseado em um processo no qual um grupo de formigas procura o menor caminho entre o ninho e a fonte de comida [DORIGO 2006].

Para Carvalho [2008] diversos problemas combinatoriais são utilizados para a resolução de impasses logísticos, tais como, roteamento de veículos, Caixeiro Viajante, otimização de roteiros e entre outros. A partir disso o algoritmo ACO foi escolhido para determinar a solução ótima do PCV, sendo que os resultados obtidos serão comparados com outras heurísticas, a fim de inferir qual delas apresentará uma melhor resposta ao problema em questão. Dessa forma, esses estudos contribuíram para a formulação do modelo matemático e resolução do código para o Problema do Caixeiro Viajante.

3. Formulação Matemática do Problema

Através da definição apresentada sobre o problema do caixeiro viajante múltiplo, tornou-se necessário articular o m-TSP através da formulação demonstrada por Bektas [2006]. Sendo assim, o problema pode ser descrito através de um grafo G=(V,A), na qual V={1,2,...,n} é o conjunto dos vértices que simbolizam as n cidades presentes no m-TSP e A o conjunto das arestas que representam as conexões entre os vértices.

A cada aresta percorrida é associado um custo, que pode ser definido a partir de distâncias, despesas ou tempo. A representação pode ser representada da seguinte forma:

c_{ij} - custo de um caixeiro viajar da cidade i ∈ V para cidade j ∈ V.

Ainda, deve-se levar em consideração outras variáveis que influenciam diretamente a resolução do m-TSP, como o número de caixeiros, um conjunto para os vértices do grafo e as variáveis binárias. No qual seguem as seguintes simbologias:

S: número de elementos do subconjunto dos vértices do grafo atribuídos para cada caixeiro;
m: número de caixeiros viajantes;
x_{ij}=1: se um caixeiro viaja da cidade i ∈ V para cidade j ∈ V;
x_{ij}=0: se o caixeiro não viaja da cidade i ∈ V para cidade j ∈ V.

Com todas as variáveis definidas, o modelo matemático pôde ser estruturado. Contendo a função objetivo (1), que garante a minimização da distância total percorrida por todos os caixeiros viajantes. Também, foram implementadas as restrições impostas pelo mTSP.

$$Min \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} \cdot X_{ij} \quad (1)$$

$$\sum_{j \in S}^{n} X_{1j} = m \quad (2)$$

$$\sum_{i \in S}^{n} X_{i1} = m \quad (3)$$

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

$$\sum_{i \in S}^{\sum_{j \in S}^{X}} ij \;\geq 1, \quad \forall S \subseteq V \backslash \{1\}, S \neq \emptyset \tag{4}$$

$$\sum_{j=1}^{n} Xij = 1, \; i = 2, \dots, n \tag{5}$$

$$\sum_{i=1}^{n} Xij = 1, \; j = 2, \dots, n \tag{6}$$

$$Xij \in \{0,1\}, \forall \, i, \, j \in V \tag{7}$$

A restrição 2 tem como função garantir que todos os m caixeiros saiam da cidade de origem, já a 3 certifica que os mesmos m viajantes retornem para a mesma cidade. Em outras palavras, existem m arestas vinculadas a cidade de partida que reproduzem as arestas de retorno dos caixeiros. A quarta (4) limitação imposta tem como objetivo assegurar a não formação de subconjuntos de cardinalidade S que não contenham no depósito, ou seja, os elementos serão delimitados apenas a quantidade presentes em S. Já as restrições 5, 6 e 7 correspondem às regras de atribuição presentes no m-TSP.

3.1 ACO

A metaheurísticas escolhida para a resolução do problema do caixeiro viajante múltiplo foi a Colônia de Formigas (ACO). Na qual, Stutzle [2004] afirma que, as soluções são geradas num processo iterativo no qual agentes inteligentes, as formigas, percorrem todas as cidades de uma instância de TSP, por exemplo, escolhendo a cada iteração a próxima cidade de sua rota de maneira estocástica através da Regra de Transição de Estado (RTE). A RTE do Ant System é mostrada na equação a seguir:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum [\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}, & \text{onde j é uma aresta permitida à } k \\ 0 & \text{caso contrário} \end{cases}$$

Equação 1: RTE do *Ant System*

A equação, que determina a probabilidade de um vértice i ser escolhido por uma formiga representa o vértice no qual a formiga k se encontra, e j um dos possíveis vértices para o qual ela pode se deslocar. Os vértices permitidos à formiga são aqueles ainda não visitados até a fase atual da construção da solução. Cada formiga armazena os nós já visitados em sua memória, que é denominada como lista tabu. τij (t) corresponde `a quantidade de feromônio presente na aresta entre os vértices i e j no instante t. $\eta ij$ corresponde `a visibilidade da aresta, que ́é definida pelo valor 1/cij, onde cij é o custo de deslocamento da aresta. α e β são parâmetros que definem o peso da trilha de feromônio e da visibilidade, respectivamente, na escolha do próximo vértice pela formiga.

- **A ALF (regra de Atualização Local de Feromônio)**

A ALF (regra de Atualização Local de Feromônio) é definida pela equação 2, onde ξ é um parâmetro com o melhor valor experimentalmente calculado como $\xi = \rho = 0{,}1$. Já $\tau 0$ é outro parâmetro que é utilizado tanto na atualização local de feromônio quanto na quantidade inicial de feromônio depositada em todas as arestas no início da execução do algoritmo. O melhor valor para $\tau 0$ é calculado com base no custo de uma solução gerada aplicando o algoritmo do vizinho mais próximo à instância em análise, pode ser calculada na equação 3, onde n é o número de cidades na

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

instância e Cnn é o custo da solução obtida utilizando o algoritmo do vizinho mais próximo na instância.

$$\tau_{ij} = (1-\xi)\tau_{ij} + \xi\tau_0$$

Equação ALF (2)

$$\tau_0 = 1/nC^{nn}$$

Equação (3)

- **TACO (*Team Ant Colony Optimization*, Otimização por Colônia com Equipes de Formigas)**

O TACO (Team Ant Colony Optimization, Otimização por Colônia com Equipes de Formigas) é um algoritmo desenvolvido por [Vallivaara 2008] que adapta o ACS para instâncias de MTSP. Em vez de usar formigas individuais para gerar soluções para o TSP, o TACO utiliza equipes de formigas compostas por m elementos, onde cada formiga representa um caixeiro na construção de uma solução para o MTSP, e cada equipe tem sua própria lista tabu.

No início da construção da solução, todas as formigas de todas as equipes são colocadas no depósito. Para distribuir a carga de trabalho, a formiga com a menor rota parcial, em qualquer momento da construção da solução, escolhe sua próxima cidade. Embora seja eficiente na distribuição da carga de trabalho, muitas vezes as escolhas das formigas resultam em soluções sub-ótimas.

Para lidar com esse problema, a cada escolha é verificado se ceder a cidade escolhida para qualquer outra formiga da equipe resultaria em uma solução total menor, considerando também o retorno ao depósito. Se for o caso, a formiga com a menor rota parcial se move e pode fazer uma nova escolha para o próximo vértice.

Matematicamente, considerando uma formiga k na posição do vértice Vk, onde Vo representa o depósito da instância, C(Vi, Vj) é o custo de deslocamento entre dois vértices quaisquer Bi e Vj, CP(k) é o custo parcial total da rota já percorrida pela formiga k, e Vj é o vértice escolhido como o próximo. O método verifica, antes da formiga se mover, se existe qualquer outra formiga l da instância em que:

$$C(V_l, V_j) + C(V_j, V_0) + CP(l) < C(V_k, V_j) + C(Vj, V_0) + CP(k)$$

Equação (4)

Caso haja, permite-se então que a formiga l escolha, o próximo e novo vértice, se movimentando para o mesmo.

No processo de otimização das instâncias do problema MSPT, é utilizada a técnica de busca local 2-opt, que consiste em trocar duas arestas na solução, sendo aplicada a todas as soluções geradas. Além disso, a busca local 3-opt é aplicada apenas à melhor solução obtida entre as N soluções geradas pelos times, com o objetivo de comutar três arestas simultaneamente. Para implementar essa abordagem, foi desenvolvido um protótipo baseado em uma variação do TACO de Vallivaara [2008], que é descrito em detalhes a seguir.

Tabela 1: Pseudocódigo

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

Algoritmo ACO para MTSP		
	Entrada: Instâncias MTSP e parâmetros ACO	(1)
	Saída: Melhor Solução MTSP criada	(2)
1	**Início**	(3)
2	n <- Número de cidades da instância, incluindo depósito;	(4)
3	m <- Número de Caixeiros da Instância;	(5)
4	*v0* <- Cidade depósito da instância;	(6)
5	N <- Número de soluções geradas em cada ciclo;	(7)
6	Definir o objetivo do MTSP;	(8)
7	Construir a matriz de custo da instância	(9)
8	**Inicialize a matriz de feromônio;**	(10)
9	**Enquanto** o critério de para não for aceito **faça**	(11)
10	**Para** contador_de_soluções <- 1 até N	(12)
11	Crie um time com m formigas;	(13)
12	Esvazie a lista tabu da colônia;	(14)
13	Posicione as m formigas do time em *v0*	(15)
14	**Enquanto** houver cidades não visitadas **faça**	(16)
15	Selecione uma formiga k do time;	(17)
16	Crie a lista de canditados para k;	(18)
17	**Escolha a próxima cidade j da rota de k;**	(19)
18	Verifique o movimento selecionado;	(20)
19	se houver uma formiga l que resulta num melhor movimento **então**	(21)
20	Selecione a formiga l;	(22)
21	Atualize a lista de canditados l;	(23)
22	**Escolha uma nova próxima cidade *j para l*;**	(24)
23	**Fim**	(25)
24	Movimente a formiga selecionada para a cidade escolhida;	(26)
25	Insira a cidade visitada na lista tabu;	(27)
26	**Aplique a atualização local de feromônio à aresta percorrida;**	(28)
27	**Fim**	(29)
28	Retorne todas as m formigas para *v0;*	(30)
29	Aplique a busca local à solução MTSP criada;	(31)
30	Atualize a melhor solução do ciclo S_{m_ciclo};	(32)
31	**Fim**	(33)
32	Atualize a melhor solução da execução S_{m_exec};	(34)
33	**Aplique a ataulização global de feromônio;**	(35)
34	**Fim**	(36)
35	**Retorne** Sm_exec;	(37)
36	**Fim**	(38)

Fonte: Autores [2023]

- **Inicialização**

As instâncias de MTSP e os parâmetros utilizados neste problema, são as entradas do algoritmo ACO. A instância é definida pela variação de contexto aos quais o problema será resolvido, sendo estes os números de caixeiros (m) e cidades, ou clientes, a serem atendidos (n). O retorno deste modelo é a melhor solução de MTSP, sendo usado apenas um único depósito, ao qual os caixeiros devem sair e retornar.

O Algoritmo ACO se inicializa declarando as principais variáveis. Estas estão presentes na linha 2 e 4 onde são declarados, de acordo com as instâncias, o valor inteiro n de cidades, e m de caixeiros, onde V0 equivale ao depósito em cada ciclo do algoritmo. De forma ilustrativa, N representa o número de formigas que criam, de forma simultâneas, soluções para o problema. As soluções são criadas de forma sequencial, e quando se atinge o critério de paradas, encerra-se o ciclo. O objetivo do MTSP é estabelecido na linha 6, que neste caso é minimizar o custo total da solução.

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

Na linha 7 é construída a matriz de custos da instância. É utilizado custos previstos, a fim de melhor representar o ambiente real. É por estes e outros pormenores que a matriz assume um caráter assimétrico e todos os custos terão que ser fornecidos para sua confecção. Logo o custo de deslocamento entre dois pontos pode ser diferente dependendo do sentido.

Na linha 8, é inicializada a matriz de feromônio, ocorrendo com que todas as arestas recebem a mesma quantidade deste, indicando que informações sobre as instâncias são inexistentes até o momento. Na linha 9, dá-se início aos ciclos do algoritmo, onde os mesmos são limitados pelo critério de parada. Usar diferentes critérios de parada será uma possibilidade útil em experimentos futuros, permitindo identificar qual é o mais adequado para cada algoritmo desenvolvido.

- **Construção das soluções**

Iniciam-se a construção das soluções na linha 10. Já na linha 11, times de m formigas são criados, sendo cada uma a representação de cada caixeiro, onde apenas um time constrói uma solução de cada vez. Em uma lista tabu comum são armazenadas as cidades já visitadas pelos caixeiros durante a construção da solução, onde são esvaziadas na linha 12, estabelecendo que todas as cidades ainda possuem chances de serem exploradas. Por fim, na linha 13, todas as formigas são alocadas ao depósito, de onde partirão para mais uma rota.

Os movimentos são inicializados na linha 14, onde as formigas viajam de sua cidade atual para a próxima da rota. Na linha 15, pode-se notar que a formiga escolhida para movimentar é aquela que possui a rota parcial com o menor custo, sendo este método aplicado por Vallivaara [2008]. Em caso de empate, a escolha será feita de forma aleatória, selecionando a formiga que possui o menor número de identificação no protótipo. Este método é eficaz para o minmax MTSP, pois busca equilibrar o custo das rotas durante o período de escolha formiga que inicializará o movimento.

Após a seleção da formiga, há a criação da lista, como mostra a linha 16, composta pelas Clsize cidades que estão em locais mais próximos da formiga selecionada. Com finalidade de restringir o espaço de busca das arestas que conferem uma melhor solução, a lista de candidatos também busca reduzir o tempo computacional. Em seguida, a linha 17 corresponde à Regra de Transição de Espaço (RTE) da ACO, ou seja, quando a formiga escolhe a próxima cidade de sua rota a ser visitada.

É realizada, após a escolha da próxima cidade da formiga selecionada, uma verificação que precede a inicialização do movimento, como ressaltado na linha 18. Como as escolhas das formigas podem tender a soluções não ótimas, de acordo com o menor custo de sua rota, essa verificação torna-se imprescindível. A fim de tratar este problema, faz-se uma verificação a cada ação da formiga se, caso a cidade escolhida fosse atribuída a qualquer outra formiga, não ocasionaria em um menor deslocamento que o da formiga selecionada, levando-se em consideração também o retorno à origem (depósito).

Na linha 20, a formiga que resulta no menor caminho é escolhida, mas não se move de forma obrigatória para a cidade que a formiga anterior escolheu, sendo refeita para a posição de nova formiga, a lista de candidatos. Logo, na linha 21 e 22, a RTE é executada novamente, possibilitando a escolha de uma nova cidade.

Na linha 24 há a inclusão da cidade na rota da formiga que a escolheu, posteriormente a posição dessa formiga é atualizada. Há também a inclusão da cidade visitada na lista tabu, como mostra a linha 25. Desta forma a cidade não pode ser selecionada novamente durante a construção da solução atual. Já na linha 26, ocorre a atualização do feromônio. Todo esse processo, que abrange a linha 15 à linha 26, é repetido até que não haja mais cidades a serem visitadas.

4. Resultados

O código do algoritmo foi testado utilizando dois computadores, sendo o primeiro, um notebook Dell Vostro, com um processador Intel Core i7-7500 CPU @ 2.90 GHz e memória *ram* de 16 GB. Já o segundo dispositivo consiste em um computador ASUS com um processador

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

Intel(R) Core(TM) i5-1035G1 CPU @ 1.20 GHz e uma memória RAM de 6 GB. O código foi redigido em *Python* e usa bibliotecas, tais como *random* e *math* para gerar números aleatórios e realizar operações matemáticas

Por se tratar de um estudo comparativo com um trabalho já publicado, a obtenção dos resultados para este artigo em questão seguiu etapas similares às do artigo defrontado. Neste, foi testado metaheurísticas diferentes para a resolução do problema do caixeiro viajante. Outro aspecto importante deste estudo é que, assim como o trabalho base, as metaheurísticas foram todas executadas utilizando o mesmo tempo computacional estabelecido em He e Hao [2022]: (n/100) × 240 segundos. Dessa forma, se o critério de parada de uma metaheurísticas for alcançado antes do limite de tempo máximo, o método é reiniciado.

No trabalho base, os métodos BRKGA e ILS foram os que apresentaram os melhores resultados no estudo, conforme destacado no trabalho de referência. Por outro lado, o método LNS obteve os piores resultados. Em um total de 11 instâncias avaliadas, o BRKGA alcançou um RPD igual a zero em todas as instâncias, enquanto o ILS obteve um RPD igual a zero em 7 das instâncias analisadas

Para início de análise, vale ressaltar que a designação das instâncias foi feita seguindo o padrão mstp n_K. Onde n é o número de clientes e k o número de caixeiros. No total foram realizados testes em 3 instâncias, cenários diferentes, onde houve uma variação do número de caixeiros, sendo estes compreendidos em 3, 5 e 10. Vale ressaltar que os dados de entrada foram os mesmos utilizados pelo estudo defrontado, ressaltando algumas exceções. Já para o número de clientes foi fixado um valor igual a 51.

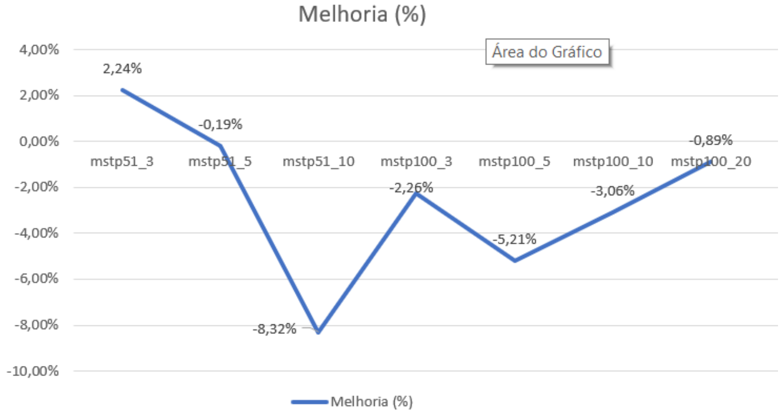
Foram utilizadas 1000 interações para a metaheurísticas Colônia de Formigas (ACO). Para cada uma das 3 instâncias, executou-se 10 vezes o algoritmo. Sendo os valores obtidos para a função, apresentados na tabela 1. Os resultados foram analiticamente observados e comparados com os valores da função objetivo (FO), e os valores ótimos oriundos das funções das metaheurísticas utilizadas pelo artigo defrontado.

Tabela 1: Resultados

Instâncias	HSRN [He e Hao, 2022]	Alves et al. (2022)					Guimarães & Silva (2023)	
		GRASP	BRKGA	LNS	VNS	ILS	ACO	Melhoria (%)
	S*	S_Min	S_Min	S_Min	S_Min	S_Min	S_Min	
mstp51_3	446	446,0	446,0	447,6	446,0	446,0	436,0	2,24%
mstp51_5	471,1	474,7	471,7	484,6	471,1	471,7	472,0	-0,19%
mstp51_10	580,7	579,9	580,1	587,5	579,7	580,2	629,0	-8,32%
mstp100_3	21797,6	22.008,6	21.939,5	23.000,1	22.113,0	22.134,6	22.290,0	-2,26%
mstp100_5	23174,9	23.450,3	23.249,5	25.343,4	23.648,6	23.397,7	24.383,0	-5,21%
mstp100_10	26926,6	27.299,2	26.963,5	31.517,4	27.553,0	27.274,0	27.751,0	-3,06%
mstp100_20	38245,1	38.414,4	38.259,7	48.155,0	38.738,6	38.409,9	38.584,0	-0,89%

Fonte: Autores [2023]

Figura 2: Gráfico de Melhoria



Fonte: Autores [2023]

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

Ao analisar a tabela, pode-se dizer que os valores obtidos a partir da ACO, apresentou resultado significativamente melhor em uma das três instâncias analisadas. Essa solução corresponde à instância mstp 51_3, ou seja, 3 caixeiros viajantes atendendo a uma demanda de 51 cidades, com valor de custo minimizado igual a 436. Se a compararmos às soluções trazidas pelas melhores metaheurísticas ressaltadas pelos estudo Vieira, sendo estas o BRKGA e a ILSR. Nota-se que esta solução é a melhor obtida em todo o estudo comparativo, contemplando à ACO um lugar de destaque entre os demais métodos.

Outro ponto a se analisar são os dados obtidos pela Figura 2, onde, é possível entender que a heurística ACO não é a melhor para a resolução do problema do caixeiro viajante múltiplo. Isso pode ser explicado devido à complexidade do mTSP, que apresenta um grande número de soluções a cada instância adicionada. Sendo assim, a ACO apresenta dificuldades em explorar todas as opções, dessa forma, essa metaheurísticas pode gerar resultados piores em comparação a outras heurísticas.

Devido ao baixo número de instâncias e a pouca exploração de cenários variados, dificultou-se identificar um padrão de comportamento bem definido para a heurística ACO. Em termos de estudo comparativo, a ACO obteve a melhor solução para a função ótima, sendo igual a 436. Porém apresentou resultados piores que os demais métodos em cerca 85 por cento das instâncias analisadas. Deste modo, pode ser conferido ao método, um posicionamento de eficiência equivalente aos métodos BRKGA e a ILSR.

5. Conclusão

Neste artigo, foi abordado o problema dos caixeiros viajantes múltiplos (mTSP), e foram comparados alguns métodos de solução baseados em metaheurísticas, com o objetivo de encontrar boas soluções para esse problema. Para realizar essa análise, um conjunto de 7 instâncias com diferentes cenários de mTSP foi testado, e os resultados mostraram que o algoritmo ACO obteve os melhores resultados, se comparados às metaheurísticas trabalhadas por Alves et. Al. [1995], sendo ela BRKGA, ILS, VNS, LNS e GRASP com instâncias menores.

Embora os resultados obtidos neste estudo não tenham superado os resultados da literatura [He e Hao 2022], houve uma taxa aparentemente considerável de instâncias em que o resultado na ACO foi sorrateiramente pior que os dos métodos dos outros trabalhos. Entretanto, os resultados são úteis para entender como as metaheurísticas podem ser aplicadas para encontrar boas soluções em problemas desafiadores, como é o caso do mTSP. Além disso, esses resultados permitem avaliar como diferentes estratégias de diversificação e intensificação podem melhorar ou piorar cada técnica, dependendo do contexto.

Para trabalhos futuros, é necessário planejar e aprimorar os métodos desenvolvidos, além disso, deve-se investigar novas técnicas, com o objetivo de obter soluções mais próximas ao estado da arte da literatura em relação ao mTSP. Novas melhorias podem elevar o potencial dos resultados gerando conclusões mais promissoras para o problema do caixeiro viajante múltiplo.

Referências

BELL, John E.; MCMULLEN, Patrick R. Ant colony optimization techniques for the vehicle routing problem. **Advanced engineering informatics**, v. 18, n. 1, p. 41-48, 2004.

BRAGA, Edgar Augusto Silva. **Modelagem e otimização do problema do caixeiro viajante com restrições de tempo, distância e confiabilidade via algoritmos genéticos**. 2007. Dissertação de Mestrado. Universidade Federal de Pernambuco.

CARVALHO, M. B.; YAMAKAMI, A. Meta-heurística híbrida de sistema de colônia de formigas e algoritmo genético para o problema do caixeiro viajante. **Trends in Computational and Applied Mathematics,** v. 9, n. 1, p. 31-40, 2008.

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

Colorni A, Dorigo M, Maniezzo V. Distributed Optimization by Ant Colonies. **Proceedings of European Conference on Artificial Life.** 1991; Paris, France. p. 134-142.

CORMEN, Thomas H. et al. Algoritmos: teoria e prática. **Editora Campus,** v. 2, p. 296, 2002.

CERQUEIRA, Fábio Ribeiro; CRAVO, Gildásio Lecchi. METAHEURÍSTICA COLÔNIA DE FORMIGAS APLICADA AO PROBLEMA DO CAIXEIRO VIAJANTE.

DA CUNHA, Claudio Barbieri; DE OLIVEIRA BONASSER, Ulisses; ABRAHÃO, Fernando Teixeira Mendes. Experimentos computacionais com heurísticas de melhorias para o problema do caixeiro viajante. In: **XVI Congresso da Anpet**. 2002.

DANTZIG, George; FULKERSON, Ray; JOHNSON, Selmer. Solution of a large-scale traveling-salesman problem. **Journal of the operations research society of America,** v. 2, n. 4, p. 393-410, 1954.

JOHNSON, David S.; GAREY, Michael R. Computers and intractability: A guide to the theory of NP-completeness. WH Freeman, 1979.

DORIGO, Marco; BIRATTARI, Mauro; STUTZLE, Thomas. Ant colony optimization. IEEE computational intelligence magazine, v. 1, n. 4, p. 28-39, 2006.
.
GOLDBARG, Marco Cesar; LUNA, Henrique Pacca L. Otimização combinatória e programação linear: modelos e algoritmos. Elsevier, 2000.

GONÇALVES, André Ricardo; BRUNETTO, Maria Angélica de Oliveira Camargo. Um novo modelo híbrido baseado em Otimização por Colônia de Formigas e Redes Neurais para identificação de indivíduos com DPOC. In: **Anais do XI Congresso Brasileiro de Informática em Saúde.** 2008.

HELSGAUN, Keld. An effective implementation of the Lin–Kernighan traveling salesman heuristic. **European journal of operational research**, v. 126, n. 1, p. 106-130, 2000.

KANDA, Jorge Yoshio. Sistema de meta-aprendizado para a seleção de meta-heurísticas para o problema do caixeiro viajante. In: **Anais do X Simpósio Brasileiro de Sistemas de Informaçao**. SBC, 2014. p. 651-662.

KARP, Richard M. On the computational complexity of combinatorial problems. **Networks,** v. 5, n. 1, p. 45-68, 1975.

LAPORTE, Gilbert et al. Classical and modern heuristics for the vehicle routing problem. **International transactions in operational research**, v. 7, n. 4-5, p. 285-300, 2000.

LIN, Shen; KERNIGHAN, Brian W. An effective heuristic algorithm for the traveling-salesman problem. **Operations research,** v. 21, n. 2, p. 498-516, 1973.

MACAMBIRA, Ana Flávia Uzeda et al. Tópicos em otimização inteira. **UFRJ**, 2022.

DORIGO, Marco; MANIEZZO, Vittorio; COLORNI, Alberto. Ant system: optimization by a colony of cooperating agents. IEEE transactions on systems, man, and cybernetics, part b (cybernetics), v. 26, n. 1, p. 29-41, 1996.
no

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en

REINELT, Gerhard. **The traveling salesman: computational solutions for TSP applications**. Springer, 2003.

RODRIGUES, Samuel Bellido. Metaheurística Colônia de Formigas aplicada a um Problema de Roteamento de Veículos: caso da Itaipu Binacional. 2007.

BEKTAS, Tolga. The multiple traveling salesman problem: an overview of formulations and solution procedures. omega, v. 34, n. 3, p. 209-219, 2006.

I. Vallivaara. A team ant colony optimization algorithm for the multiple travelling salesmen problem with minmax objective. In Proceedings of the 27th IASTED International Conference on Modelling, Identification and Control, pages 387–392, Anaheim, CA, USA, 2008

WHITLEY, Darrell; STARKWEATHER, Timothy; SHANER, Dan. The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination. Fort Collins: Colorado State University, Department of Computer Science, 1991.

DOI: 10.59254/sbpo-2024-193847
https://proceedings.science/p/193847?lang=en