LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2020

ALGORITMOS MEMÉTICOS APLICADOS AO PROBLEMA DO CAIXEIRO VIAJANTE COM LIMITE DE CALADO

Bruno Duarte, Lucas Caldeira de Oliveira, Marcelo Teixeira, Marco Antonio Barbosa
Universidade Tecnológica Federal do Paraná
Via do Conhecimento, s/n - km 01, Pato Branco - PR
{brunoduarte, lucasoliveira.2017}@alunos.utfpr.edu.br,
{marceloteixeira, mbarbosa}@utfpr.edu.br

RESUMO

Problemas de otimização combinatória, como o Problema do Caixeiro Viajante e suas variações, são recorrentes na literatura. Uma das variações é o Problema do Caixeiro Viajante com Limite de Calado, que possui relação direta com o comércio e transporte marítimo por abordar o cálculo de rotas a serem seguidas por navios cargueiros sem violar as restrições de limite de calado, visando reduzir os custos totais do transporte. Uma solução para esse problema, quando obtida usando-se abordagens exatas, requer um esforço computacional de ordem exponencial e, portanto, inviável para casos práticos. Nesse sentido, abordagens heurísticas e metaheurísticas costumam representar alternativas à abordagens exatas, por viabilizar a obtenção de soluções aproximadas, com menor esforço. Neste trabalho propõe-se a implementação de um Algoritmo Memético, sob a ótica dos Algoritmos Evolutivos, para resolver o problema em questão. Ao fim do trabalho, os resultados obtidos são comparados com a literatura, demonstrando ser competitivos com eles.

PALAVRAS CHAVE. Algoritmos Meméticos. Problema do Caixeiro Viajante. Melhor rota.

Tópicos: MH - Metaheurísticas, OC - Otimização Combinatória

ABSTRACT

Combinatorial optimization problems, such as the Traveling Salesman Problem and its variations, are main focus of research and discussions. One of its variations is the Traveling Salesman Problem with Draft Limits which has a direct relation with maritime commerce and transportation, by addressing the calculation of router to be followed by a cargo ship without violating draft limit restrictions, aiming to reduce total transports costs. A solution to this problem, when obtained using exact approaches, requires computational effort of an exponential order and, therefore, unfeasible for practical cases. In this sense, heuristic and metaheuristic approaches usually represent alternatives to exact approaches, as it makes it possible to obtain approximate solutions, with less effort. In this article it is proposed a Memetic Algorithm implementation, under the perspective of Evolutionary Algorithms, to solve the problem. At the end of the work, the results obtained are compared with the literature, proving to be competitive with them.

KEYWORDS. Memetic Algorithms. Traveling Salesman Problem. Best course.

Paper Topics: MH - Metaheuristics, CO - Combinatorial Optimization

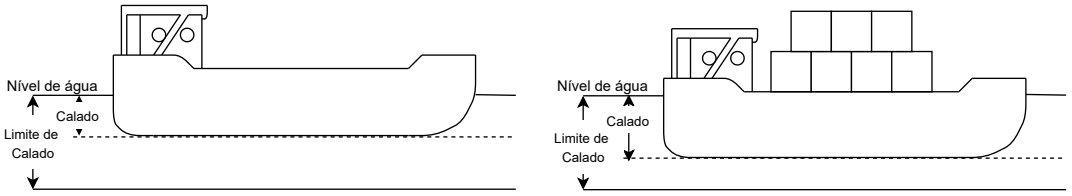
Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

1. Introdução

O Problema do Caixeiro Viajante (PCV) é tema frequente de estudos e discussões. Propõe-se que um caixeiro viajante percorra um determinado número de cidades através da menor distância possível. Essa rota deve ser escolhida de modo que se passe uma única vez por cada cidade, retornando-se, ao final, à cidade de origem [Garey e Johnson, 1979].

Por permear-se por entre diversas áreas do conhecimento, dentre as quais a logística possui maior proximidade, o PCV apresenta inúmeras variações, a maioria adicionando diferentes restrições para a rota a ser percorrida, tais como o PCV com janelas de tempo [Baldacci et al., 2012] e o PCV com entregas e coletas [Battarra et al., 2010].

Outra variação compõe o Problema do Caixeiro Viajante com Limite de Calado (PCVLC) [Glomvik Rakke et al., 2012]. Propõe-se que um navio, com uma certa carga inicial, deve visitar um número n de portos e, como no PCV clássico, cada porto deve ser visitado uma única vez e, ao final da rota, deve-se voltar ao porto de origem. Entretanto, o PCVLC impõe uma restrição adicional à ordem com que os portos são visitados: cada porto possui um limite de calado, como mostra a Figura 1, associado à profundidade máxima que o casco inferior de um navio pode alcançar. Um navio somente pode aportar se o calado máximo, resultante da carga, for menor que o limite de calado do porto em questão, visto que, do contrário, encalhará. A carga do navio diminui de acordo com a demanda – que para os testes efetuados é unitária – do porto visitado, permitindo assim adentrar a outros portos de calado cada vez menor.



(a) Navio sem carga. (b) Navio com carga.

Figura 1: Representação do limite de calado

O problema pode ser exposto como um circuito Hamiltoniano, no qual, a partir de um vértice de origem, busca-se passar por todos os demais vértices uma única vez e retornar ao vértice de origem [Papadimitriou e Steiglitz, 1982]. Quanto a complexidade, o PCVLC é classificado como um problema NP-Difícil, visto que se deriva do PCV clássico com a restrição adicional do limite de calado. A medida em que o número de portos aumenta, cresce também – de maneira exponencial – o número de soluções possíveis. Consequentemente o custo computacional para obter soluções eficientes também aumenta.

Ao propor inicialmente o problema, Glomvik Rakke et al. [2012] apresentam duas formulações matemáticas para sua estruturação, bem como sugerem instâncias de teste e soluções exatas obtidas por meio de algoritmos *branch-and-cut*. Posteriormente, Battarra et al. [2014] agregam três novas formulações para o PCVLC e utilizaram-se novamente do algoritmo *branch-and-cut* para resolver o problema em duas destas formulações, e um algoritmo *branch-cut-and-price* para a terceira, estabelecendo também resultados exatos para o conjunto de instâncias propostos por Glomvik Rakke et al. [2012]. Utilizando-se de duas estratégias baseadas em busca através de vizinhanças variáveis, GVNS (*General Variable Neighborhood Search*), Todosijević et al. [2014] aborda também o problema utilizando 8 estruturas de vizinhança, encontrando resultados ótimos para o conjunto de instâncias propostas por Glomvik Rakke et al. [2012] e estabelecendo novas

instâncias, com maiores números de portos. Essa abordagem, baseada em GVNS, foi na verdade a primeira proposta inspirada em metaheurística para resolver o PCVLC.

A partir dela, outras soluções utilizando metaheurísticas para o PCVLC podem ser encontradas na literatura. Uma abordagem ao problema com a metaheurística GRASP (*Greedy Randomized Adaptative Search Procedure*) foi apresentada por Machado et al. [2015], enquanto Morais et al. [2017] abordaram o problema utilizando o *Simulated Annealing* e Lângaro e Barbosa [2019] apresentaram uma implementação do método *Colônia de Formigas*, todas obtendo excelentes resultados para a grande maioria das instâncias testadas. Entretanto, nenhum destes conseguiu atingir valores ótimos para todas as instâncias, principalmente aquelas com maior número de portos, sendo este um motivo para que resolver o PCVLC utilizando uma metaheurística permaneça como um desafio relevante.

Alternativamente, outra abordagem ao PCVLC são os algoritmos evolutivos que buscam encontrar soluções a partir de modelos dos processos de evolução orgânica, isto é, tendo a natureza como inspiração [Back, 1996]. Os *Algoritmos Meméticos* (AMs) são exemplos desse tipo de abordagem e exploram a evolução de maneira cultural, na qual os indivíduos – agentes – transmitem o conhecimento a respeito de uma determinada situação ou problema às gerações futuras através de técnicas de imitação, os memes [Moscato et al., 1989]. Os AMs contrastam com os Algoritmos Genéticos (AGs), também evolutivos, mas que baseiam-se nos princípios da seleção natural através da qual indivíduos – cromossomos – mais adaptados ao meio em que vivem possuem mais chances de se reproduzir e passar suas características genéticas para novas gerações. Diferentemente, os AMs utilizam-se de um mecanismo de evolução através de gerações com seleção dos mais adaptados para que possam combinar seus memes, compartilhando seu conhecimento com as gerações futuras, visando a obtenção de melhores agentes. De acordo com o conhecimento dos autores deste artigo, até o momento não existem trabalhos abordando o PCVLC sob a ótica dos algoritmos evolutivos.

Este trabalho propõe uma solução para o PCVLC usando uma abordagem baseada em Algoritmos Meméticos. A abordagem é implementada e utilizada como plataforma para a condução de um conjunto de experimentos computacionais. Os resultados dos experimentos são analisados e comparados com outros resultados da literatura relacionada ao PCVLC. O comparativo sugere uma abordagem competitiva em relação às soluções do problema já reportadas na literatura, com capacidade de alcançar uma solução ótima para $79,16\%$ das instâncias.

O presente artigo estrutura-se como segue. A seção 2 apresenta conceitos básicos, como a formulação matemática do problema e uma breve explicação sobre os AMs. A seção 3 detalha a técnica utilizada para implementação do algoritmo. A seção 4 introduz os resultados dos experimentos e, por fim, a seção 5 apresenta algumas conclusões desse trabalho e perspectivas futuras.

2. Conceitos Básicos

Nesta seção, apresenta-se a formulação matemática para o PCVLC bem como conceitos básicos sobre os algoritmos meméticos.

2.1. Formulação Matemática do PCVLC

Na proposta seminal, Glomvik Rakke et al. [2012] estruturam o PCVLC sobre duas formulações matemáticas, que são posteriormente estendidas para mais três formulações por Battarra et al. [2014]. Este trabalho se baseia nas formulações propostas por Glomvik Rakke et al. [2012], por terem sido a fundamentação principal para o problema.

Formalmente, o PCVLC pode ser definido como um grafo direcionado $G = (V, A)$, incluindo um conjunto de vértices $V = \{1, \cdots, n\}$ para representar os portos (nós do grafo), e um conjunto de arestas $A = \{(i, j) \mid i, j \in V \ \wedge \ i \neq j\}$ que representam as rotas entre os diferentes portos. Cada aresta (i, j) possui um custo c_{ij} associado à distância entre os portos. O vértice 1

representa um depósito – ponto de partida – e os demais portos constituem o conjunto $V\backslash\{1\}$. O depósito possui uma demanda $d_1 = 0$ e cada porto $i \in V\backslash\{1\}$ possui uma demanda inteira não negativa d_i. Ademais, cada porto possui um limite de calado $l_i,\ i \in V\backslash\{1\}$. A carga inicial do navio é dada por $\sum_{j=1}^{n} d_j$, e a cada porto visitado seu valor decresce de acordo com a demanda daquele porto. O navio não pode exceder o limite de calado l_i e, assim, a rota r deve satisfazer $\sum_{j=1}^{n} d_j - \sum_{r=1}^{t-1} d_{i_r} \leq l_{i_t}$ para $t = 2, \cdots, n$.

Considerando-se x_{ij} como uma variável binária cujo valor é 1 se, e somente se, a aresta (i, j) é utilizada na solução ótima, y_{ij} representando a carga no navio na aresta (i, j) e sendo c_{ij} o custo associado à aresta (i, j), então o modelo do PCVLC pode ser descrito por

$$min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

sujeito a

$$\sum_{i\in V} x_{ij} = 1,\ j \in V, \tag{2}$$

$$\sum_{j\in V} x_{ij} = 1,\ i \in V, \tag{3}$$

$$\sum_{i\in V} y_{ij} - \sum_{i\in V} y_{ji} = d_j,\ j \in V\backslash\{1\}, \tag{4}$$

$$\sum_{j\in V} y_{1j} = \sum_{i\in V} d_i, \tag{5}$$

$$\sum_{i\in V} y_{i1} = 0, \tag{6}$$

$$0 \leq y_{ij} \leq l_j x_{ij},\ (i, j) \in A, \tag{7}$$

$$x_{ij} \in \{0, 1\},\ (i, j) \in A. \tag{8}$$

As restrições impostas pelas equações (2) e (3) definem os graus de entrada e saída de cada vértice. A equação de restrição (4) garante que a demanda de cada vértice é satisfeita, impedindo também a criação de subciclos. As restrições implementadas pelas equações (5) e (6) garantem que o navio saia do depósito completamente carregado e retorne a ele vazio. A restrição (7) impõe o limite de calado para cada vértice. Por fim, a restrição (8) corresponde à variável binária explicada anteriormente.

2.2. Algoritmos Meméticos

Proposto por Moscato et al. [1989], o termo memético descreve um método de evolução cultural, baseado na transmissão de conhecimentos por meio da imitação, ou mímica. Essa transmissão ocorre através dos memes (análogos aos genes dos Algoritmos Genéticos), que podem representar ideias, melodias, frases de efeito, moda de roupa, dentre outros conhecimentos que são transmitidos e adaptados às gerações futuras [Dawkins, 2016].

Diferentemente de outras metaheurísticas, como o *Simulated Annealing* e a Busca Tabu, que trabalham com uma solução inicial buscando sua otimização [Talbi, 2009], o AM opera sobre um conjunto de soluções, denominado população. Para um AM, cada indivíduo é denominado um agente e os agentes passam certos períodos envolvidos em buscas independentes, intercalados com períodos em que interagem uns com os outros [Moscato e Norman, 1992].

De modo geral, Moscato et al. [1989] descrevem o funcionamento dos AMs a partir de uma representação do problema de otimização em questão. Dela, um determinado número de indivíduos é criado, de acordo com um procedimento de inicialização, que pode ser uma heurística de construção. Após a geração da população, cada indivíduo realiza uma busca local para atingir um ótimo local ou para melhorar os agentes até determinado nível. Em seguida os indivíduos passam a interagir com os demais, o que pode ser feito tanto de maneira competitiva, semelhante aos métodos de seleção empregados nos Algoritmos Genéticos, quanto cooperativa, que corresponde a troca de informações obtidas pela recombinação entre agentes, tal qual o cruzamento dos AGs, caracterizando assim o processo evolutivo. O processo composto por busca local, competição e cooperação é repetido até atingir um critério de parada. A função *fitness* representa o quão adaptado o agente é para o problema que, na aplicação abordada neste artigo, representa o custo da rota. O Algoritmo 1 sintetiza o comportamento genérico de um AM.

Algoritmo 1: ALGORITMO MEMÉTICO

```
1  início
2      Gerar população inicial
3      Avaliar a fitness de cada indivíduo da população
4      Armazenar a melhor solução
5      repita
6          Selecionar um conjunto de pais
7          Obter nova geração através de recombinação a partir dos pais
8          Executar busca local
9          Atualizar a população com os filhos adequados
10         Atualizar a melhor solução
11     até o critério de parada ser satisfeito;
12 fim
```

3. O Método Implementado

Para o problema a ser explorado nesse artigo, o grafo $G = (V, A)$, modelando a conectividade entre portos e rotas, foi implementado na forma de uma matriz de inteiros $D_{n\times n} = (d_{ij})$, em que n representa o número de portos a serem visitados e cada elemento d_{ij} da matriz armazena a distância entre os vértices i e j. A representação de uma solução é feita na forma de um vetor de inteiros, com cada posição correspondendo a ordem em que cada porto é visitado. O porto de origem ocupa a última posição do vetor, representando o retorno ao vértice inicial. Um exemplo de solução, para uma instância de 10 cidades é $S = \{10, 4, 2, 5, 8, 6, 9, 7, 3, 1\}$ – admitindo que a solução proposta atenda às restrições de limite de calado apresentadas na equação de restrição (4) – em que o porto 1 representa a origem e destino final da rota.

A população de agentes foi implementada na forma de um vetor de soluções, devido à simplicidade de acesso a cada posição, tanto da matriz D, quanto dos vetores de solução. As subseções seguintes detalham as demais implementações, referentes às principais etapas apresentadas no Algoritmo 1.

3.1. Geração de população inicial

Por se tratar de um algoritmo baseado em população, se faz necessário construir um conjunto inicial de agentes. Para isso, Moscato e Cotta [2003] propõe a utilização de um mecanismo de geração aleatória, ou por meio de um mecanismo mais elaborado como, por exemplo, uma heurística de construção.

Para o problema explorado neste artigo, a população inicial foi obtida por meio de uma construção semi-aleatória com características gulosas, visando a geração de soluções com certa qualidade.

No método implementado, ilustrado no Algoritmo 2, dois índices são sorteados aleatoriamente (linhas 5 e 6) para comporem o primeiro e o segundo porto a serem visitados. Nesse sorteio exclui-se o porto 1, que é assumido ser a origem da rota em todos os casos, somente é inserido na última posição do vetor como o destino final. Caso os portos atendam à restrição do limite de calado, são inseridos na rota da solução em questão (linhas 7 e 8) . Os demais portos são escolhidos de maneira gulosa, ou seja, para um porto i, escolhe-se como $i + 1$ o porto com menor distância para i que atenda às restrições (linhas 10 a 17).

Algoritmo 2: CONSTRUÇÃO DE SOLUÇÃO

```
   Entrada: n /* Número de portos da rota */
   Saída: S /* Solução */
 1 início
 2     S ← SolucaoVazia()
 3     s ← 3
 4     repita
 5         i ← indicesAleatoriosPortos(n)
 6         j ← indicesAleatoriosPortos(n)
 7         S.rota[1] ← i
 8         S.rota[2] ← j
 9     até i, j atendem restricao de calado
10     repita
11         para k ∈ [1, n] \ {i, j} faça
12             se distancia(s,k) for a menor e atende restricoes então
13                 S.rota[s] ← k
14                 s ← s + 1
15             fim
16         fim
17     até |S| = n
18 fim
19 retorna S
```

A *fitness* de cada solução é então calculada e armazenada na própria solução ($S.distancia$). O cálculo leva em consideração todos os deslocamentos do navio, desde a saída do porto de origem até o regresso no final do percurso.

3.2. Busca Local

Com o intuito de melhorar o *fitness* dos agentes da população, é aplicado sobre cada um deles uma heurística de refinamento denominada *busca local*. Enquanto as etapas de recombinação e de mutação atuam executando uma busca global em todas as soluções possíveis tentando encontrar o espaço de busca mais promissor, a busca local examina os vizinhos de uma solução inicial [Neri et al., 2011], com o objetivo de encontrar melhores soluções.

Nos AMs, a busca local consiste de um mecanismo por meio do qual os agentes se utilizam do conhecimento a respeito do problema para adaptarem-se a ele, varrendo todas as soluções vizinhas ao agente em questão. É essencialmente a inclusão da busca local que diferencia os Algo-

ritmos Meméticos dos Genéticos.

Algoritmo 3: Busca Local

```
   Entrada: S,n
   Saída: S /*Solução após passar por busca local*/
1  início
2      para i ∈ [1, n) faça
3          para j ∈ [i+1 , n] faça
4              trocar(S_copy.rota[i], S_copy.rota[j])
5              se S_copy é solucao e S_copy.distancia < S.distancia então
6                  S ← S_copy
7              fim
8          fim
9      fim
10 fim
11 retorna S
```

O Algoritmo 3 ilustra a busca local implementada neste artigo. Trata-se de um método de *swap* fixo, em que um porto é trocado com todos os portos subsequentes a ele na rota (linhas 2 a 4). Caso alguma combinação se mostre menor que a solução analisada, esta é substituída pela nova (linha 6). Ao final do procedimento, retorna-se a melhor solução encontrada em todo o processo.

3.3. Seleção de Agentes Para Recombinação

Para que se possa realizar a recombinação dos memes, se faz necessária uma maneira de selecionar agentes adequados para serem os progenitores. De acordo com Talbi [2009], o mecanismo de seleção é um dos principais componentes de pesquisa nos algoritmos evolutivos, e quanto melhor um indivíduo, maiores as chances dele ser selecionado como progenitor. Os algoritmos de seleção mais utilizados são baseados no método da *roleta*, em que se atribui a cada indivíduo um lugar na roleta, proporcional a sua *fitness*, o que dimensiona suas chances de ser sorteado; e o método do *torneio*, em que as *fitness* dos indivíduos são comparadas diretamente.

Algoritmo 4: SELEÇÃO POR TORNEIO

```
   Entrada: tamPop
   Saída: Pai_1, Pai_2
1  início
2      nCandidatos ← numAleatCandidatos(tamPop)
3      listaCandidatos ← sorteiaCandidatos(nCandidatos)
4      enquanto nCandidatos > 2 faça
5          i, j ← sorteiaDoisCandidatos(listaCandidatos)
6          comparaEliminaMaior(i, j)
7          nCandidatos ← nCandidatos - 1
8      fim
9      Pai_1, Pai_2 ← candidatosRestantes(listaCandidatos)
10 fim
11 retorna Pai_1, Pai_2
```

O Algoritmo 4 sintetiza o método do torneio, que foi escolhido para implementação neste trabalho. Inicialmente, um número aleatório de candidatos é selecionado (linha 2). Através de embates diretos, o perdedor (com maior *fitness*) é eliminado da competição e reduz-se o número de participantes a dois (linhas 4 a 7). Estes são então escolhidos como os progenitores (linha 9).

3.4. Recombinação dos Memes

Para realizar a recombinação, implementou-se um operador muito utilizado para cruzamento nos algoritmos genéticos. Trata-se de uma variação do *Order Crossover*, proposto por Davis [1985]. No método em questão constrói-se uma nova solução a partir de um segmento de portos da rota de um dos progenitores. Com base no segmento selecionado, selecionam-se os demais portos do outro progenitor, de modo a não repetir os portos. Dessa maneira, busca-se preservar a ordem relativa de precedência dos portos.

A diferença principal do *Order Crossover* para a solução implementada é que busca-se preencher o $Filho_1$ a partir do final, recebendo assim o último elemento do Pai_2 e preenchendo do fim para o começo, não sobrescrevendo as posições preenchidas pelo segmento copiado do Pai_1 e ignorando portos que já ocupam uma posição na solução. Assim, visa-se respeitar o limite de calado. O mesmo é feito para o $Filho_2$. A Figura 2 ilustra o processo de cruzamento e a origem de uma solução a partir de dois pais para uma rota com 10 portos. Para fins de simplificação, a solução obtida foi considerada como uma solução válida.

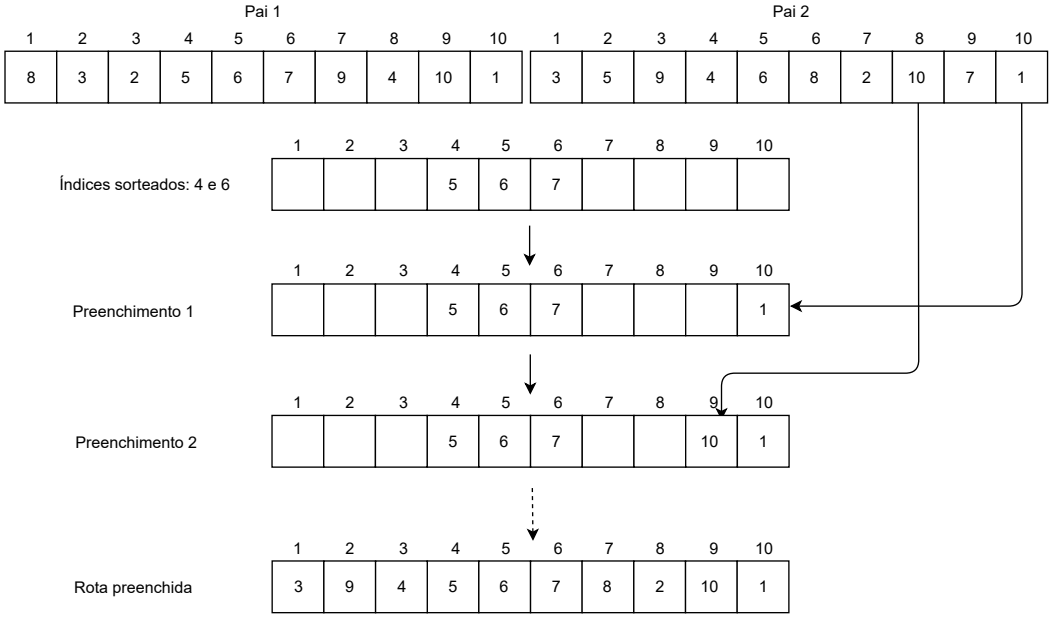


Figura 2: Exemplo de cruzamento de dois indivíduos e consequente geração de um dos filhos.

A escolha por implementar uma variação do *Order Crossover* neste artigo decorre do fato de que os portos com limite de calado mais restritivos normalmente são visitados ao final do percurso, e realizando o preenchimento dessa maneira, o limite não é ferido. Assim, reduz-se a geração de filhos inviáveis por desrespeitar a restrição do calado. Caso sejam gerados filhos inviáveis, estes são descartados e substituídos por novos filhos.

3.5. Etapa de Mutação nos Memes

Para permitir maior variabilidade nos memes dos agentes durante a execução do algoritmo, implementou-se também um operador de mutação (tal qual em um algoritmo genético). De acordo com Talbi [2009], a mutação representa pequenas mudanças nos indivíduos da população. O operador de mutação implementado consiste simplesmente em alterar a ordem de visita de dois

portos sorteados aleatoriamente, com índices entre 1 e n. O procedimento de mutação é ilustrado no Algoritmo 5.

Algoritmo 5: OPERADOR DE MUTAÇÃO

```
Entrada: S, n /*S é o agente a sofrer mutação*/
Saída: S /*O mesmo agente é devolvido com memes alterados*/
início
    repita
        i ← sorteiaIndicesAleatorios(n)
        j ← sorteiaIndicesAleatorios(n)
        trocarIndices(S.rota[i], S.rota[j])
        se S não é solução então
            trocarIndices(S.rota[i], S.rota[j]) /*desfaz movimento inválido*/
        fim
    até S ser solução válida;
fim
retorna S
```

3.6. Reavaliação e Atualização da População

Neste trabalho, considera-se que, a cada geração, os dois piores pais são substituídos pelos dois melhores filhos, melhorando assim a qualidade da população ao longo das gerações. Além disso, sempre que o número de gerações sem melhora for divisível por 4, executa-se o procedimento de busca local, visando melhorar a população como um todo.

Optou-se também verificar, em todas as gerações, se algum dos indivíduos possui menor *fitness* que a melhor solução encontrada até então. Caso positivo, atualiza-se a melhor solução.

Executa-se este procedimento até que o critério de parada estabelecido seja satisfeito. Este, por sua vez, remete a um número máximo de gerações transcorridas sem que se tenha constatado melhora na solução marcada até então como a melhor.

4. Experimentos Computacionais

Os experimentos foram executados em uma máquina com processador Intel® Core™ i5-7200U 2.5 GHz - 3.1 GHz, 8 GB de RAM, sistema operacional Windows 10 Home com subsistema linux Ubuntu 18.04.1. O algoritmo foi implementado na linguagem C (gcc 7.4.0).

O algoritmo proposto foi executado para as instâncias apresentadas em Glomvik Rakke et al. [2012]. As 240 instâncias propostas são adaptações de estruturas de teste contidas na biblioteca TSPLIB [Reinelt, 1991], um conjunto de instâncias utilizado para testes do PCV clássico. Para os experimentos deste trabalho, usou-se os seguintes conjuntos de instâncias: burma14, ulysses16, ulysses22, fri26, bayg29, gr17, gr21 e gr48. Cada conjunto possui 3 subconjuntos, com 10 instâncias cada, os quais apresentam variações do limite de calado em 10%, 25% e 50% dos portos.

Os resultados dos testes, bem como os presentes na literatura, são apresentados nas Tabelas 1 e 2. Por questões de simplificação, optou-se por uma exposição na forma de média simples, agrupando-se conjuntos de 10 instâncias com mesma variação de limite de calado, facilitando assim sumarizar as 240 instâncias. A primeira coluna identifica os subconjuntos de cada instância. A segunda coluna, BCP, apresenta os resultados exatos obtidos por Battarra et al. [2014] com o método *branch-cut-and-price*. A terceira coluna apresenta a implementação da metaheurística GVNS, de Todosijević et al. [2014], que se mostrou capaz de alcançar soluções ótimas para todas as instâncias. Por fim, a coluna AM apresenta os resultados obtidos com a abordagem proposta nesse trabalho.

Tabela 1: Apresentação dos Resultados Agrupados por Grupos com Mesma Variação de Limite de Calado

Instância	BCP		GVNS-2		AM					
	Sol.	T(s)	Sol.	T(s)	Sol. Méd.	T(s)	$\Delta_{Med}(\%)$	Sol. Mel.	T(s)	$\Delta_{Me}(\%)$
bayg29_10	1713,60	11,99	1713,60	0,00	1822,30	0,60	6,34	1737,80	0,61	1,41
bayg29_25	1792,60	11,05	1792,60	0,00	1891,30	0,98	5,51	1806,00	0,95	0,75
bayg29_50	2091,00	13,38	2091,00	0,00	2155,60	0,99	3,09	2101,10	0,91	0,48
burma14_10	3386,70	0,52	3386,70	0,00	3388,80	0,03	0,06	3386,70	0,02	0,00
burma14_25	3596,80	0,37	3596,80	0,00	3598,90	0,03	0,06	3596,80	0,02	0,00
burma14_50	3862,30	0,35	3862,30	0,00	3862,90	0,02	0,02	3862,30	0,02	0,00
fri26_10	963,80	12,37	963,80	0,00	1002,50	0,34	4,02	963,80	0,35	0,00
fri26_25	1104,70	16,43	1104,70	0,00	1143,00	0,37	3,47	1107,30	0,39	0,24
fri26_50	1178,70	16,26	1178,70	0,00	1203,50	0,36	2,10	1178,70	0,35	0,00
gr17_10	2150,30	25,12	2150,30	0,00	2158,60	0,06	0,39	2150,30	0,07	0,00
gr17_25	2237,70	11,05	2237,70	0,00	2248,20	0,07	0,47	2237,70	0,06	0,00
gr17_50	2710,30	5,54	2710,30	0,00	2716,80	0,06	0,24	2710,30	0,04	0,00
gr21_10	2833,60	11,63	2833,60	0,00	2926,90	0,18	3,29	2833,60	0,20	0,00
gr21_25	2962,60	11,44	2962,60	0,00	3049,60	0,20	2,94	2962,60	0,22	0,00
gr21_50	3738,10	10,32	3738,10	0,00	3793,20	0,21	1,47	3738,10	0,19	0,00
ulysses16_10	6868,20	50,34	6868,20	0,00	6879,20	0,05	0,16	6868,20	0,04	0,00
ulysses16_25	7165,40	18,58	7165,40	0,00	7183,30	0,05	0,25	7165,40	0,06	0,00
ulysses16_50	7590,30	4,60	7590,30	0,00	7594,30	0,05	0,05	7590,30	0,05	0,00
ulysses22_10	7087,60	32,02	7087,60	0,00	7143,90	0,19	0,79	7087,60	0,20	0,00
ulysses22_25	7508,70	24,61	7508,70	0,00	7611,30	0,23	1,37	7508,70	0,22	0,00
ulysses22_50	8425,60	26,55	8425,60	0,00	8511,90	0,28	1,02	8425,60	0,29	0,00

Ressalta-se que, apesar de existirem outras abordagens na literatura, optou-se por não compará-las por não alcançarem resultados ótimos para todas as instâncias. Ademais, como as condições dos experimentos da literatura não puderam ser reproduzidas na sua íntegra (como recursos de hardware, por exemplo), então o tempo de execução não deve ser levado em consideração para avaliação dos resultados desse artigo. Dessa forma, as tabelas identificam apenas os tempos de cada método, tal como reportados na literatura.

Na Tabela 1 apresentam-se os resultados obtidos para cada subgrupo de 10 instâncias. A coluna *AM-Sol. Méd.* apresenta as médias relativas a 100 execuções para cada instância, enquanto a coluna *AM-Sol. Mel.* reporta as melhores soluções encontradas. As colunas Δ_{Med}**%** e Δ_{Me}**%** apresentam o percentual de distância da média das execuções e o erro percentual entre as melhores soluções, encontradas dentre todas as execuções, ambas em relação à solução ótima.

Como é possível observar nas tabelas, a solução proposta é bastante competitiva, atingindo a solução ótima para boa parte das instâncias testadas. A média da coluna $\Delta_{Med}\%$ na Tabela 1 é de $1,77$ %, o que indica a estabilidade do método ao se aproximar do ótimo global. Entretanto, para as instâncias do grupo gr48 não foi possível obter soluções ótimas. Acredita-se que o motivo deve-se à utilização de somente uma estrutura de vizinhança empregada na etapa de busca local. A Tabela 2 apresenta os resultados obtidos em 100 execuções do AM para cada instância, bem como os resultados de cada método da literatura.

5. Conclusões

O Problema do Caixeiro Viajante com Limite de Calado é um problema de otimização com aplicação direta no comércio e transporte marítimo. Como é possível reduzi-lo ao Problema do Caixeiro Viajante Assimétrico, trata-se de um problema NP-difícil que, por natureza, demanda de tempo exponencial para ser resolvido de maneira ótima.

Tabela 2: Resultados Detalhados da Instância gr48

Instância	BCP		GVNS-2		AM					
	Sol.	T(s)	Sol.	T(s)	Sol. Méd.	$\Delta_{Md.}(\%)$	T(s)	Sol. Mel.	$\Delta_{Mel.}(\%)$	T(s)
gr48_10_1	5046	813,29	5046	0,00	5583	10,64	4,40	5253	4,10	4,38
gr48_10_2	5542	341,27	5542	0,10	6090	9,89	4,94	5661	2,15	4,97
gr48_10_3	5300	3314,02	5300	0,06	5826	9,92	4,98	5543	4,58	4,94
gr48_10_4	5293	7057,36	5293	0,45	5588	5,57	4,30	5424	2,47	4,30
gr48_10_5	5679	496,29	5679	30,27	6244	9,95	4,65	5962	4,98	4,64
gr48_10_6	5610	41,04	5610	0,42	5910	5,35	4,07	5800	3,39	4,09
gr48_10_7	5063	22,79	5063	1,20	5693	12,44	4,27	5386	6,38	4,30
gr48_10_8	5103	188,77	5103	1,15	5451	6,82	4,30	5378	5,39	4,27
gr48_10_9	5153	892,87	5153	0,17	5597	8,62	3,61	5427	5,32	3,58
gr48_10_10	5055	359,47	5055	0,01	5704	12,84	4,63	5433	7,48	4,61
gr48_25_1	5524	1073,43	5524	0,47	5951	7,73	3,74	5845	5,81	3,75
gr48_25_2	5895	361,90	5895	0,03	6558	11,25	12,34	6302	6,90	12,30
gr48_25_3	5754	47,85	5754	0,56	6196	7,68	4,37	5974	3,82	4,38
gr48_25_4	5588	126,05	5588	2,34	6245	11,76	4,98	5923	5,99	4,99
gr48_25_5	6159	1793,96	6159	4,55	6687	8,57	3,68	6441	4,58	3,67
gr48_25_6	5760	3053,47	5760	8,49	6108	6,04	4,18	5973	3,70	4,16
gr48_25_7	5955	172,08	5955	41,29	6214	4,35	3,10	6003	0,81	3,11
gr48_25_8	5562	1070,86	5562	0,43	6065	9,04	8,94	5849	5,16	8,92
gr48_25_9	5792	184,24	5792	2,46	6496	12,15	4,73	6196	6,98	4,75
gr48_25_10	6014	730,82	6014	5,08	6694	11,31	5,20	6253	3,97	5,18
gr48_50_1	6096	157,47	6096	0,64	6745	10,65	8,43	6437	5,59	8,44
gr48_50_2	6629	21,44	6629	12,30	7456	12,48	6,64	7161	8,03	6,65
gr48_50_3	5896	132,45	5896	13,79	6454	9,46	4,89	6274	6,41	4,90
gr48_50_4	6404	20,75	6404	0,49	6974	8,90	5,26	6702	4,65	5,24
gr48_50_5	6617	19,41	6617	0,03	7195	8,74	5,83	6643	0,39	5,83
gr48_50_6	8533	66,09	8533	0,48	9120	6,88	4,86	8601	0,80	4,85
gr48_50_7	6166	1520,10	6166	19,59	6913	12,11	7,14	6626	7,46	7,12
gr48_50_8	6535	20,93	6535	27,76	6986	6,90	3,96	6717	2,79	3,94
gr48_50_9	7150	42,73	7150	7,92	7959	11,31	4,98	7453	4,24	5,01
gr48_50_10	6331	112,13	6331	0,04	6816	7,66	4,17	6615	4,49	4,17

Neste artigo, propôs-se a implementação de um Algoritmo Memético para resolver o PC-VLC. Os resultados obtidos mostraram-se capazes de competir com os encontrados na literatura, mesmo com algumas instâncias não atingindo o valor ótimo. Além disso, apresentou-se uma nova abordagem, sob o ponto de vista dos algoritmos evolutivos, para o problema em questão. Utilizando-se de somente uma estrutura de vizinhança na etapa de busca local, a abordagem proposta nesse trabalho mostrou-se capaz de alcançar valores ótimos para cerca de $79,2\%$ das instâncias testadas.

Para trabalhos futuros, estuda-se a possibilidade de utilização de mecanismos de ordenação, para melhorar a política de atualização de gerações. Além disso, objetiva-se a implementação de outras políticas de busca local, para aumentar o número de vizinhanças a serem exploradas.

Referências

Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

Baldacci, R., Mingozzi, A., e Roberti, R. (2012). New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 24(3):356–371.

Battarra, M., Erdoğan, G., Laporte, G., e Vigo, D. (2010). The traveling salesman problem with pickups, deliveries, and handling costs. *Transportation Science*, 44(3):383–399.

Battarra, M., Pessoa, A. A., Subramanian, A., e Uchoa, E. (2014). Exact algorithms for the traveling salesman problem with draft limits. *European Journal of Operational Research*, 235(1):115–128.

Davis, L. (1985). Applying adaptive algorithms to epistatic domains. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*, p. 162–164, San Francisco, CA. Morgan Kaufmann Publishers Inc.

Dawkins, R. (2016). *The selfish gene*. Oxford university press.

Garey, M. R. e Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman Co., USA. ISBN 0716710447.

Glomvik Rakke, J., Christiansen, M., Fagerholt, K., e Laporte, G. (2012). The traveling salesman problem with draft limits. *Comput. Oper. Res.*, 39(9):2161–2167. ISSN 0305-0548.

Lângaro, R. R. e Barbosa, M. A. (2019). Metaheurística colônia de formigas híbrida aplicada ao problema do caixeiro viajante com limite de calado. In *Anais do LI Simpósio Brasileiro de Pesquisa Operacional*, Limeira, SP. SOBRAPO.

Machado, V. M. C., Ochi, L. S., e Neves, T. A. (2015). Grasp para o problema do caixeiro viajante com limite de calado. In Bastos Filho, C. J. A., Pozo, A. R., e Lopes, H. S., editors, *Anais do 12 Congresso Brasileiro de Inteligência Computacional*, p. 1–6, Curitiba, PR. ABRICOM.

Morais, W. B., Rosa, M., Teixeira, M., e Barbosa, M. A. (2017). O problema do caixeiro viajante com limite de calado: uma abordagem usando simulated annealing. In *Anais do XLIX Simpósio Brasileiro de Pesquisa Operacional*, p. 2029–2040, Blumenau, SC. SOBRAPO.

Moscato, P. e Cotta, C. (2003). A gentle introduction to memetic algorithms. In *Handbook of metaheuristics*, p. 105–144. Springer.

Moscato, P. et al. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989.

Moscato, P. e Norman, M. G. (1992). A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. *Parallel computing and transputer applications*, 1:177–186.

Neri, F., Cotta, C., e Moscato, P. (2011). *Handbook of Memetic Algorithms*. Springer Publishing Company, Incorporated. ISBN 3642232469.

Papadimitriou, C. H. e Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., USA. ISBN 0131524623.

Reinelt, G. (1991). Tspblib - a traveling salesman problem library. In *INFORMS Journal on Computing*, p. 376–384.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. Wiley Publishing.

Todosijević, R., Mjirda, A., Mladenović, M., Hanafi, S., e Gendron, B. (2014). *A general variable neighborhood search variants for the travelling salesman problem with draft limits*. *Optimization Letters*, p. 1–10. ISSN 1862-4480.