O Problema do Caixeiro Viajante com Limite de Calado: uma Abordagem usando *Simulated Annealing*

Wall Berg Morais, Marcelo Rosa, Marcelo Teixeira, Marco Antonio Barbosa
Department of Informatics of the Federal University of Technology – Paraná – Brazil
Via do Conhecimento, Km 01 – 85503-390
{wall, marcelorosa}@alunos.utfpr.edu.br,
{marceloteixeira,mbarbosa}@utfpr.edu.br

RESUMO

Este artigo reporta uma abordagem de solução para o *Problema do Caixeiro Viajante com Limite de Calado* (PCVLC) usando a metaheurística *Simulated Annealing*. O PCVLC é uma variação do *Problema do Caixeiro Viajante* (PCV) aplicado a problemas de transporte marítimo. O calado de um navio é a distância entre a base inferior da embarcação até o nível da água. Muitos portos possuem restrições relacionadas ao limite de calado permitido para que uma embarcação possa adentrar no porto de modo seguro. Este problema pode ser reduzido ao *Problema do Caixeiro Viajante Assimétrico* e por conta disto é, também, um problema NP-Difícil com grande interesse prático por soluções. Neste artigo são apresentados resultados obtidos a partir do uso da metaheurística *Simulated Annealing*, os quais foram confrontados com os principais resultados encontrados no estado da arte sobre o problema. A análise comparativa mostra que a abordagem proposta é uma alternativa competitiva quando comparada a outras metaheurísticas.

PALAVRAS CHAVE. Simulated Annealing. Metaheurística. otimização.

Tópicos: MH - Metaheurísticas, OC - Otimização Combinatória

ABSTRACT

This paper reports a solution approach to the *Traveling Salesman Problem with Draft Limit* (TSPDL) using a *Simulated Annealing* metaheuristic. TSPDL is a variation of the *Traveling Salesman Problem* (TSP) applied to maritime transportation. The draft of a ship is a distance from a lower base of the vessel to the water level. Many ports have restrictions related to the allowable draft limit so that a vessel can enter the port safely. TSPDL can be reduced to the *Asymmetric Traveling Salesman Problem* so it is also a NP-Hard problem with great practical interest for solutions. This paper presents results obtained by using the metaheuristic *Simulated Annealing*, which have been confronted against the main results found in the state of the art about the problem. The comparative analysis shows that the proposed approach is a competitive alternative when compared to other metaheuristics.

KEYWORDS. Simulated Annealing. Metaheuristic. Optimization.

Paper topics: MH - Metaheuristics, OC - Combinatorial Optimization

1. Introdução

O Problema do Caixeiro Viajante (PCV) é um dos problemas mais investigados na área de otimização combinatória. Sua formulação relativamente simples, porém, de solução complexa, o torna um importante objeto de estudo em função de sua aplicação teórica e prática [Cook, 2011].

As aplicações práticas do PCV são as mais variadas e em Glomvik Rakke et al. [2012] foi apresentada uma nova formulação do problema aplicado a transportes marítimos de cargas, o denominado *Problema do Caixeiro Viajante com Limite de Calado* (PCVLC).

O calado de um navio é a distância entre a base inferior do navio até o nível da água. O navio, após ser carregado, tende aumentar o tamanho do calado devido ao acréscimo do peso de suas cargas. Um problema frequente no transporte marítimo é que, muitos portos não suportam um navio que possua um calado superior ao limite do porto. Desta forma, um navio que não respeita esta restrição pode vir a ficar encalhado ao tentar atracar neste porto. Em função desta restrição, o objetivo do PCVLC é fazer com que o navio percorra uma rota de menor distância, saindo do porto de origem, passando por todos os portos uma única vez e retornado ao porto de origem (circuito Hamiltoniano de custo mínimo), porém, obedecendo o limite do calado de cada porto. A Figura 1, apresentada em Machado et al. [2015], ilustra o calado para um navio quando este está descarregado e carregado, respectivamente.

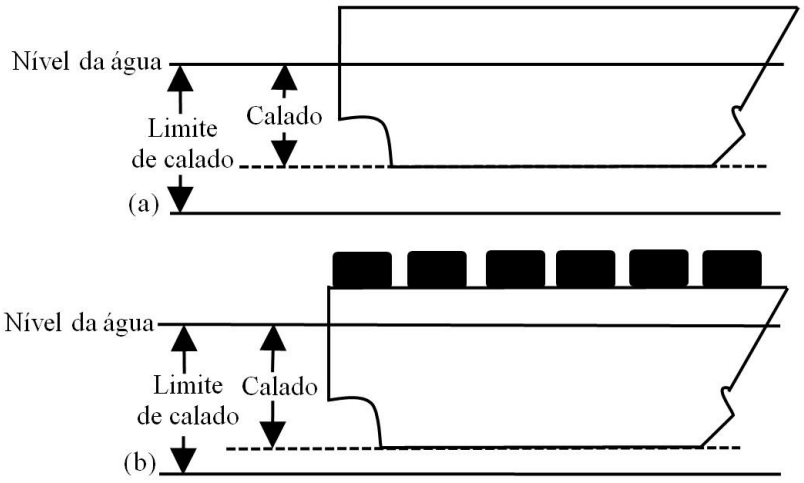

Figura 1: (a) Navio sem carga (b) Navio com carga. Fonte: Machado et al. [2015]

Considerando-se que o PCVLC pode ser reduzido ao Problema do Caixeiro Viajante Assimétrico, este também é, portanto, um problema da classe NP-Difícil [Glomvik Rakke et al., 2012].

Ao apresentar o problema, Glomvik Rakke et al. [2012] apresentaram duas formulações matemáticas para o PCVLC e utilizaram o método *branch-and-cut* para fornecer soluções exatas para o problema. Em Battarra et al. [2014] três novas formulações matemáticas para o PCVLC foram apresentadas e os autores também fizeram uso do método *branch-and-cut* para obterem soluções exatas para o problema e puderam aplicar estas soluções a instâncias maiores.

Em Todosijević et al. [2014] os autores apresentaram duas soluções metaheurísticas para o problema, as quais constituem-se de variações da metaheurística VNS

(*Variable Neighborhood Search*). Com as soluções metaheurísticas propostas pelos autores, os mesmos foram capazes de resolver instâncias maiores e para a realização de seus experimentos os autores propuseram um novo conjunto de instâncias de teste.

Machado et al. [2015] apresentaram uma solução para o PCVLC utilizando a metaheuristica GRASP (*Greedy Randomized Adaptive Search Procedure*).

De acordo com o conhecimento dos autores deste artigo, até o presente momento, este seria o estado da arte para o PCVLC. Considerando-se o interesse do problema e as poucas alternativas de solução encontradas na literatura, neste artigo é apresentada uma solução alternativa ao PCVLD utilizando-se a metaheurística *Simulated Annealing* (SA).

O SA faz analogia a um processo físico utilizado na confecção de cristais. Neste processo, um cristal é moldado em altas temperaturas e ao passar por um processo de resfriamento lento chega a um estado sólido inquebrável. Por outro lado, ao ser submetido a resfriamento por queda abrupta de temperatura, o cristal acaba sendo um material frágil [Johnson et al., 1989]. Computacionalmente, o SA tem um procedimento de controle de temperatura que, dependendo da velocidade de sua variação, estabelece a qualidade final da solução obtida.

Neste artigo, foram realizados experimentos computacionais e estes foram comparados com as soluções atualmente encontradas na literatura. O resultados mostram que o SA pode ser uma alternativa competitiva e que apresenta bons resultados. Para a maioria das instâncias testadas o SA conseguiu atingir a solução ótima. Para as instâncias maiores quando o SA não conseguiu atingir as soluções ótimas, apresentou uma boa taxa de aproximação.

O presente artigo está estruturado como segue. Na seção 2 são apresentados como conceitos básicos os modelos matemáticos para o PCVLC e a metaheurística Simulated Annealing. A seção 3 detalha a implementação da solução proposta neste artigo. Na seção 4 são apresentados os resultados e algumas discussões parciais. Na seção 5 são apresentadas as conclusões deste trabalho.

2. Conceitos Básicos

Nesta seção é apresentada a formulação matemática originalmente proposta em Glomvik Rakke et al. [2012]. Atualmente o PCVLC apresenta cinco formulações matemáticas. Duas destas formulações são devidas a Glomvik Rakke et al. [2012] e três outras formulações para o problema são apresentadas em Battarra et al. [2014]. Além do modelo matemático, esta seção apresenta ainda uma breve introdução à metaheurística Simulated Annealing [Kirkpatrick et al., 1983; Cerny, 1985]. Neste trabalho, optou-se pela formulação apresentada em Glomvik Rakke et al. [2012] por ser a mais relaxada e por ser a que deu origem ao problema.

2.1. Formulação Matemática do PCVLC

Formalmente, o PCVLC pode ser definido como um grafo direcionado $G = (V, A)$, onde $V = \{0, 1, \ldots, n\}$ é o conjunto de vértices (nós), representando os portos, e $A = \{(i, j) \mid i, j \in V \text{ e } i \neq j\}$ é o conjunto de arestas, modelando as conexões entre os portos. O porto de origem é representado pelo nó 0. Cada aresta (i, j) possui um custo c_{ij} associado. Os demais nós do conjunto $V \setminus \{0\}$ representam os portos que devem ser visitados. Cada porto, exceto o de origem, possui uma demanda de carga denotada por $d_i, i \in V \setminus \{0\}$ e possui, também, um limite de calado $l_i, i \in V \setminus \{0\}$. A carga inicial do navio é denotada por $Q = \sum_{i \in V \setminus \{0\}} d_i$ e seu valor decresce a cada porto visitado. O navio não pode entrar no porto i se estiver com a carga atual maior do que o limite de calado l_i do porto sob o risco de encalhar a embarcação.

A formulação matemática introduzida a seguir leva a uma solução para o PCVLC. Nela, considera-se o uso da variável binária x_{ij}, que recebe o valor 1 se a aresta (i, j) é utilizada na solução e 0 em caso contrário. A variável y_{ij} denota a carga do navio na aresta (i, j). O modelo é então definido como:

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

sujeito a :

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

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

$$\sum_{i\in V} y_{ij} - \sum_{i\in V} y_{ji} = d_j,\ \forall j \in V\backslash\{0\} \tag{4}$$

$$\sum_{i\in V} y_{0i} = \sum_{i\in V\backslash\{0\}} d_i \tag{5}$$

$$\sum_{i\in V} y_{i0} = 0 \tag{6}$$

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

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

As restrições 2 e 3 estabelecem os graus de entrada e de saída de cada vértice, assegurando que cada porto seja visitado uma única vez. As restrições em 4 asseguram que a demanda de cada porto seja satisfeita e impede a criação de subciclos. As restrições 5 e 6 asseguram que o navio saia da origem totalmente carregado e que retorne para a origem totalmente descarregado. As restrições em 7 impõem que o navio respeite o limite do calado dos portos a serem visitados. Por fim, as restrições em 8 impõem que, de forma binária, uma aresta esteja ou não presente em uma solução.

2.2. Simulated Annealing

A metaheurística Simulated Annealing aplicada a problemas de otimização combinatória surge dos trabalhos pioneiros de Kirkpatrick et al. [1983], aplicado ao particionamento de grafos, e Cerny [1985] aplicado ao desenho de circuitos VLSI. Devido a sua simplicidade e eficiência o SA é atualmente largamente utilizado nos mais diversos problemas de otimização [Talbi, 2009].

O SA baseia-se nos princípios da mecânica estatística nos quais o processo de recozimento de materiais, como por exemplo o metal, requer o aquecimento do material a altas temperaturas e, depois, um arrefecimento lento para se obter uma estrutura cristalina forte [Talbi, 2009]. A solidez de uma estrutura está relacionada à taxa de redução da temperatura. Quando a temperatura decresce rapidamente, de forma abrupta, o metal irá apresentar uma estrutura fraca e imperfeita. Ao ser submetido a um processo lento de resfriamento, o metal apresenta uma estrutura bastante sólida em função do melhor ajuste molecular, ou seja, menos espaçamento entre suas moléculas, o que torna o material mais sólido e resistente.

O pseudo código apresentado no algoritmo 1 ilustra o funcionamento da metaheurística SA. O algoritmo parte de uma solução inicial e, sobre esta, realiza um procedimento de refinamento (busca local). Esta etapa está descrita na linha 1. Uma alternativa muito utilizada para a obtenção da solução inicial é a criação de uma solução puramente aleatória [Kirkpatrick et al., 1983]. A qualidade da solução obtida pelo SA depende, sobretudo, da temperatura inicial (linha 2). Uma temperatura muita alta, associada a um esquema de resfriamento muito lento, pode conduzir a excelentes soluções porém, com um tempo de CPU bastante elevado. Por outro lado, uma temperatura demasiadamente baixa tende a consumir menos tempo de CPU porém, pode gerar soluções bastante pobres. Além disso, sob altas temperaturas a busca no espaço de soluções é praticamente randômica, enquanto que sob temperaturas muito baixas a busca torna-se praticamente *gulosa*, ou seja, apenas os melhores vizinhos são escolhidos [Youssef et al., 2001]. Portanto, o parâmetro de temperatura inicial é, por si só, um objeto de experimentações para escolha da melhor configuração inicial.

Algoritmo 1: O Algoritmo Simulated Annealing

Entrada: Esquema de resfriamento
Saída: Melhor solução encontrada

```
1  s ← s_0 /* Geração da solução inicial */
2  T ← T_max /* Temperatura inicial*/
3  repita
4      repita
5          Escolher um vizinho aleatório s' ∈ N(S)
6          ΔE ← f(s') − f(s)
7          se ΔE < 0 então
8              s ← s' /* Aceita a solução vizinha */
9          fim
10         senão
11             Aceita s' com probabilidade e^(−ΔE/T)
12         fim
13     até Condição de equilibrio;
14     /* por exemplo, um dado número de iterações executadas para cada
          temperatura T */
15     T ← g(T)
16 até critério de parada /*e.g. T < T_min*/;
```

Após as inicializações de solução e temperatura, o algoritmo inicia um procedimento de busca estocástica no seu espaço de soluções S (linha 5). Isto é, um vizinho $s' \in N(S)$ é escolhido aleatoriamente do espaço de soluções. Na linha 6 é calculado um coeficiente ΔE que mede a qualidade da solução vizinha em relação à solução atual, com base na função objetivo f. O coeficiente ΔE é o resultado da diferença entre o custo da solução vizinha $f(s')$ e o custo da solução atual $f(s)$. Os possíveis domínios que ΔE pode assumir correspondem a um valor negativo, positivo ou igual a 0. Um valor negativo ($\Delta E < 0$) indica que a solução vizinha é melhor que a solução atual. Um valor positivo ($\Delta E > 0$) indica que a solução vizinha é pior que a solução atual. Por fim, $\Delta E = 0$, indica que as soluções têm os mesmos resultados. É importante destacar que o algoritmo

1 está voltado à solução de um problema de minimização. Um problema de maximização, diferentemente, seria calculado como $\Delta E \leftarrow f(s) - f(s')$.

Se a solução vizinha s' for uma solução melhor que solução atual (linha 7) então s' passa a ser a solução corrente (linha 8). Caso a solução s' seja pior que a solução atual, então um fator probabilístico, calculado na linha 11, irá determinar se a solução vizinha s' será ou não aceita como solução atual. Essa escolha por uma eventual pior solução é o mecanismo utilizado pelo SA como tentativa de escapar de ótimos locais. A iteração entre as linhas 4-13 ocorre até que uma situação de equilíbrio seja observada. Por exemplo, uma possível condição de equilíbrio poderia ser limitar um número máximo de vezes que o algoritmo irá admitir uma solução vizinha pior que a atual. Na linha 15 a temperatura T é atualizada de acordo com o esquema de resfriamento definido na função $g(T)$. E, por fim, a linha 3 define a condição de parada. Uma possível condição de parada pode ser o decréscimo da temperatura T até atingir uma temperatura mínima T_{min} definida *a priori*.

3. A Solução Proposta

Pode-se considerar que a solução para o PCVLC, conforme concebida neste trabalho, é uma solução híbrida, no sentido de que ela combina a fase de construção da metaheurística GRASP com a metaheurísticas SA, que é usada como heurística de refinamento. Tal combinação é descrita no algoritmo 3 e uma análise dos resultados obtidos a partir dessa abordagem é apresentada na Seção 4.

3.1. Construção da Solução Inicial

Após a realização de diversos testes, aptou-se pelo uso de uma heurística construtiva em detrimento de um procedimento puramente aleatório para a geração da solução inicial. Testes mostraram que soluções puramente aleatória geravam os piores resultados. O uso de uma heurística puramente gulosa também não gerou resultados satisfatórios. Por conta disso, a solução aqui proposta utiliza, para a geração da solução inicial, um método guloso aleatorizado. Este método é usualmente adotado pela metaheurística GRASP [Feo e Resende, 1995] em sua fase de construção da solução inicial. O GRASP caracteriza-se, essencialmente, como uma metaheurística de duas etapas: (a) a fase de *construção*; e (b) a fase de *busca local* (ou *refinamento*). O algoritmo 2 detalha o procedimento para a escolha da solução inicial utilizada neste trabalho e que corresponde à fase de construção do GRASP.

Algoritmo 2: CONSTRUTIVO_GRASP

Entrada: $\alpha, G = (V, A), n$
Saída: S /*uma solução viável ou $\{\}$ */
1 Inicialize a LC
2 $S \leftarrow \emptyset$
3 **enquanto** $LC \neq \emptyset$ **faça**
4 criar a LCR
5 $x \leftarrow aleatorio(LCR)$
6 $S \leftarrow S \cup \{x\}$
7 $LC \leftarrow LC \backslash \{x\}$
8 **fim**
9 **retorna** S

A *Lista de Candidatos* (LC) contém os portos ainda não visitados. A solução S é inicializada como um conjunto vazio. A cada iteração do algoritmo um porto é escolhido

para compor o conjunto S e fazer parte da solução inicial. Para limitar a escolha dos candidatos, entre aqueles que são considerados os melhores, é criada uma *Lista Restrita de Candidatos* (LRC). Esta lista é criada ordenando-se os portos candidatos, em função das distâncias entre o último porto visitado e os portos candidatos à solução. Somente farão parte da LRC os candidatos à solução que não violem às restrições de limite de calado e que não ultrapassem o tamanho máximo da lista, o que é definido pelo parâmetro de entrada α. Após escolher um porto, a LRC será reconstruída para a escolha do próximo candidato à solução. O processo prossegue até que todos os portos estejam presentes na solução S ou que se verifique que, para a instância testada, não há uma solução viável.

3.2. O Algoritmo Simulated Annealing

O algoritmo recebe como parâmetros de entrada uma instância do problema, representada na forma de um grafo $G = (V, A)$, a quantidade n de portos a serem visitados, um coeficiente α para limitar o tamanho da lista de candidatos à solução a serem escolhidos, T_{ini}, que indica a temperatura inicial (linha 1), T_{min}, que denota a temperatura mínima a ser utilizada como critério de parada do algoritmo, $equilibrio$ determina a condição de equilíbrio do algoritmo (número de iterações sob uma mesma temperatura) e $converge$ determina o número iterações em que não houve melhora na solução.

Como apresentado na seção 2, o SA necessita de uma solução inicial. Esta solução é gerada na linha 2 e esta passa a ser a melhor solução, representada por S^* na linha 3.

A iteração da linha 4 controla o decréscimo da temperatura e o acréscimo do tempo. Este acréscimo é definido pelo coeficiente β (linha 27). A condição de parada deste laço de repetição é definida pelo parâmetro que estabelece a temperatura mínima T_{min}. O laço de repetição da linha 6, estabelece um número máximo ($equilibrio$) de vezes que uma solução pior que a solução corrente será aceita e o $converge$ estabelece o número máximo de iterações que o algoritmo irá executar sem que haja melhora na solução.

Na linha 7 é realizada a escolha aleatória de um vizinho s' de S. Se o custo de s' for menor que o custo da solução atual S, então s' passa a ser a solução atual. Caso esta solução s' seja a melhor das soluções encontradas até o momento, como descrito na linha 11, então a solução ótima local S^* é atualizada. Por fim, considerando-se que a solução vizinha s' seja uma solução pior do que a solução corrente, calcula-se a a probabilidade de aceitação desta solução (linha 16). Quando nenhumas das condições acima são satisfeitas, a variável $semMudanca$ é atualizada, como descrito na linha 22. Isto ocorre para que o algoritmo não execute muitas iterações sem melhora da solução sob uma mesma temperatura. Por fim, o algoritmo retorna como saída o custo da melhor rota.

Algoritmo 3: SIMULATED ANNEALING

Entrada: $G = (V, A), n, \alpha, T_{ini}, T_{min}, equilibrio, converge$
Saída: Custo da menor rota encontradaRota

$T \leftarrow T_{ini}$ /*Inicializa a temperatura inicial */
$S \leftarrow contrutivo\_GRASP(G, n, \alpha)$ /*Gera solução inicial */
$S^* \leftarrow S$ /*atualiza a melhor solução corrente*/
enquanto $T > T_{min}$ **faça**
  $piora \leftarrow 0$ /*número soluções piores aceitas */
  enquanto $piora \leq equilibrio \wedge semMudanca \leq converge$ **faça**
    $s' \leftarrow escolhe\_vizinho(G, S, n)$
    se $custo\_rota(s') < custo\_rota(S)$ **então**
      $S \leftarrow s'$
      $semMudanca \leftarrow 0$
      se $custo\_rota(s') < custo\_rota(S^*)$ **então**
        $S^* \leftarrow s'$
      fim
    fim
    senão
      se $aleatorio[0, 1) < e^{\frac{custo\_rota(S) - custo\_rota(s')}{T}}$ **então**
        $S \leftarrow s'$
        $piora \leftarrow piora + 1$
        $semMudanca \leftarrow 0$
      fim
      senão
        $semMudanca \leftarrow semMudanca + 1$
      fim
    fim
  fim
  $T \leftarrow T_{ini} * e^{-t/P_2}$
  $t \leftarrow \beta + t$
fim
retorna $custo\_rota(S^*)$

4. Experimentos Computacionais

O algoritmo proposto neste trabalho foi executado para todas as instâncias apresentadas em Glomvik Rakke et al. [2012]. São ao todo 240 instâncias de teste com a quantidade de portos variando entre 14 a 48. Essas estruturas são adaptações de oito instâncias clássicas disponibilizadas na biblioteca TSPLIB [Reinelt, 1991], são elas: *burma14*, *ulysses16*, *ulysses22*, *fri26*, *bayg29*, *gr17*, *gr21* e *gr48*. As instâncias usadas contém uma matriz de distâncias entre os portos bem como, a demanda e o limite de calado de cada porto. Cada grupo de instâncias é composto por 30 ocorrências divididas em 3 subgrupos de 10. Cada subgrupo apresenta variações do limite de calado em 10%, 25% e 50% dos portos, como ilustrado na Tabela 1. A Tabela 2 ilustra com mais detalhes estas subdivisões.

Tabela 1: Média dos Melhores Resultados

	BCP		GVNS-1		GVNS-2		GRASP		SA	
Instância	Sol.	T(s)	Sol.	T(s)	Sol.	T(s)	Sol.	T(s)	Sol.	T(s)
burma14_10	3386,70	0,52	3386,70	0,00	3386,70	0,00	3386,70	0,01	3386,70	0,44
burma14_25	3596,80	0,37	3596,80	0,00	3596,80	0,00	3596,80	0,00	3596,80	0,27
burma14_50	3862,30	0,35	3862,30	0,00	3862,30	0,00	3862,00	0,00	3862,30	0,19
Ulysse16_10	6868,20	50,34	6868,20	0,00	6868,20	0,00	6868,20	0,02	6868,20	0,56
Ulysse16_25	7165,40	18,58	7165,40	0,00	7165,40	0,00	7165,40	0,02	7165,40	0,31
Ulysse16_50	7590,30	4,60	7590,30	0,00	7590,30	0,00	7590,30	0,02	7590,30	0,22
Ulysse22_10	7087,60	32,02	7087,60	0,04	7087,60	0,00	7087,60	0,00	7087,60	0,55
Ulysse22_25	7508,70	24,61	7508,70	0,01	7508,70	0,00	7508,70	0,00	7508,70	0,34
Ulysse22_50	8425,60	26,55	8425,60	0,04	8425,60	0,03	8425,60	0,00	8425,60	0,26
fri26_10	963,80	12,37	963,80	0,02	963,80	0,00	963,80	0,04	963,80	0,54
fri26_25	1104,70	16,43	1104,70	0,01	1104,70	0,00	1104,70	0,04	1104,70	0,34
fri26_50	1178,70	16,26	1178,70	0,01	1178,70	0,01	1178,70	0,05	1178,70	0,27
bayg29_10	1713,60	11,99	1713,60	0,00	1713,60	0,02	1713,60	0,06	1713,60	0,60
bayg29_25	1792,60	11,05	1792,60	0,14	1792,60	0,02	1792,60	0,07	1792,60	0,38
bayg29_50	2091,00	13,38	2091,00	0,01	2091,00	0,05	2091,00	0,08	2091,00	0,29
gr17_10	2150,30	25,12	2150,30	0,00	2150,30	0,02	2150,30	0,02	2150,30	0,41
gr17_25	2237,70	11,05	2237,70	0,14	2237,70	0,02	2237,70	0,02	2237,70	0,27
gr17_50	2710,30	5,54	2710,30	0,01	2710,30	0,05	2710,30	0,02	2710,30	0,20
gr21_10	2833,60	11,63	2833,60	0,00	2833,60	0,01	2833,60	0,02	2833,60	0,49
gr21_25	2962,60	11,44	2962,60	0,14	2962,60	0,02	2962,60	0,02	2962,60	0,32
gr21_50	3738,10	10,32	3738,10	0,01	3738,10	0,05	3738,10	0,02	3738,10	0,22

As instâncias testadas nos experimentos foram comparadas aos resultados para o PCVLC encontrados na literatura, como mostram as tabelas 1 e 2. Na primeira coluna das tabelas encontram-se os nomes das instâncias com os respectivos percentuais de portos com limite de calado. A coluna BCP detalha os resultados exatos, obtidos com o método *branch-cut-and-price* apresentado em Battarra et al. [2014]. As colunas GVNS-1 e GVNS-2 apresentam os resultados obtidos com as variações da metaheurística GVNS (*General Variable Neighborhood Search*) apresentadas em Todosijević et al. [2014]. A coluna GRASP apresenta os resultados desta metaheurística encontrados em Machado et al. [2015]. Por fim, a coluna SA apresenta os resultados da abordagem de solução proposta neste trabalho.

A tabela 1 apresenta a média dos melhores resultados para cada subgrupo de 10 instâncias. Como, para a maioria das instâncias, os métodos heurísticos chegaram à solução ótima, optou-se apresentar os resultados na forma de média simples, evitando replicar as tabelas para cada grupo de 30 instâncias.

Salienta-se que, diante da impossibilidade de replicar os experimentos em equipamentos equivalentes aos utilizados na literatura, os tempos de execução não foram considerados para efeitos comparativos diretos entre os métodos, sendo esses apresentados tais como foram medidos durante a execução dos experimentos em seus trabalhos de origem.

Como pode-se observar nas tabelas, o SA é bastante competitivo e consegue atingir a solução ótima para a maioria das instâncias testadas. A instância $gr48$ foi a única para a qual não se conseguiu atingir as soluções ótimas. Em função disto, a 2 apresenta os testes exaustivos para todos as instâncias do grupo, apresentando o melhor resultado obtido por cada método para cada instância. Em negrito são destacados os valores do SA que não correspondem à solução ótima.

Os experimentos computacionais foram realizados em um equipamento com pro-

cessador IntelR CoreTM i5-4210, 1.70 GHz - 2.40 GHz, com 4 GB de memória RAM e um sistema operacional Debian 8 Jessie. O algoritmo foi implementado na linguagem C (gcc 4.9.2).

Tabela 2: A instância gr48 detalhada

	BCP		GVNS-1		GVNS-2		GRASP		SA	
Instância	Sol.	T(s)	Sol.	T(s)	Sol.	T(s)	Sol.	T(s)	Sol.	T(s)
gr48_10_1	5046	813,29	5046	0,01	5046	0,00	**5524**	0,24	**5164,00**	1,48
gr48_10_2	5542	341,27	5542	0,01	5542	0,10	**5895**	0,68	**5648,00**	0,98
gr48_10_3	5300	3314,02	5300	0,03	5300	0,06	**5754**	0,29	**5358,00**	1,08
gr48_10_4	5293	7057,36	5293	1,32	5293	0,45	**5588**	1,77	**5329,00**	1,02
gr48_10_5	5679	496,29	5679	0,21	5679	30,27	**6159**	29,99	**5826,00**	0,84
gr48_10_6	5610	41,04	5610	0,39	5610	0,42	**5760**	8,49	**5699,00**	0,80
gr48_10_7	5063	22,79	5063	0,00	5063	1,20	**5955**	15,35	**5271,00**	1,02
gr48_10_8	5103	188,77	5103	0,01	5103	1,15	**5562**	2,35	**5227,00**	0,93
gr48_10_9	5153	892,87	5153	0,00	5153	0,17	**5792**	1,87	**5311,00**	1,33
gr48_10_10	5055	359,47	5055	0,00	5055	0,01	**6014**	2,73	**5224,00**	1,57
gr48_25_1	5524	1073,43	5524	0,46	5524	0,47	5524	0,24	**5631,00**	0,62
gr48_25_2	5895	361,90	**5901**	0,35	5895	0,03	5895	0,68	**6186,00**	0,62
gr48_25_3	5754	47,85	5754	0,04	5754	0,56	5754	0,29	**5845,00**	0,76
gr48_25_4	5588	126,05	5588	2,34	5588	2,34	5588	1,77	**5700,00**	0,65
gr48_25_5	6159	1793,96	6159	7,62	6159	4,55	6159	29,99	**6389,00**	0,57
gr48_25_6	5760	3053,47	**5777**	10,73	5760	8,49	5760	8,49	**5831,00**	0,61
gr48_25_7	5955	172,08	5955	101,04	5955	41,29	5955	15,35	**6248,00**	0,55
gr48_25_8	5562	1070,85	5562	2,29	5562	0,43	5562	2,35	**5797,00**	0,65
gr48_25_9	5792	184,24	5792	2,46	5792	2,46	5792	1,87	**5966,00**	0,64
gr48_25_10	6014	730,82	6014	5,08	6014	5,08	6014	2,73	**6216,00**	0,58
gr48_50_1	6096	157,47	6096	0,59	6096	0,64	6096	1,24	**6499,00**	0,60
gr48_50_2	6629	21,44	6629	31,53	6629	12,30	6629	1,89	**6897,00**	0,48
gr48_50_3	5896	132,45	5896	3,94	5896	13,79	5896	4,94	**6289,00**	0,51
gr48_50_4	6404	20,75	6404	0,07	6404	0,49	6404	0,51	**6486,00**	0,47
gr48_50_5	6617	19,41	6617	0,06	6617	0,03	6617	0,23	**6755,00**	0,49
gr48_50_6	8533	66,09	8533	5,46	8533	0,48	8533	1,14	**8658,00**	0,45
gr48_50_7	6166	1520,10	6166	0,60	6166	19,59	6166	1,33	**6401,00**	0,51
gr48_50_8	6535	20,93	6535	0,18	6535	27,76	6535	9,73	**6698,00**	0,45
gr48_50_9	7150	42,73	7150	0,25	7150	7,92	7150	6,36	**7494,00**	0,48
gr48_50_10	6331	112,13	6331	17,04	6331	0,04	6331	0,59	**6538,00**	0,48

5. Conclusão

O PCV é um clássico problema em otimização combinatória e tem sido investigado por décadas. O transporte marítimo é uma aplicação do PCV que tem atraído atenção da indústria devido a redução de custos que um planejamento de rota otimizado pode ocasionar. Entretanto, a aplicabilidade do PCV a transportes marítimos pode sofrer outras restrições além das tradicionais impostas pela obtenção de um circuito hamiltoniano de menor custo.

Uma severa restrição é o problema do limite de calado. Este limite impede que uma embarcação atraque em um porto sob pena de ficar encalhada. O *Problema do Caixeiro Viajante com Limite de Calado* (PCVLC) é, portanto, uma varição do PCV que visa atender a esta restrição. O PCVLC é um problema de pesquisa relativamente novo. Ele foi introduzido por [Glomvik Rakke et al., 2012] e, até o presente momento, a literatura apresenta poucas abordagens de soluções para o problema.

O trabalho aqui proposto tem a finalidade confrontar soluções atualmente presentes na literatura para o PCVLC com uma outra metaheurística clássica, o SA. A solução proposta foi testada para as 240 instâncias apresentadas em Glomvik Rakke et al. [2012]. Com base nos resultados obtidos, pode-se observar que o SA é uma alternativa viável quando comparada a outras metaheurísticas. O SA conseguiu atingir a solução ótima para 7 dos 8 grupos de instância testada e falhou em atingir o ótimo apenas para um deles, o grupo $gr48$.

Em função desses resultados, como trabalho futuro os autores pretendem explorar novas técnicas de busca local para a perturbação da vizinhança, de forma a se atingir melhor resultados para instâncias maiores.

Referências

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.

Cerny, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1): 41–51.

Cook, W. J. (2011). *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press.

Feo, T. A. e Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.

Glomvik Rakke, J., Christiansen, M., Fagerholt, K., e Laporte, G. (2012). The traveling salesman problem with draft limits. *Computers & Operations Research*, 39(9):2161–2167.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., e Schevon, C. (1989). Optimization by simulated annealing: An experimental evaluation. part i, graph partitioning. *Operation Research*, 37(6):865–892. ISSN 0030-364X.

Kirkpatrick, S., Gelatt, C. D., e Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

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 12º Congresso Brasileiro de Inteligência Computacional*, p. 1–6, Curitiba, PR. ABRICOM.

Reinelt, G. (1991). Tsplib - a traveling salesman problem library. *INFORMS Journal on Computing*, 3(4):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.

Youssef, H., Sait, S. M., e Adiche, H. (2001). H.: Evolutionary algorithms, simulated annealing and tabu search: a comparative study. *Engineering Applications of Artificial Intelligence*.