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

Uma heurística híbrida GRASP+VND para o Problema de Alocação de Berços com Múltiplas Cargas

Breno Leite Zupeli, Maria Claudia Silva Boeres, Renato Elias Nunes de Moraes
Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari, 514, Goiabeiras, Vitória, ES, CEP 29075-910
brenolzupeli@gmail.com, {maria.boeres,renato.moraes}@ufes.br

RESUMO

Terminais de Granéis Sólidos (TGS) manipulam e armazenam cargas secas a granel, transportadas sem embalagem em grandes quantidades. Utilizados globalmente, facilitam a exportação de materiais como minério de ferro e grãos. Este estudo é motivado por um dos maiores TGS do mundo, localizado no Brasil. Estudamos o Problema de Alocação de Berços (PAB) em TGS, onde berços operam diferentes tipos de cargas a taxas variadas, com tempos dependentes do berço e da carga. O PAB designa um conjunto de navios a um layout de berços em um horizonte de tempo definido. Apresentamos um Procedimento de Busca Adaptativa Gulosa Randomizada (GRASP em inglês) e sua hibridização com o algoritmo Descida por Vizinhança Variável (VND em inglês), GRASP+VND, que resolvem instâncias geradas a partir de dados reais do Complexo Portuário de Tubarão. Os resultados são comparados aos de um modelo matemático do problema, demonstrando que os algoritmos fornecem soluções de alta qualidade.

PALAVRAS CHAVE. Problema de Alocação de Berços, Logística de portos graneleiros, metaheurística híbrida GRASP+VND.

Tópicos (indique, em ordem de PRIORIDADE, o(s) tópicos(s) de seu artigo): OC - Otimização Combinatória, MH - Meta-heurísticas.

ABSTRACT

Dry Bulk Terminals (DBT) handle and store dry bulk cargo, transported unpackaged in large quantities. Used worldwide, they facilitate the export of materials such as iron ore and grains. This study is motivated by a real case of one of the world's largest DBT, located in Brazil. We study the Berth Allocation Problem (BAP) in DBT, where berths operate different types of cargo at varying rates, with times depending on both the berth and the cargo. The BAP involves assigning a set of vessels to a berth layout within a defined time horizon. We present a Greedy Randomized Adaptive Search Procedure (GRASP) heuristic and its hybridization with the Variable Neighborhood Descent (VND) algorithm, GRASP+VND. They solve instances generated from real data of the Port of Tubarão Complex. The results are compared against those generated by a mathematical model fot the problem, demonstrating the algorithms provide high-quality solutions.

KEYWORDS. Berth Allocation Problem, Bulk ports logistics, Hybrid metaheuristic GRASP+VND.

Paper topics (indicate in order of PRIORITY the paper topic(s)): OC - Combinatorial Optimization, MH - Metaheuristics.

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

1. Introdução

Terminais de granéis sólidos (*Dry Bulk Terminals* - DBT) são centros de transporte para a movimentação de materiais secos a granel, transportados sem embalagem (sem recipientes) em grandes quantidades. Produtos a granel secos típicos, como minério de ferro, carvão e grãos, são essenciais para o desenvolvimento das atividades de produção e para a manutenção do bem estar do indivíduo. Diante do aumento da demanda global por esses materiais [Lodewijks et al., 2007], houve um crescimento significativo no transporte desses produtos nas últimas décadas.

Este estudo foi motivado pelo terminal de exportação do Porto de Tubarão, localizado em Vitória, no estado do Espírito Santo. O porto está entre os terminais de exportação de minério de ferro mais rápidos do mundo [Vale, 2023]. A eficiência de distribuição aos respectivos navios (ou embarcações) das cargas em um terminal de exportação como o Porto de Tubarão deve ser controlada pelo operador portuário de forma a proporcionar um planejamento logístico seguro e econômico. Um aspecto importante para uma melhor gestão em um terminal portuário é a alocação dos navios que chegam aos berços de forma eficiente. Assim, o foco deste artigo é o problema de alocação de berços (*Berth Allocation Problem* - BAP), considerando o berço como um recurso discreto (o cais é dividido em várias posições de atracação específicas e apenas um navio pode ser atendido por vez) em um DBT de exportação com comportamento dinâmico de chegada dos navios (os horários de chegada das embarcações são fixos, portanto, os navios não podem atracar antes da hora prevista de chegada).

O problema de alocação de berços consiste em, dado um conjunto de embarcações a serem atracadas, suas características e demandas, determinar as posições e os tempos de atracação de cada embarcação, minimizando o custo total e respeitando as restrições do porto. O objetivo mais usual é minimizar o tempo de permanência dos navios no porto (reduzir o tempo de espera para atracar) mais o tempo para operar [Bierwirth e Meisel, 2010]. Abrangentes revisões da literatura sobre aplicações, modelos e algoritmos para o tema são encontradas em [Stahlbock e Voß, 2008; Bierwirth e Meisel, 2010; Rashidi e Tsang, 2013; Bierwirth e Meisel, 2015]. Especificamente para DBT tem-se os trabalhos de [Umang et al., 2013; Kordić et al., 2016; Barros et al., 2011; Ernst et al., 2017]. Umang et al. [2013] propuseram vários modelos e heurísticas para o BAP em terminais de granel com berços especializados em movimentação de cargas específicas. [Kordić et al., 2016] propôs um modelo e algumas heurísticas para regras de implementação de BAP discreto fornecidas pelo operador do terminal. Alguns autores propuseram modelos para o BAP para portos de granel, considerando restrições de marés [Barros et al., 2011; Ernst et al., 2017].

Na versão do BAP tratada neste artigo o tempo de operação depende da relação entre o berço que o navio irá atracar e da carga transportada, portanto, o tempo de operação não é conhecido *a priori*. Esse problema foi tratado por Banos et al. [2016], que o denominaram Problema de Alocação de Berços com Múltiplas Cargas (PAB-MC). Eles apresentaram uma formulação matemática baseada no Problema de Roteamento de Veículos com Múltiplos Depósitos e com Janela de Tempo proposto por Cordeau et al. [2005] e uma meta-heurística *Simulated Annealing* para resolver instâncias com base nos dados reais do Complexo Portuário de Tubarão.

Diferentemente dos métodos exatos e heurísticos utilizados na literatura para resolver o PAB-MC e suas variantes [Banos et al., 2016; Rosa et al., 2017], este estudo emprega uma abordagem com metaheurística híbrida, chamada GRASP+VND, para resolver as instâncias do problema apresentadas em Banos et al. [2016]. Portanto, o objetivo deste estudo é investigar o desempenho do método híbrido no PAB-MC, considerando para o problema: (1) a implementação da meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) [Feo e Resende, 1995] e (2) a implementação da meta-heurística híbrida GRASP+VND na qual a busca local do GRASP é aprimorada pelo uso da meta-heurística *Variable Neighborhood Descent* (VND) [Hansen et al., 2017].

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

Os experimentos computacionais relatados demonstram a eficiência do algoritmo proposto, que foi capaz de encontrar todas as soluções ótimas conhecidas e aprimorou os melhores valores conhecidos de todas as instâncias utilizadas.

O restante deste artigo está organizado da seguinte maneira. Na Seção 2 são apresentadas as características específicas do PAB-MC considerado. Na Seção 3 são descritos os algoritmos propostos e implementados para resolver o PAB-MC. Os experimentos e os resultados computacionais são descritos na Seção 4. Por fim, a Seção 5 apresenta as principais conclusões baseadas nos resultados experimentais obtidos e possíveis trabalhos futuros.

2. Definição do Problema

Neste trabalho, é estudado o problema discreto de alocação de berços com múltiplas cargas (DBAP-MC) [Banos et al., 2016]. O DBAP-MC é baseado em [Cordeau et al., 2005], que resolveu o DBAP inspirado no Problema de Roteamento de Veículos com Múltiplos Depósitos e Janela de Tempo. A principal diferença entre os modelos matemáticos propostos por [Banos et al., 2016] e por [Cordeau et al., 2005] está nos tempos de atendimento dos navios. Enquanto Cordeau et al. [2005] consideram os tempos de atendimento dos navios em cada berço, uma entrada do modelo, Banos et al. [2016] propõem que o tempo de atendimento dependa da relação entre o navio, sua carga e o berço, portanto, sendo conhecido somente após a resolução do modelo.

No DBAP-MC são considerados como conjuntos de entrada o conjunto de navios $\mathcal{V} = \{1, 2, \ldots, |\mathcal{V}|\}$; o conjunto de berços $\mathcal{B} = \{1, 2, \ldots, |\mathcal{B}|\}$ e o conjunto de cargas operadas pelo porto $\mathcal{C} = \{1, 2, \ldots, |\mathcal{C}|\}$. Os parâmetros associados a cada navio $i \in \mathcal{V}$ e a cada berço $b \in \mathcal{B}$ são apresentados, respectivamente, nas Tabelas 1 e 2.

Tabela 1: Parâmetros relacionados aos navios

Parâmetro	Descrição
a_i	tempo de chegada
c_i	tempo máximo de permanência, toda a operação no navio i deve ser concluída antes deste prazo
o_i	comprimento
f_i	calado
$Q_i \subseteq \mathcal{C}$	conjunto de cargas
q_{ij}	quantidade (toneladas) da carga $j \in Q_i$ a ser processada no porto pelo navio i

Tabela 2: Parâmetros relacionados aos berços

Parâmetro	Descrição
s^b	tempo de abertura
e^b	tempo de fechamento
w^b	comprimento
h^b	profundidade do cais
$R^b \subseteq \mathcal{C}$	Conjunto de cargas que podem ser manipuladas no berço b
$N^b \subseteq \mathcal{V}$	Conjunto de navios que podem ser manipulados no berço b
l_j^b	capacidade de processamento da carga $j \in R^b$ pelo berço b

Cada navio $i \in \mathcal{V}$ deve ser atribuído a apenas um berço $b \in \mathcal{B}$, que por sua vez, pode atender no máximo um navio por vez. Um navio $i \in \mathcal{V}$ pode ser atribuído a um berço $b \in \mathcal{B}$ se e somente se o berço b manipula a carga do navio i, isto é, se $Q_i \subseteq R^b$ e se seu comprimento e calado forem compatíveis ($(o_i, f_i) \leq (w_b, h_b)$). A função de atracação $k : \mathcal{V} \to \mathcal{B}$ associa a cada navio $i \in \mathcal{V}$, o berço $b \in \mathcal{B}$ que satisfaça às condições acima. Além disso, para cada navio $i \in \mathcal{V}$, seu tempo de atendimento $t_i^{k(i)}$ depende do berço $k(i) = b \in \mathcal{B}$ onde é atracado e da carga Q_i armazenada no navio i, a ser manipulada pelo berço b tal que $Q_i \subseteq R^b$. Desta forma, o tempo de atendimento $t_i^{k(i)}$ de um navio $i \in \mathcal{V}$, atracado no berço $k(i)$ é dado por $t_i^{k(i)} = \sum_{j \in Q_i} q_{ij}/l_j^{k(i)}$. Como resposta, espera-se o conjunto dos tempos de atracação $T_i^{k(i)}$, para cada um dos navios $i \in \mathcal{V}$ quando atracados nos berços $k(i) \in \mathcal{B}$. Uma melhor descrição e formulações matemáticas do DBAP-MC são fornecidas por [Banos et al., 2016; Rosa et al., 2017].

Neste contexto, o tempo de atendimento de um navio é definido como o tempo entre a sua chegada ao porto e a conclusão das operações de carregamento ou descarregamento, $T_i^{k(i)} -$

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

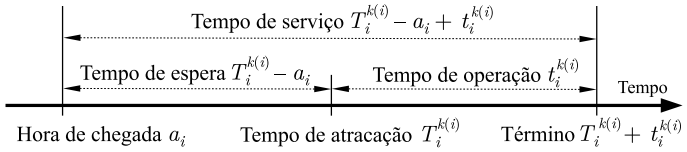


Figura 1: Representação das variáveis de tempo

$a_i + t_i^{k(i)}$ (veja Figura 1). Uma solução para o DBAP-MC é representada como um conjunto $S = \{T_1^{k(1)}, T_2^{k(2)}, \ldots, T_{|\mathcal{V}|}^{k(|\mathcal{V}|)}\}$, ou seja, a solução é a atribuição do tempo de atracação $T_i^{k(i)}$ para todo navio $i \in \mathcal{V}$ junto com a determinação do berço $k(i) \in \mathcal{B}$ ao qual i foi alocado. O objetivo do DBAP-MC é minimizar o tempo total de atendimento e, dada uma solução S, sua função objetivo (FO) é calculada como $FO(S) = \sum_{i \in \mathcal{V}} (T_i^{k(i)} - a_i + t_i^{k(i)})$.

Tabela 3: Exemplo de Parâmetros dos navios

Navios					quantidade de carga (q_{ij})			
i	a_i	c_i	o_i	f_i	$j=1$	$j=2$	$j=3$	$j=4$
1	5	16	20	12	2	–	–	–
2	6	18	25	15	–	8	–	10
3	3	12	20	15	–	–	4	–
4	3	22	30	18	2	–	4	–
5	11	20	25	15	–	–	–	30

Um exemplo para o problema DBAP-MC é o apresentado pelas Tabelas 3, 4 e 5 (entrada) e Figura 2 (saída). Um conjunto de valores para os parâmetros dos navios são mostrados na Tabela 3, para os parâmetros dos berços na Tabela 4 e para os tempos de atendimento ($t_i^{k(i)}$) na Tabela 5. No caso dinâmico, os navios podem chegar a qualquer momento ao longo do horizonte de planejamento, indicado pela barra verde em cada berço (veja Figura 2). Observe na figura, que tempos ociosos podem aparecer na janela de tempo de funcionamento do berço. Os berços 1 e 2 são abertos nos tempos 4 e 3, e fechados nos tempos 20 e 18, respectivamente. Observe ainda que o navio $i = 2$ não pode atracar no berço $b = 2$ porque este berço não manipula a carga $j = 2$. O navio $i = 4$ não pode atracar no berço $b = 2$ porque $o_4 > w^2$ (Veja Tabelas 3 e 4).

Tabela 4: Exemplo de Parâmetros dos berços

Berços					Capacidade de processamento (l_j^b)			
b	s^b	e^b	w^b	h^b	$j=1$	$j=2$	$j=3$	$j=4$
1	4	20	30	20	2	2	1	5
2	3	18	25	20	1	–	4	6

Na solução DBAP-MC apresentada na Figura 2, os navios v_1 e v_2 são atribuídos ao berço 1, enquanto os navios v_3, v_4 e v_5 são atribuídos ao berço 2. O navio v_1 chega no tempo 5 e logo é atribuído ao berço 1, ficando por 1 unidade de tempo lá, conforme indicado na Tabela 5, $t_1^{k(1)} = 1$. O navio v_2 é atribuído ao berço 1 após a saída do navio v_1 e permanece por 6 unidades de tempo. As 6 unidades de tempo correspondem (ver Tabela 5) às 4 unidades de tempo necessárias para processar $q_{22} = 8$ toneladas da carga $j = 2$ no berço 1 com capacidade $l_2^{k(2)=1} = 2$, ou seja, $q_{22}/l_2^{k(2)=1} = 8/2 = 4$ somadas às 2 unidades de tempo necessárias para processar $q_{24} = 10$ toneladas da carga $j = 4$ no berço 1 com capacidade $l_4^{k(2)=1} = 5$, ou seja, $q_{24}/l_4^{k(2)=1} = 10/5 = 2$.

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

Tabela 5: Exemplo de Tempos de Atendimento

	$q_{ij}/l_j^{k(i)=1}$				$q_{ij}/l_j^{k(i)=2}$				$t_i^{k(i)}$	
i	$j=1$	$j=2$	$j=3$	$j=4$	$j=1$	$j=2$	$j=3$	$j=4$	$k(i)=1$	$k(i)=2$
1	2/2	–	–	–	2/1	–	–	–	1	2
2	–	8/2	–	10/5	–	–	–	–	6	–
3	–	–	4/1	–	–	–	4/4	–	4	1
4	2/2	–	4/1	–	2/1	–	4/4	–	5	3
5	–	–	–	30/5	–	–	–	30/6	6	5

A mesma análise pode ser feita para determinar os tempos de atendimento dos navios v_3, v_4 e v_5.

Observa-se que, como o navio v_4 chega no tempo $a_4 = 3$ e é atribuído ao berço $k(4) = 2$ no tempo $T_4^{k(4)} = 4$, ele precisa esperar $T_4^{k(4)} - a_4 = 1$ unidade de tempo. Por esse motivo, é necessária mais uma unidade de tempo para as operações de carga e descarga iniciarem. Considerando que $t_4^{k(4)} = 3$ (veja Tabela 5), o tempo total que o navio v_4 permanece no porto é igual a 4 unidades de tempo, que são contabilizadas no valor da função objetivo. Ou seja $T_4^{k(4)} - a_4 + t_4^{k(4)} = 4 - 3 + 3$, onde $k(4) = 2$. Supondo que o navio v_4 fosse atendido pelo berço 1, ou seja $k(4) = 1$, o valor contabilizado seria $T_4^{k(4)} - a_4 + t_4^{k(4)} = 4 - 3 + 5$.

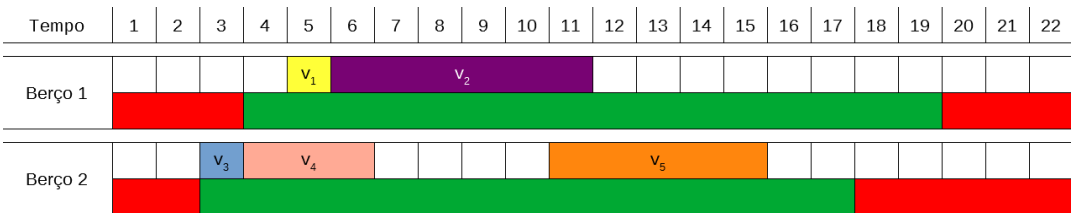


Figura 2: Exemplo de solução de alocação de berços

A solução DBAP-MC da Figura 2 pode ser representada como $S = \{T_1^1 = 5, T_2^1 = 6, T_3^2 = 3, T_4^2 = 4, T_5^2 = 11\}$. Nesse exemplo, $t_1 = 1$, $t_2 = 6$, $t_3 = 1$, $t_4 = 3$ e $t_5 = 5$. Portanto, os navios v_1 a v_5 permanecem no porto por 1, 6, 1, 4 e 5 unidades de tempo, respectivamente, e o valor da função objetivo é $f_{DBAP-MC} = 17$ unidades de tempo.

3. Heurísticas para Solucionar o DBAP-MC

O GRASP [Feo e Resende, 1995] é uma meta-heurística bastante utilizada em vários tipos de aplicações de problemas de otimização. Consiste de um processo iterativo de múltiplos inícios (*multi-start*). Sua implementação mais simples considera, a cada iteração, a construção de uma solução inicial seguida de uma busca local para melhoria dessa solução.

Neste trabalho é proposto um algoritmo utilizando o GRASP, que dispõe de um mecanismo baseado em uma Lista Restrita de Candidatos (LRC) utilizada na construção da solução inicial. O uso da LRC é uma estratégia construtiva aleatória. Dados um conjunto $X = \{x^1, x^2, ..., x^{|X|}\}$ de todos os elementos candidatos a compor uma solução e uma função gulosa g para computar seus custos, onde $\underline{g}(X) = min_{1 \leq j \leq |X|} g(x^j)$ e $\overline{g}(X) = max_{1 \leq j \leq |X|} g(x^j)$ sejam, respectivamente, o mínimo e máximo valor guloso sobre todos os elementos candidatos viáveis para compor a solução, a LRC é formada por todos os elementos viáveis $x^\ell \in X$, $1 \leq \ell \leq |X|$, tal que $g(x^\ell) \leq \underline{g}(X) + \alpha(\overline{g}(X) - \underline{g}(X))$, com $0,0 < \alpha \leq 1,0$. O caso $\alpha = 0,0$ corresponde a um algoritmo guloso puro, enquanto $\alpha = 1,0$ é equivalente a uma construção totalmente aleatória. Em seguida, um elemento x^ℓ é aleatoriamente selecionado da LRC para ser parte da solução. O tamanho da LRC é determinado pelo parâmetro α, que pode assumir uma constante ou se auto-ajustar

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

a cada β iterações, atualizando a probabilidade de cada α ser escolhido, chamado α reativo. Neste último caso, um conjunto $\mathcal{A}$ de valores reais entre 0 e 1 é definido para α. Os valores de α que gerarem melhores soluções terão suas probabilidades aumentadas e, consequentemente, nas próximas iterações terão maior chance de serem escolhidos. O GRASP com α reativo é denotado de GRASP Reativo [Prais e Ribeiro, 2000]. Seguindo o processo de *multi-start*, soluções iniciais diferentes são criadas a fim de promover a diversificação, sendo cada uma submetida a uma busca local. A melhor solução entre todas as iterações é a saída do algoritmo.

Nesta seção propomos e descrevemos o GRASP reativo para resolver o DBAP-MC. A Seção 3.1 descreve o algoritmo GRASP básico. O procedimento construtivo aleatório é apresentado na Seção 3.2. As buscas locais e o VND são apresentados na Seção 3.3.

3.1. GRASP Reativo para o DBAP-MC

Algoritmo 1: GRASP Reativo para o DBAP-MC

Entrada: $\mathcal{V}, \mathcal{B}, seed, \mathcal{A}, \beta$
Saída: A melhor solução encontrada S_{best}
1 $S_{best} \leftarrow \emptyset$;
2 $FO(S_{best}) \leftarrow +\infty$;
3 $\alpha \leftarrow$ `Escolhe_Alfa`$(\mathcal{A}, seed)$;
4 **repita**
5 $\quad S \leftarrow$ `ConstrutivoAleatorio`$(\mathcal{B}, \mathcal{V}, \alpha)$;
6 $\quad S' \leftarrow$ `BuscaLocal`$(S, \mathcal{B}, \mathcal{V})$;
7 $\quad$ **se** $FO(S') < FO(S_{best})$ **então**
8 $\quad\quad S_{best} \leftarrow S'$;
9 $\quad$ **fim se**
10 $\quad \alpha \leftarrow$ `Atualiza_Alfa`$(\mathcal{A}, \beta)$;
11 **até** critério de parada alcançado;
12 **retorna** S_{best}

No Algoritmo 1 é apresentado o o pseudocódigo do GRASP para resolver o DBAP-MC. A entrada para o algoritmo consiste do conjunto de navios $\mathcal{V}$, junto com os parâmetros da Tabela 1, o conjunto de berços $\mathcal{B}$, junto com os parâmetros da Tabela 2 e os parâmetros para implementação do α reativo: a semente aleatória $seed$, o conjunto $\mathcal{A}$ com os valores de α e o parâmetro β que define a quantidade de iterações para que se atualize as probabilidades de α.

Nas linhas 1 a 3 do Algoritmo 1 são inicializadas, respectivamente, a melhor solução encontrada até então S_{best} como $\emptyset$ (vazia), o seu valor de função objetivo $FO(S_{best})$ como $+\infty$ e o valor de α, usando a função `Escolhe_Alfa()` que retorna um valor aleatório do conjunto $\mathcal{A}$ baseado no valor da semente $seed$. No laço das linhas 4 a 11, ocorre a fase de construção aleatória da solução inicial (linha 5), onde se cria, iterativamente, uma solução viável S (ver Seção 3.2). Em seguida, na linha 6, S é submetida à busca local. No decorrer da fase de busca local refina-se a solução S através de uma busca em sua vizinhança. Ao final dessa fase, é encontrada a solução ótima local (ver Seção 3.3). Caso a solução ótima local seja melhor que a melhor solução encontrada até o momento S_{best}, a melhor solução é atualizada com o valor da solução atual (linhas 7 a 9). Na linha 10 o parâmetro α é ajustado usando a estratégia reativa descrita em [Prais e Ribeiro, 2000] com a distribuição de probabilidade sendo atualizada a cada β iterações.

Esse processo de duas fases do GRASP se repete até que uma determinada condição de parada seja alcançada, linha 11. A melhor solução encontrada ao final de todas as iterações do GRASP é retornada como resultado (S_{best}) na linha 12.

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

3.2. Heurística Construtiva Aleatória

Na fase de construção da solução inicial (linha 5 do Algoritmo 1), uma heurística construtiva aleatória é utilizada para construir uma solução viável S que será então submetida à fase de busca local (linha 6 do Algoritmo 1). A heurística implementada está baseada em Scarpino et al. [2018], na qual, dada uma sequência arbitrária de navios (ordem de leitura da instância de entrada), seleciona-se um berço aleatoriamente de uma LRC. A construção da LRC é guiada por uma função gulosa baseada no número de navios atualmente alocados em cada berço. Essa função é definida como $g(\mathcal{B}) = \min_{1 \leq b \leq |\mathcal{B}|} |n^b|$, onde $|n^b|$ é o número de navios atualmente alocados no berço b. A primeira etapa do algoritmo termina quando todo navio $i \in \mathcal{V}$ foi alocado a exatamente um berço $k(i) = b \in \mathcal{B}$, respeitadas todas as restrições do problema.

A segunda etapa da heurística construtiva consiste em estabelecer a ordem de atendimento e, consequentemente, o tempo de atracação $T_i^{k(i)}$ de cada navio i alocado ao berço $k(i)$. Considerando que no DBAP-MC o tempo de operação depende da relação entre o berço que o navio irá atracar e da carga transportada, neste trabalho, é proposto o critério de ordenação baseado no custo total da operação dos navios nos berços, buscando reduzir o custo total de operação dos berços. Para isso, levam-se em consideração o tempo de processamento dos navios e o horário de abertura do berço. Durante a ordenação em um berço b, ao comparar dois navios i e j subsequentes, é calculado o tempo total de processamento desses dois navios nessa ordem, ou seja, $max(s_b, a_i) + t_i^b + max(max(s_b, a_i) + t_i^b, a_j) + t_j^b$, que, observando a disposição da solução apresentada na Figura 3 e, utilizando o berço 2 e os navios v_4 e v_3 como i e j respectivamente, resulta em $max(3,3)+3+max(max(3,3)+3,3)+1 = 13$. Em seguida, é realizada uma troca na ordem de processamento dos dois navios, que resulta em $max(3,3)+1+max(max(3,3)+1,3)+3 = 10$. Caso a troca resulte em um valor maior, a troca é desfeita. Se resultar em um valor menor, como nesse caso, a troca é mantida, refletindo essa redução no valor da função objetivo. Esse processamento é realizado em cada par de navios para cada berço. O resultado do processamento do algoritmo é exibido na Figura 3. Ao término da segunda etapa, tem-se uma solução viável $S = \{T_1^{k(1)}, T_2^{k(2)}, \ldots, T_{|\mathcal{V}|}^{k(|\mathcal{V}|)}\}$.

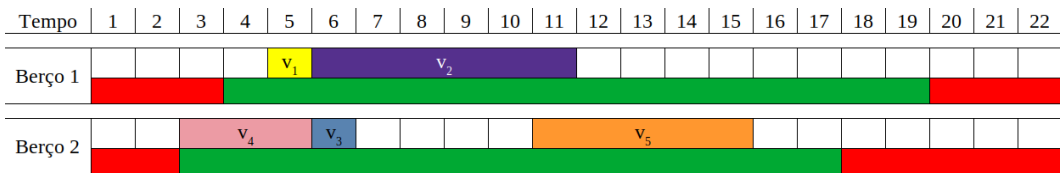


Figura 3: Exemplo de solução de alocação de berços usando o critério de ordenação por custo da operação

3.3. Busca Local

A segunda fase do GRASP (linha 6 do Algoritmo 1) consiste na aplicação, a cada iteração, de busca local na solução construtiva aleatória realizada na primeira fase (linha 5 do Algoritmo 1).

Neste trabalho, a proposta foi implementar um GRASP no qual a estratégia de busca local é o método de Descida em Vizinhança Variável (VND – *Variable Neighborhood Descent*) [Hansen et al., 2017]. O VND é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de vizinhanças, aceitando somente soluções de melhoria da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

O método VND utilizado possui dois tipos de movimentos (descritos em Oliveira et al. [2012]), visando alterar a solução realizando buscas em sua vizinhança: o movimento *move* move um navio atracado a um determinado berço, para um berço diferente e o movimento *swap* seleciona

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

dois navios atracados a berços diferentes, trocando-os. Após o movimento dos navios, a sequência de navios nos berços afetados é reordenada utilizando o critério por tempo de chegada ou o critério por custo da operação (Seção 3.2). Uma busca pela vizinhança consiste em aplicar todas as possibilidades de movimento (*move* ou *swap*) e escolher o melhor vizinho (melhor aprimorante).

Neste trabalho os dois movimentos (*move* e *swap*) e os dois critérios de ordenação de navios nos berços (tempo de chegada e custo da operação) foram combinados para formar o conjunto $N = \{N_1, N_2, N_3, N_4\}$, de quatro estruturas de vizinhança utilizadas na heurística VND, cuja implementação segue a versão *Sequential VND using the best improvement search strategy*, descrita em Hansen et al. [2017]. A ordem de aplicação foi definido como: N_1, movimento *move* com critério de ordenação por tempo de chegada; N_2, movimento *swap* com critério de ordenação por tempo de chegada; N_3, movimento *move* com critério de ordenação por custo da operação e N_4, movimento *swap* com critério de ordenação por custo da operação.

4. Resultados Computacionais

Todos os algoritmos foram codificados em C e compilados com o compilador *gcc* versão 5.4.0. Os experimentos foram realizados em um computador com processador Intel Core i7-6800K de 3.4GHz com 64GB de memória RAM e sistema operacional Ubuntu 16.04.

A Tabela 6 apresenta detalhes das onze instâncias utilizadas nos testes e disponibilizadas por Banos et al. [2016], como o número de berços ($|\mathcal{B}|$) e navios ($|\mathcal{V}|$). Banos et al. [2016] apresenta os resultados ótimos para as 5 primeiras instâncias da Tabela 6 e os melhores resultados conhecidos para as outras 6 instâncias.

Tabela 6: Conjunto de instâncias para o DBAP-MC

Instância	$\|\mathcal{B}\|$	$\|\mathcal{V}\|$	Instância	$\|\mathcal{B}\|$	$\|\mathcal{V}\|$	Instância	$\|\mathcal{B}\|$	$\|\mathcal{V}\|$
B04N010	4	10	B15N010	15	10	B11N100	11	100
B07N010	7	10	B07N100	7	100	B11N150	11	150
B09N010	9	10	B07N150	7	150	B11N250	11	250
B11N010	11	10	B07N250	7	250			

Foram executados um algoritmo que realiza a construção de uma solução de forma puramente gulosa ($alpha = 0,0$) e três versões do algoritmo GRASP, a saber: o GRASP^{temp}, cuja busca local é executada com o movimento *move* e, em seguida, com o movimento *swap* ambos com critério de ordenação por tempo de chegada; o GRASP^{oper}, com execução da busca local de forma análoga ao GRASP^{temp}, porém utilizando critério de ordenação por custo da operação e finalmente o GRASP^{VND}, que utiliza como busca local a heurística VND descrita na Seção 3.3.

Os algoritmos GRASP^{temp}, GRASP^{oper} e GRASP^{VND} foram executados 10 vezes para cada instância, com diferentes sementes para a geração de números aleatórios, considerando como critério de parada o tempo de execução de uma hora ou até atingir 1200 iterações sem melhoria da melhor solução. Esse número de iterações foi escolhido arbitrariamente. A distribuição da probabilidade da estratégia de α reativo foi atualizada após cada $\beta = 20$ iterações.

A Tabela 7 fornece os resultados médios obtidos por todos os algoritmos para cada instância. A coluna Guloso indica o valor da função objetivo encontrada pelo algoritmo construtivo utilizando o parâmetro α igual a $0,0$, consequentemente, não há variação no resultado médio pois a solução encontrada é sempre a mesma. A Fig. 4 mostra o comportamento padrão do algoritmo aleatorizado implementado com estratégia reativa. Na Fig. 4 foi utilizada a instância B07N250 e o GRASP^{VND} restrito a 200 iterações, o suficiente para mostrar o comportamento padrão de cada iteração. Para cada iteração (eixo x) a cruz violeta representa o valor da função objetivo da solução inicial construída pelo construtivo aleatório e o xis laranja representa o valor da função objetivo

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

Tabela 7: Resultados médios (10 execuções para cada instância do problema)

		GRASP^{temp}			GRASP^{oper}			GRASP^{VND}			
Instância	Guloso	t(s)	iter.	FO	t(s)	iter.	FO	t(s)	iter.	FO	g.(%)
B04N010	660,63	0,01	246,60	517,69	0,00	5,40	517,69	0,00	5,40	517,69	21,64
B07N010	587,04	0,00	2,20	555,98	0,00	2,20	555,98	0,00	2,20	555,98	5,29
B09N010	902,03	0,03	240,80	661,40	0,00	1,10	661,40	0,00	1,10	661,40	26,68
B11N010	455,15	0,12	712,70	245,74	0,00	1,40	241,02	0,00	1,40	241,02	47,05
B15N010	264,31	0,00	1,20	117,84	0,00	1,10	115,82	0,00	1,10	115,82	56,18
B07N100	38153,21	0,30	7,20	21462,75	254,41	970,70	16816,52	257,12	944,10	16753,94	56,09
B07N150	66652,27	8,99	568,70	60334,47	646,69	769,80	40720,12	285,74	313,50	39952,76	40,06
B07N250	169792,70	145,91	286,80	63878,70	749,31	58,00	61911,49	752,03	55,90	58672,83	65,44
B11N100	9870,08	18,62	676,30	3297,40	67,69	399,30	3553,77	40,17	214,40	3235,46	67,22
B11N150	18976,74	23,78	247,00	6594,08	192,20	226,70	7007,83	106,34	111,30	6395,09	66,30
B11N250	60954,11	111,40	225,40	11934,45	835,34	101,50	17806,33	813,60	92,60	11856,54	80,55
Média	33388,02	28,11	292,26	15418,23	249,60	230,65	13628,00	205,00	158,45	12632,59	62,16

após a aplicação da busca local na solução inicial. A linha turquesa representa o melhor valor da função objetivo encontrado até aquele momento. A diferença média, ou ganho médio, entre o valor da função objetivo da solução inicial (Guloso) e após a aplicação da busca local é mostrada na Tabela 7 na coluna g.(%) calculada como $(\text{GRASP}^{VND}(\text{FO}) - \text{Guloso})/(\text{GRASP}^{VND}(\text{FO}))$. O ganho, conforme mostrado, pode chegar a $80,55\%$. As colunas FO dos algoritmos GRASP^{temp} e GRASP^{oper} mostram ganhos semelhantes mas menores que os alcançados pelo GRASP^{VND} que se beneficia da combinação de todas as busca locais implementadas.

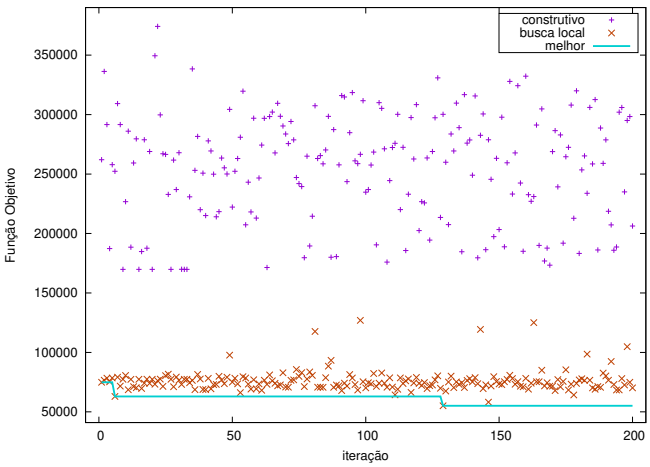


Figura 4: Evolução das soluções para a instância B07N250

A Tabela 7 também mostra, respectivamente, na coluna t(s) e iter., o tempo médio em segundos e número médio de iterações que cada algoritmo leva para encontrar a melhor solução. O tempo e as iterações posteriores que não melhoraram a melhor solução não foram computadas. É possível verificar que o GRASP^{temp} leva no máximo 145,91 segundos para encontrar a melhor solução, enquanto que os demais algoritmos podem gastar até 835,34 segundos. O maior tempo e número de iterações dos algoritmos GRASP^{oper} e GRASP^{VND} mostram que as busca locais com critério de ordenação por custo da operação aumentam a diversificação e intensificação da busca local fazendo o tempo computacional crescer e, ao mesmo tempo, aumentando a efetividade e ro-

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

Tabela 8: Resultados obtidos por todos os algoritmos para todas as instâncias.

		GRASP^{temp}			GRASP^{oper}			GRASP^{VND}		
Instância	MSC[a]	t(s)	Melhor	g.(%)	t(s)	Melhor	g.(%)	t(s)	Melhor	g.(%)
B04N010	*517,69	0,04	517,69	**0,00**	0,00	517,69	**0,00**	0,00	517,69	**0,00**
B07N010	*555,98	0,00	555,98	**0,00**	0,00	555,98	**0,00**	0,00	555,98	**0,00**
B09N010	*661,40	0,10	661,40	**0,00**	0,00	661,40	**0,00**	0,00	661,40	**0,00**
B11N010	*241,02	0,22	243,15	-0,88	0,00	241,02	**0,00**	0,00	241,02	**0,00**
B15N010	*115,82	0,00	117,84	-1,74	0,00	115,82	**0,00**	0,00	115,82	**0,00**
B07N100	24480,00	0,50	21462,75	12,33	695,21	16405,28	32,98	713,57	16393,62	**33,03**
B07N150	51690,00	20,23	60331,87	-16,72	1204,49	40013,02	22,59	858,67	39216,92	**24,13**
B07N250	128570,00	331,02	63628,99	50,51	2657,35	58237,36	54,70	1996,04	55080,22	**57,16**
B11N100	5060,00	35,99	3273,77	35,30	214,77	3383,61	33,13	142,56	3203,69	**36,69**
B11N150	12150,00	75,48	6560,84	46,00	687,88	6755,50	44,40	731,52	6050,23	**50,20**
B11N250	30230,00	347,51	11796,79	60,98	3484,45	16709,38	44,73	3026,88	11261,49	**62,75**

[a] MSC marcadas em * (asterisco) são valores ótimos obtidos pelo CPLEX [Banos et al., 2016].

bustez dos algoritmos ao explorar mais soluções na vizinhança. Essas características podem também ser visualizadas na Fig. 5 que mostra uma rápida convergência do GRASP^{oper} e do GRASP^{VND} quando comparados ao GRASP^{temp}. A Fig. 5 exibe um gráfico de tempo até o alvo (*time to target* [Aiex et al., 2007; Resende e Ribeiro, 2016]) para a instância de teste B09N010 (com o valor de alvo definido como o valor ótimo de $661,40$ e 200 execuções aleatoriamente independentes). Nesse tipo de gráfico cada algoritmo é representado por uma curva e quanto mais a esquerda estiver a curva, melhor será o seu desempenho. Os valores alvo foram escolhidos de forma a permitirem a comparação entre os algoritmos mais e menos efetivos. No gráfico apresentado na Figura 5, a probabilidade de que GRASP^{oper} e do GRASP^{VND} encontrarem o valor alvo em até 10^{-3} segundos é de 100%, caindo para menos de 5% para GRASP^{temp}. A Fig. 6 exibe um gráfico de tempo até o alvo para a instância de teste B07N150 (com o valor de alvo definido como $43000,00$, ou seja $5,6\%$ maior que o melhor valor $40720,12$ encontrado pelo GRASP^{oper} e 200 execuções aleatoriamente independentes). Em nenhuma execução o algoritmo GRASP^{temp} convergiu para o alvo estabelecido e por isso não é mostrado na Fig. 6. No gráfico apresentado na Fig. 6, a probabilidade de que o GRASP^{VND} encontre o valor alvo em 10^1 segundos é de aproximadamente 65%, caindo para aproximadamente 12% para GRASP^{oper}.

Figura 5: Distribuição de tempo para a instância B09N010 com alvo $661,40$.

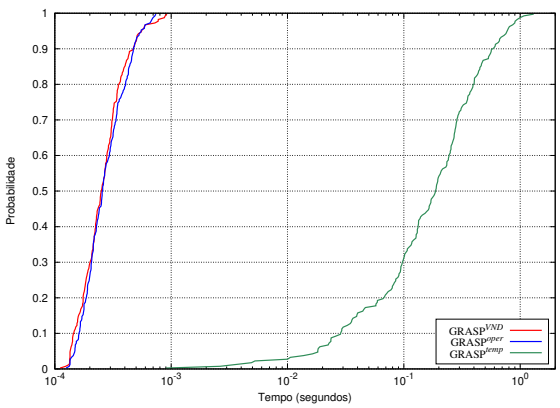
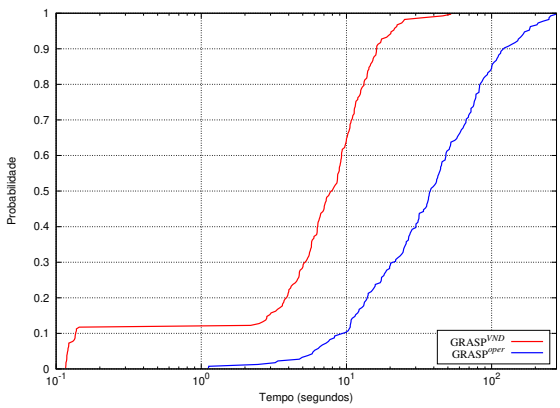


Figura 6: Distribuição de tempo para a instância B07N150 com alvo $43000,00$.



A Tabela 8 apresenta os melhores resultados encontrados por cada algoritmo para cada

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

instância, e os compara com os melhores resultados conhecidos da literatura. A tabela também mostra o maior tempo t(s) em segundos que cada algoritmo levou para alcançar o melhor resultado (o tempo e as iterações posteriores que não melhoraram a melhor solução não foram computados) e ganho percentual g.(%) da solução de cada algoritmo quando comparada à Melhor Solução Conhecida (MSC) na literatura, calculada como $(\text{Melhor} - \text{MSC})/(\text{Melhor})$. Os maiores ganhos estão marcados em negrito. A Tabela 8 mostra que o GRASP^{temp} gasta menos tempo que os outros algoritmos e é capaz de encontrar valores de função objetivo melhores que o GRASP^{oper} nas instâncias maiores: B11N100, B11N150 e B11N250. Os resultados também mostram que quando combinadas as buscas locais de GRASP^{temp} de GRASP^{oper} para formar o GRASP^{VND}, esse último é capaz de encontrar todas os melhores valores de função objetivo sem aumento no tempo computacional quando comparado ao GRASP^{oper}. O GRASP^{VND} foi capaz de melhorar em até 62,75% o valor da melhor solução conhecida das instâncias analisadas.

5. Conclusões e Trabalhos Futuros

Neste artigo foram propostas variações do algoritmo GRASP para solução do DPAB-MC. Dentre elas, a variante híbrida com a busca local aprimorada pelo uso do VND, destacou-se por encontrar os melhores resultados. O sucesso dessa versão pode ser atribuído ao fato de combinar estratégias de busca local propostas para o BAP padrão com outras desenvolvidas nesse trabalho especificamente para o problema PAB-MC característico de terminais marítimos de granéis sólidos.

Os experimentos computacionais mostraram que a heurística híbrida desenvolvida neste trabalho teve um desempenho muito bom, obtendo soluções subótimas muito boas para problemas de teste em tempos de computação razoáveis. O algoritmo é muito robusto, encontrando sistematicamente a mesma solução em um pequeno intervalo de tempo. Além disso, a hibridização do GRASP com o VND contribuiu para tornar a heurística mais efetiva, encontrando todos os melhores resultados independente das características das instâncias.

Para trabalhos futuros, sugere-se a hibridização de outras metaheurísticas, como a busca tabu, com a incorporação de novas buscas locais, vizinhanças e módulos de mineração de dados. Outra linha de investigação que pode ser iniciada é a construção de mais instâncias para DPAB-MC.

6. Agradecimentos

Esta pesquisa foi realizada com apoio parcial da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 e da Fundação de Amparo à Pesquisa do Espírito Santo (FAPES), projetos 368/2022 - P: 2022-NGKM5 e 129/2021 - P: 2021-GL60J.

Referências

Aiex, R. M., Resende, M. G., e Ribeiro, C. C. (2007). Ttt plots: a perl program to create time-to-target plots. *Optimization Letters*, 1(4):355–366.

Banos, R. S., Rosa, R. A., Mauri, G. R., e Ribeiro, G. M. (2016). Modelo matemático e metaheurística simulated annealing para o problema de alocação de berços com múltiplas cargas. *Transportes*, 24(1):51–62.

Barros, V. H., Costa, T. S., Oliveira, A. C., e Lorena, L. A. (2011). Model and heuristic for berth allocation in tidal bulk ports with stock level constraints. *Computers & Industrial Engineering*, 60(4):606 – 613.

Bierwirth, C. e Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202(3):615 – 627.

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br

Bierwirth, C. e Meisel, F. (2015). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 244(3):675 – 689.

Cordeau, J.-F., Laporte, G., Legato, P., e Moccia, L. (2005). Models and tabu search heuristics for the berth-allocation problem. *Transportation Science*, 39(4):526–538.

Ernst, A. T., Oğuz, C., Singh, G., e Taherkhani, G. (2017). Mathematical models for the berth allocation problem in dry bulk terminals. *Journal of Scheduling*, 20(5):459–473.

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

Hansen, P., Mladenović, N., Todosijević, R., e Hanafi, S. (2017). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, 5(3):423–454.

Kordić, S., Davidović, T., Kovač, N., e Dragović, B. (2016). Combinatorial approach to exactly solving discrete and hybrid berth allocation problem. *Applied Mathematical Modelling*, 40(21): 8952 – 8973.

Lodewijks, G., Schott, D., e Ottjes, J. (2007). Modern dry bulk terminal design. *Bulk Solids Handling*, 27(6):364.

Oliveira, R. M., Mauri, G. R., e Lorena, L. A. N. (2012). Clustering search for the berth allocation problem. *Expert Systems with Applications*, 39(5):5499 – 5505.

Prais, M. e Ribeiro, C. C. (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12:164–176.

Rashidi, H. e Tsang, E. P. (2013). Novel constraints satisfaction models for optimization problems in container terminals. *Applied Mathematical Modelling*, 37(6):3601–3634.

Resende, M. G. e Ribeiro, C. C. (2016). *Optimization by GRASP*. Springer.

Rosa, R. D. A., Ribeiro, G. M., Mauri, G. R., e Fracaroli, W. (2017). Planning the berth allocation problem in developing countries with multiple cargos and cargo priority by a mathematical model and a clustering search metaheuristic. *International Journal of Logistics Systems and Management*, 28(4):397–418.

Scarpino, A. B. A., dos Reis, W. W. F., Moraes, R. E. N., e Boeres, M. C. S. (2018). Heuristic methods to solve the berth allocation problem in iron ore export terminals. In *L Simpósio Brasileiro de Pesquisa Operacional*, p. 1–12, Rio de Janeiro.

Stahlbock, R. e Voß, S. (2008). Operations research at container terminals: a literature update. *OR spectrum*, 30(1):1–52.

Umang, N., Bierlaire, M., e Vacca, I. (2013). Exact and heuristic methods to solve the berth allocation problem in bulk ports. *Transportation Research Part E: Logistics and Transportation Review*, 54:14 – 31.

Vale (2023). Administração do complexo portuário de tubarão e praia mole (tpm). `https://www.marinha.mil.br/cpes/sites/www.marinha.mil.br.cpes/files/Porto%20de%20Tubar%C3%A3o%20e%20Praia%20Mole.pdf`. [Online; accessed 22-May-2024].

DOI: 10.59254/sbpo-2024-193664
https://proceedings.science/p/193664?lang=pt-br