UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

WILLIAN ANDERSON DE SOUSA MATEUS

UM *FRAMEWORK* PARA APOIAR ESPECIALISTAS NO PROBLEMA DE RECUPERAÇÃO DE CONTÊINERES EM TERMINAIS PORTUÁRIOS

RUSSAS

2026

WILLIAN ANDERSON DE SOUSA MATEUS

UM *FRAMEWORK* PARA APOIAR ESPECIALISTAS NO PROBLEMA DE RECUPERAÇÃO DE CONTÊINERES EM TERMINAIS PORTUÁRIOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Mayrton Dias de Queiroz

RUSSAS

2026

REVISTA CADERNOS CAJUÍNA

V. 11 N. 4 ANO 2026

Revista Interdisciplinar

e2428

Um *framework* para apoiar especialistas no problema de recuperação de contêineres em terminais portuários

A framework to support specialists in the problem of container recovery at port terminals

Un marco para apoyar a los especialistas en la problemática de la recuperación de contenedores en las terminales portuarias

DOI: 10.52641/cadcajv11i4.2428
Submitted on: 3.24.2026 | Accepted on: 3.26.2026 | Published on: 4.6.2026

Rildian Beserra Maia de Souza[1]
Pedro Henrique Ferreira da Silva[2]
Willian Anderson de Sousa Mateus[3]
Mayrton Dias de Queiroz[4]

RESUMO: Com a utilização de contêineres em terminais portuários, surge o desafio de manipulá-los com intuito de obter um menor tempo para realização das operações. Entre os problemas encontrados nesse contexto, é possível destacar o Problema de Recuperação de Contêineres que busca determinar uma sequência de operações com os contêineres realizadas em uma baia de um pátio por meio dos guindastes. Frente à quantidade de algoritmos de otimização encontrados na literatura, é necessária uma estratégia que possibilite a construção de soluções. O objetivo deste trabalho consiste em identificar uma alternativa capaz de auxiliar os especialistas na construção de soluções. Com a revisão da literatura, foi possível observar a necessidade de um framework que fosse genérico e de fácil integração. Para verificar o comportamento do framework, foram integrados dois algoritmos: o PRC_GREEDY e o PRC_CONSTRUCTIVE. Nos resultados, foi possível visualizar as soluções encontradas pela técnica e o tempo gasto ao executar as instâncias adotadas na literatura. Através do framework, foi possível visualizar e analisar as soluções geradas, que permitiu verificar o impacto das soluções aplicadas ao problema.

Palavras-chave: *framework*, problema de recuperação de contêineres, terminal portuário, otimização.

[1] Graduando em Ciência da Computação, Universidade Federal do Ceará (UFC), Russas, Ceará, Brasil.
E-mail: rildian@alu.ufc.br
[2] Graduado em Ciência da Computação, Universidade Federal do Ceará (UFC), Russas, Ceará, Brasil.
E-mail: pedrohferreira@alu.ufc.br
[3] Graduando em Engenharia de Software, Universidade Federal do Ceará (UFC), Russas, Ceará, Brasil.
E-mail: williamanderson@alu.ufc.br
[4] Doutor em Ciência da Computação, Universidade Federal do Ceará (UFC), Russas, Ceará, Brasil.
E-mail: mayrton@ufc.br

ISSN: 2448-0916

ABSTRACT: With the use of containers in port terminals, the challenge arises of handling them in order to obtain a shorter time for operations. Among the problems encountered in this context, the Container Recovery Problem stands out, which seeks to determine a sequence of operations with containers performed in a yard bay using cranes. Given the number of optimization algorithms found in the literature, a strategy that allows the construction of solutions is necessary. The objective of this work is to identify an alternative capable of assisting specialists in building solutions. Through a literature review, it was possible to observe the need for a generic and easily integrated framework. To verify the framework's behavior, two algorithms were integrated: PRC_GREEDY and PRC_CONSTRUCTIVE. The results showed the solutions found by the technique and the time spent executing the instances adopted from the literature. Through the framework, it was possible to visualize and analyze the generated solutions, which allowed us to verify the impact of the solutions applied to the problem.

Keywords: framework, container retrieval problem, port terminal, optimization.

RESUMEN: Con el uso de contenedores en terminales portuarias, surge el desafío de manejarlos para obtener un tiempo de ejecución más corto. Entre los problemas que se presentan en este contexto, destaca el Problema de Recuperación de Contenedores, que busca determinar una secuencia de operaciones con contenedores realizadas en una bahía de patio utilizando grúas. Dado el número de algoritmos de optimización encontrados en la literatura, es necesaria una estrategia que permita la construcción de soluciones. El objetivo de este trabajo es identificar una alternativa capaz de ayudar a los especialistas en la construcción de soluciones. A través de la revisión de la literatura, se pudo observar la necesidad de un marco genérico y fácilmente integrable. Para verificar el comportamiento del marco, se integraron dos algoritmos: PRC_GREEDY y PRC_CONSTRUCTIVE. En los resultados, fue posible visualizar las soluciones encontradas por la técnica y el tiempo empleado en la ejecución de las instancias adoptadas en la literatura. A través del marco, fue posible visualizar y analizar las soluciones generadas, lo que permitió verificar el impacto de las soluciones aplicadas al problema.

Palabras clave: estructura, problema de recuperación de contenedores, terminal portuaria, optimización.

1. INTRODUÇÃO

Segundo COMEX (2025), no período de janeiro a maio dos três últimos anos no Brasil, foi possível observar os seguintes valores em bilhões para exportação: US$ 1.32 - 2023; US$ 1.33 - 2024; US$ 1.35 - 2025, e para importação: US$ 0.981 - 2023; US$ 0.990 -

ISSN: 2448-0916

2024; US$ 1.10 - 2025. Diante desse crescimento gradativo, é possível destacar a importância de se encontrar soluções eficientes para melhorar as operações realizadas na cadeia logística responsável pelo transporte desses produtos até o destinatário (Oliveira *et al.*, 2026). No que se refere ao comércio exterior, o modal marítimo se destaca devido ao volume elevado de produtos transportados por viagem e ao baixo custo quando comparado ao modal aéreo. No entanto, devido à crescente demanda, surge a problemática de reduzir o máximo possível o tempo necessário para o transporte das mercadorias.

No que se refere ao modal marítimo, é possível destacar os terminais portuários responsáveis pelo recebimento e entrega dos produtos, entre os navios, caminhões e trens. Com o intuito de padronizar o espaço ocupado pelos produtos, são adotados os contêineres que possuem dimensões predefinidas, de forma que os terminais e os veículos reservem espaços padronizados, agilizando as operações e evitando o extravio durante o trajeto. Dessa forma, surge o desafio de operar os contêineres nos terminais portuários com o intuito de agilizar o transporte desses produtos (Queiroz e Silva, 2024).

Nos terminais portuários, é possível notar uma região chamada de pátio que recebe os contêineres de importação dos navios e os contêineres de exportação dos caminhões ou trens (Queiroz *et al.*, 2019). Comumente, os contêineres são inseridos no pátio na ordem de chegada, no entanto, ao serem retirados, a sequência de recuperação será distinta, visto que ela pode levar em consideração o peso do contêiner, o equilíbrio do navio, a disponibilidade dos veículos, entre outros.

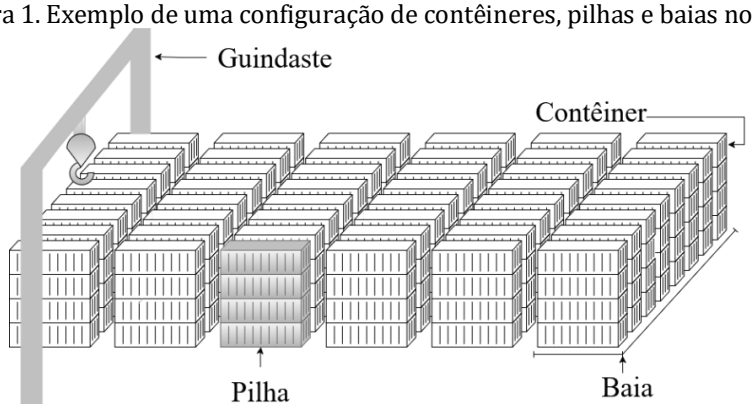
Diante desse cenário, surge o Problema de Recuperação de Contêineres, que consiste em encontrar uma sequência de operações com os contêineres com o intuito de minimizar o tempo total das operações. Portanto, se faz necessária a adoção de uma estratégia capaz de apoiar na construção de soluções para o problema. Diante dessa problemática, o objetivo deste trabalho consiste em encontrar uma estratégia, um *framework*, para auxiliar os especialistas na construção de soluções para o Problema de Sequenciamento de Contêiner em terminais portuários.

ISSN: 2448-0916

2. PROBLEMA DE RECUPERAÇÃO DE CONTÊINERES

O Problema de Recuperação de Contêineres (do inglês, *Container Retrieval Problem* – PRC) consiste em determinar uma sequência de operações com os contêineres que serão recuperados do pátio (do inglês, *yard*). Na Figura 1, é possível observar um exemplo de uma configuração de pátio, que possui um conjunto de baias. Cada baia é composta por um conjunto de pilhas, onde cada pilha é composta por um conjunto de contêineres que são dispostos um acima do outro. Adicionalmente, é importante destacar o guindaste, que ao se movimentar transversalmente consegue alterar as baias, enquanto a garra do guindaste permite que a mesma se movimente entre as pilhas de uma mesma baia.

Figura 1. Exemplo de uma configuração de contêineres, pilhas e baias no pátio.



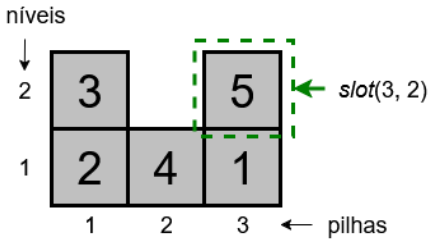
Fonte: Elaborada pelos autores.

Ao movimentar o guindaste, ele se posiciona em uma baia, assim as operações com os contêineres são realizadas na baia na qual o guindaste está posicionado. Na Figura 2, é possível notar uma representação de uma baia composta por n pilhas, m níveis e um total de C contêineres. Nesse exemplo da figura, nota-se que existem 3 pilhas e 2 níveis, onde está baia possui 5 contêineres. Cada contêiner está posicionado na baia de forma que pode ser identificado pelo *slot* (pilha, nível), ou seja, o número da pilha e o nível em que o contêiner se encontra. Quando os contêineres chegam no pátio, eles são inseridos em uma sequência conforme a ordem de chegada dos navios, caminhões ou trens, no entanto, a ordem de recuperação, geralmente, não segue essa mesma sequência. Na Figura 2, nota-

ISSN: 2448-0916

se que os quadrados que representam os contêineres possuem um número no seu interior, de forma que esse valor representa a sequência de recuperação desses contêineres, denotado de c_k onde $1 \leq k \leq C$.

Figura 2. Exemplo de uma baia.



Fonte: Elaborada pelos autores.

Dessa forma, é possível destacar dois tipos de operações que podem ser realizadas com os contêineres na baia atual:

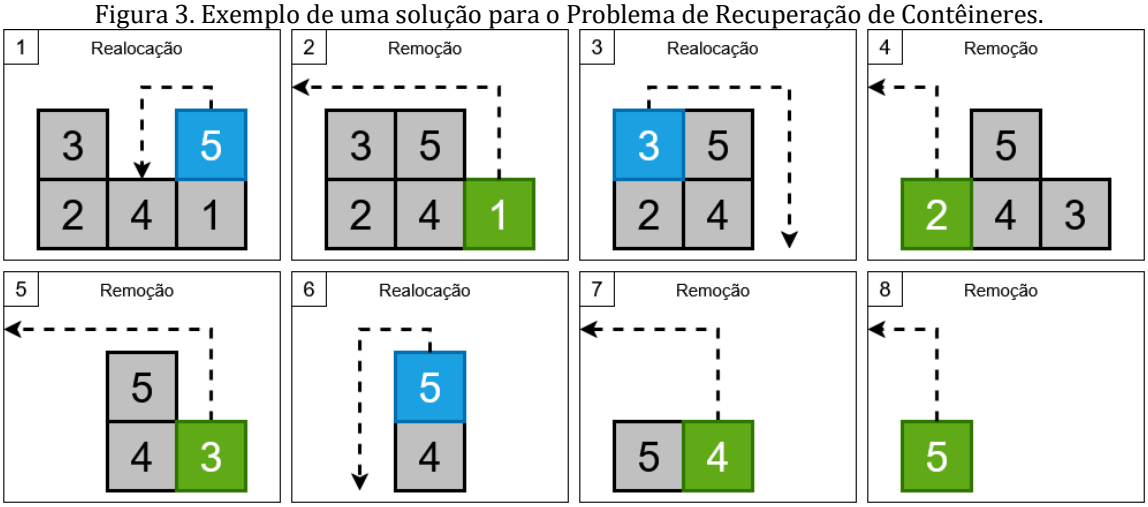
Remoção: essa operação consiste em retirar o contêiner da baia para colocá-lo em um navio, caminhão ou trem, no entanto, essa operação só pode ocorrer se o contêiner c_k, a ser removido estiver posicionado no topo da pilha e se todos os contêineres com valores inferiores, ou seja, c_1, c_2, $c_3, \ldots, c_{k-2}, c_{k-1}$ tiverem sido removidos nas operações anteriores. No exemplo da Figura 2, o contêiner 3 no *slot* (1,2), encontra-se no topo da pilha, no entanto, só poderá ser removido após a remoção dos contêineres 1 e 2.

Realocação: essa operação manipula o contêiner internamente na baia, ou seja, retira o contêiner de uma pilha e o coloca em outra pilha que compõe a baia que o guindaste está posicionado. Na Figura 2, nota-se que o contêiner 1, localizado no *slot* (3,1), será o primeiro contêiner a ser recuperado, no entanto, a operação de remoção só poderá ocorrer se o contêiner estiver no topo. Nesse exemplo, o contêiner 1 encontra-se na pilha 3, que por sua vez, tem como topo, o contêiner 5, assim, para o contêiner 1 ser removido, é preciso realocar o contêiner 5 para outra pilha, dessa forma o contêiner 1 ficará no topo da pilha 3. A operação de realocação permite que os contêineres com valores de recuperação maiores do que o próximo contêiner c_k, a ser retirado da baia, possam ser movidos para outras pilhas, permitindo a realização da próxima operação de remoção.

ISSN: 2448-0916

Diante deste cenário, surge o desafio de determinar uma sequência de operações com os contêineres de uma mesma baia de forma que a quantidade total de operações seja a menor possível. Na Figura 3, observa-se uma solução para o problema de recuperação de contêineres, tendo como instância de entrada o exemplo apresentado na Figura 2. Essa solução é composta por 8 operações, formada por 5 operações de remoção e 3 de realocação. É importante destacar que a quantidade de operações de remoção será igual à quantidade de contêineres presentes na baia, enquanto a quantidade de operações de remoção pode variar conforme a localização de cada contêiner a ser removido. Na solução apresentada na figura, nota-se que entre a remoção dos contêineres 2 e 3, não foi preciso realizar operações de remoção, visto que, esses dois contêineres já estavam no topo de suas respectivas pilhas. Já na remoção do contêiner 4, foi necessário realizar uma operação de realocação no contêiner 5 para o contêiner 4 se tornar o topo da pilha 2.

Figura 3. Exemplo de uma solução para o Problema de Recuperação de Contêineres.



Fonte: Elaborada pelos autores.

Frente a descrição do problema e uma possível solução, é importante destacar que existem diversas possibilidades para realizar uma operação de realocação, ou seja, quando se faz necessário remover um contêiner com valor de recuperação que se encontra no topo da pilha, é preciso deslocar para outra pilha. Assim, é possível notar que conforme o aumento de pilhas, níveis e contêineres presentes em uma baia o número de possibilidades aumentam, dessa forma, se faz necessário encontrar alternativas que

ISSN: 2448-0916

apõem na construção e identificação de soluções. Sendo assim, na próxima seção será apresentada a revisão da literatura para permitir identificar os principais trabalhos relacionados a esta pesquisa.

3. REVISÃO DA LITERATURA

Diante do objetivo do trabalho, foi necessária a realização de uma revisão da literatura com o intuito de identificar os principais trabalhos alinhados com essa pesquisa. Essa revisão foi organizada em três fases: Planejamento, Condução e Resultados, que serão descritas nas próximas subseções.

3.1 PLANEJAMENTO

Segundo Kitchenham *et al.* (2007), o Planejamento é a fase de definição de cada passo do protocolo, que consiste em: definição das Questões de Pesquisa; definição da *string* de busca; definição das bases de dados e definição dos critérios de inclusão e exclusão. Inicialmente, são definidas as Questões de Pesquisa (QPs) que serão respondidas durante a fase de condução. Nesse trabalho foram definidas 2 QPs:

QP1 - Qual o método de otimização adotado no trabalho?

- heurística
- exato
- outro

QP2 - Qual o algoritmo adotado?

Após a definição das QPs, definiu-se a seguinte *string* de busca: *"container retrieval problem" AND "heuristics" AND "framework".* Logo após, foram definidas as seguintes bases de dados: IEEE Xplore; Google Acadêmico e Periódicos da CAPES.

Em seguida, foram definidos os critérios de inclusão e exclusão que todos os artigos deverão satisfazer. Critérios de inclusão: estudos completos publicados em revistas ou conferências sobre o PRC; estudos teóricos ou experimentais com o objetivo de

ISSN: 2448-0916

apresentar conceitos para o entendimento da área; acessível eletronicamente. Critérios de exclusão: estudos que não estejam relacionados ao PRC; estudos que não respondem a nenhuma das questões de pesquisa; artigos duplicados, ou seja, aqueles encontrados em mais de um engenho de busca; artigos convidados, tutoriais, relatórios técnicos que não passam pelo critério de avaliação das conferências ou revistas; estudos que não estão disponíveis para *download* eletronicamente.

3.2 CONDUÇÃO

Na fase de condução, foram colocadas em prática as definições descritas no planejamento, dessa forma a *string* de busca foi inserida em cada base de dados, respeitando suas especificações. De posse dos resultados da busca, foi realizado o *download* dos estudos e aplicação dos critérios de inclusão e exclusão. Na Tabela 1, é possível identificar a quantidade de trabalhos em cada base de dados.

Tabela 1. Quantidade de artigos em cada base de dados durante busca, realização do *download* e a aplicação dos critérios de inclusão e exclusão.

Base de dados	Resultados da busca	*Download*	Inclusão/Exclusão
IEEE Xplore	4	4	1
Google Acadêmico	31	21	10
Periódicos da CAPES	5	4	1
Total	**3**	**29**	**12**

Fonte: Elaborada pelos autores.

Após os critérios de inclusão e exclusão, foram identificados 12 artigos, assim, iniciou-se a leitura de cada um com o intuito de responder às questões de pesquisa.

3.3 RESULTADOS

Nesta última fase, serão apresentados os resultados obtidos das respostas das questões de pesquisa. Na QP1, que se refere ao método de otimização adotado nos trabalhos analisados, foi possível notar quatro trabalhos que utilizaram as heurísticas. O trabalho de Zheng *et al.* (2025) utiliza uma abordagem que combina a busca heurística

ISSN: 2448-0916

com aprendizado de máquina em um modelo chamado *Intelligent Decision-Driven Model* (IDDM). Essa heurística se baseia em um processo de otimização em ciclo fechado que consiste na: identificação de alvo, análise de viabilidade, realocação dinâmica de contêineres e atualização de estado. Em Lersteau *et al.* (2021), os autores propõem uma heurística de duas etapas e com variantes para resolver o problema de empilhamento de itens. A heurística se concentra em minimizar o número de itens bloqueadores, que precisam ser realocados para acessar um item de nível menor.

O artigo de Durasević *et al.* (2024) utiliza uma heurística baseada em *ensemble learning* para o PRC, onde combinam múltiplas regras de realocação para melhorar o desempenho das soluções individuais. Em Zheng *et al.* (2018), os autores adotaram uma heurística baseada em um algoritmo de otimização de caminho aprimorado, combinado com regras heurísticas para o PRC. Em Feng (2021), o autor utiliza a heurística *Expected Minmax* (EM) e suas extensões para resolver o Problema Estocástico de Relocação de Contêineres (SCRP), buscando minimizar o número de contêineres bloqueados.

Conforme Firmino, Silva e Times (2016), os autores propuseram os algoritmos de Dijkstra e de busca A* para resolver o PRC, com o objetivo de minimizar a trajetória do guindaste, incluindo uma estratégia inteligente para explorar os nós mais promissores na árvore de busca. O trabalho de Said (2021) propõe o uso do Algoritmo de Otimização do Coronavírus (CVO), uma meta-heurística inspirada na natureza, para auxiliar na seleção de pilhas para os contêineres a serem realocados.

Díaz e Riff (2016) abordaram o PRC para minimizar o número de movimentos necessários para carregar contêineres em um navio, por meio do algoritmo GRASP. Em Li *et al.* (2024), os autores adotaram uma meta-heurística baseada no Algoritmo de Colônia de Formigas (ACO), chamada ECS-IACO, que otimiza o agendamento de saída em pátios de aço de estaleiros, com o objetivo de minimizar o consumo de energia dos guindastes.

Zhang *et al.* (2024) propuseram uma abordagem orientada por dados para resolver o PRC em cenários de incerteza, nos quais a sequência de recuperação não é totalmente conhecida. Para minimizar o número esperado de realocações, os autores utilizam um algoritmo de busca em árvore, o Algoritmo de Busca em Árvore de Monte

ISSN: 2448-0916

Carlo Adaptado (AMCTS), e uma nova heurística chamada Segurança Local e Flexibilidade Global (LSGF). Đurasević *et al.* (2025) investigaram o uso da Programação Genética (GP), uma hiper-heurística, para criar regras de realocação (RRs) para o PRC *online*.

Rathi e Upadhyay (2022) propuseram uma abordagem heurística para otimizar o tempo de manuseio de guindastes em terminais ferroviários que utilizam trens de dupla pilha. O método combina quatro heurísticas: uma para a recuperação de contêineres, uma para a realocação, outra para a atribuição de vagões e a meta-heurística *Simulated Annealing* (SA) para otimizar a sequência de visitas aos pátios.

Na QP2 foi possível observar os algoritmos adotados nos trabalhos. No Quadro 1, é possível verificar a síntese dos resultados obtidos em cada trabalho em relação às QPs.

Quadro 1. Resultados obtidos em cada Questão de Pesquisa para cada trabalho analisado.

Artigo	QP1			QP2
	Heurística	Exato	Outro	
Lersteau *et al.* (2021)	x			Busca local
Firmino *et al.* (2016)		x		Algoritmo de Dijkstra e A* *search*
Durasevic e Dumic (2024)	x			Algoritmo genético e *Ensemble Learning*
Zheng *et al.* (2018)	x			*Improved Path Optimum Algorithm*
Mehenni (2021)	x			*Coronavirus Optimization Algorithm(CVO)*
Díaz e Riff (2016)	x			GRASP
Zheng *et al.* (2025)	x			Algoritmo híbrido
Zhang *et al.* (2024)	x			*Ant Colony Optimization Algorithm (ACO)*
Rathi e Upadhyay (2022)			x	*Adapted Monte Carlo Tree Search (AMCTS)*
Durasevic e Dumic (2025)	x			*Genetic Programming (GP)*
Li *et al.* (2023)	x			*Simulated Annealing (SA)*
Feng (2023)	x			Algoritmo híbrido

Fonte: Elaborado pelos autores.

A realização da revisão da literatura permitiu identificar os principais trabalhos recentes alinhados ao Problema de Recuperação de Contêineres. De posse dos trabalhos primários, foi possível observar as estratégias e algoritmos adotados, onde destacaram-se lacunas para nortear futuras pesquisas. Dessa forma, nota-se a necessidade de uma estratégia capaz de auxiliar especialistas na construção de soluções, bem como na integração com outros algoritmos e sistemas legados. Diante deste cenário, uma alternativa é a construção de um *framework* aplicado ao PRC. Neste trabalho foi planejado e desenvolvido um *framework*, onde a descrição das etapas adotadas para sua construção

ISSN: 2448-0916

será descrita na próxima seção.

4. METODOLOGIA

Diante do objetivo deste trabalho, que consiste em identificar uma alternativa para apoiar especialistas na construção de soluções para o PRC, e de posse da revisão da literatura, surgiu a proposta de criação de um *framework* que possibilitasse a integração com outros problemas relacionados à otimização em terminais portuários. O *framework* busca facilitar a construção dos experimentos, possibilitando que os especialistas observem qual o impacto de suas soluções ao aplicá-las no problema, bem como acompanhar cada operação durante a execução da solução. Assim, esse trabalho foi organizado em três passos: Passo 1 – Revisão da Literatura; Passo 2 - Planejamento do *Framework* e Passo 3 – Resultados computacionais.

4.1 PASSO 1 – REVISÃO DA LITERATURA

O primeiro passo realizado no desenvolvimento deste trabalho foi a Revisão da Literatura com o intuito de identificar os principais trabalhos que abordam o Problema de Recuperação de Contêineres. A descrição das três fases realizadas durante a revisão pode ser observada na Seção 3. Com essa revisão, foi possível identificar a necessidade de um *framework* que pudesse auxiliar os especialistas na construção da solução para o problema. Dessa forma, na seção seguinte, será apresentado seu planejamento.

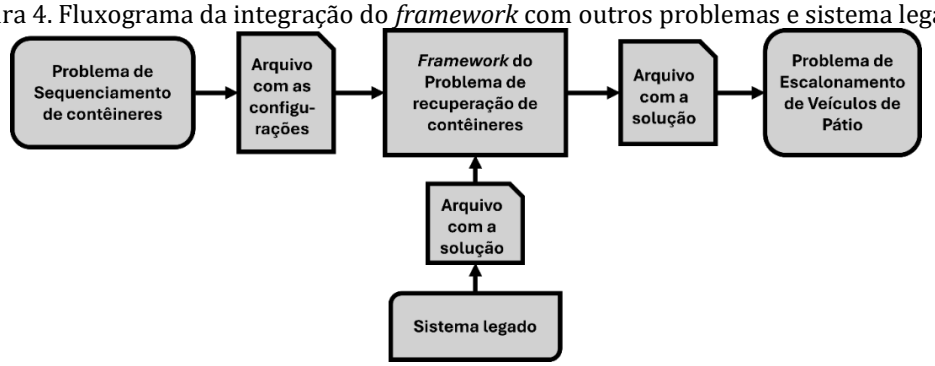
4.2 PASSO 2 – PLANEJAMENTO DO *FRAMEWORK*

Para possibilitar a integração do *framework* com sistemas legados, com algoritmos implementados em outras linguagens de programação e problemas relacionados, foi necessário investigar uma alternativa para possibilitar a comunicação entre os mesmos. Com a revisão da literatura, observou-se que os trabalhos que abordam o PRC adotam as

ISSN: 2448-0916

instâncias utilizadas em Hottung, Tanaka e Tierney (2020), sendo assim, o padrão adotado nessas instâncias foi incorporado no *framework*, para que o mesmo pudesse ser integrado com demais algoritmos e sistemas legados. Conforme mostra a Figura 4, o *framework* recebe um arquivo com as configurações que podem utilizar informações do Problema de Sequenciamento de Contêineres, bem como gerar como saída um arquivo que será usado no Problema de Escalonamento de Veículos de Pátio.

Figura 4. Fluxograma da integração do *framework* com outros problemas e sistema legado.



Fonte: Elaborada pelos autores.

4.2.1 Descrição conceitual dos algoritmos

Com o intuito de verificar a integração do *framework* com os algoritmos para solucionar o PRC, neste trabalho, foram implementados dois algoritmos: PRC_GREEDY, uma heurística determinística baseada em regras fixas de movimentação, e PRC_CONSTRUCTIVE, uma heurística construtiva que seleciona a próxima operação da solução por meio do método da roleta. No Algoritmo 1, na Figura 5, é possível observar o pseudocódigo do PRC_GREEDY. Esse algoritmo recebe como entrada uma lista de pilhas de contêineres. O dicionário localizacao, que mapeia cada contêiner à pilha em que se encontra, e o parâmetro qtd indica o número total de contêineres a serem recuperados. O algoritmo constrói como saída uma lista de operações (solucao), que registra tanto as remoções quanto as realocações executadas.

ISSN: 2448-0916

Figura 5. Pseudocódigo código do algoritmo PRC_GREEDY.

Algoritmo 1 : PRC_GREEDY(*pilhas*, *localizacao*, *qtd*)

Saída: Lista de containers removidos e realocados na ordem correta

```
1  container_atual ← 1
2  solucao ← criar_lista()
3  enquanto container_atual ≤ qtd faça
4      indice_container ← localizacao[container_atual]
5      container_topo ← desempilhar(pilhas[indice_container])
6      se container_topo = container_atual então
7          remover(localizacao, container_atual)
8          adicionar(solucao, 1, container_atual, indice_container)
9          container_atual ← container_atual + 1
10     senão
11         indice_pilha ← selecionar_pilha(pilhas, indice_container)
12         localizacao[container_topo] ← indice_pilha
13         empilhar(pilhas[indice_pilha], container_topo)
14         adicionar(solucao, 2, container_topo, indice_container, indice_pilha)
15 retorne solucao
```

Fonte: Elaborada pelos autores.

A execução ocorre no laço de repetição da linha 3, que garante a recuperação ordenada dos contêineres, respeitando as restrições do modelo de pilhas. Em cada iteração, a linha 4 identifica a pilha onde está o contêiner de interesse, e a linha 5 remove o elemento do topo desta pilha. Se o elemento retirado corresponde ao contêiner esperado, o algoritmo atualiza o dicionário localizacao (linha 7), registra a operação de retirada na solução (linha 8) e avança para o próximo contêiner (linha 9). Caso contrário, quando o contêiner desejado não está no topo, ocorre uma operação de realocação. Nesse caso, a função selecionar_pilha (linha 11) define a pilha de destino. Primeiro, verifica-se a existência de pilhas vazias, caso exista, este será o destino do atual topo. Na ausência de pilhas vazias, a escolha recai sobre a pilha de menor altura; em caso de empate, seleciona-se a pilha com maior soma dos índices dos contêineres. Em seguida, o contêiner retirado é associado à nova pilha no dicionário (linha 12), empilhado na estrutura correspondente (linha 13) e a operação de realocação é registrada na lista de solucao (linha 14). Por fim, ao término do laço, a lista de operações é retornada.

Na Figura 6, é possível observar o Algoritmo 2, correspondente ao pseudocódigo do PRC_GREEDY. O algoritmo PRC_CONSTRUCTIVE inicia realizando a etapa construtiva para cada solução (linha 3) dentro do número de (iteracoes). Para cada iteração, são

ISSN: 2448-0916

criadas cópias das pilhas e da estrutura de localizacao, preservando o estado original (linha 7–8), e é definido o contador de contêineres (container_atual) (linha 5). A cada passo do laço interno (linha 9), identifica-se a pilha que contém o contêiner atual (linha 10) e remove-se o elemento do topo dessa pilha. Caso o contêiner retirado corresponda ao contêiner desejado, ele é removido da estrutura de localizacao (linha 13), a operação é registrada na solução corrente (linha 14) e o contador é incrementado.

Quando o contêiner no topo não corresponde ao contêiner desejado, é realizada uma realocação. O algoritmo deve proceder da seguinte forma: primeiro, verifica-se a existência de pilhas vazias (linha 16); caso alguma esteja disponível, o topo é imediatamente transferido para a primeira pilha vazia encontrada (linhas 17–20). Na ausência de pilhas vazias, constrói-se uma lista de pilhas candidatas, excluindo a pilha atual (linha 22). A partir dessa lista, a pilha de destino é escolhida usando o método de roleta viciada (linha 23), que avalia o custo de movimentação considerando a quantidade de contêineres com índices maiores que o contêiner atual. Pilhas com mais contêineres de índice superior recebem menor prioridade em relação a pilhas de contêineres de índice inferior. Apesar disso, o método de roleta garante que ainda possam ser selecionadas ocasionalmente. Como os contêineres são removidos em ordem crescente (i.e., contêiner 1, depois 2, e assim por diante), contêineres de índice superior que bloqueiam outros aumentam o número de realocações necessárias.

Após determinar a pilha destino, a localização do contêiner atual é atualizada (linha 24), ele é empilhado na pilha escolhida (linha 25) e a operação de realocação é registrada na solução (linha 26). Ao final de cada iteração, o custo da solução gerada é comparado com o menor custo registrado até então (linha 27). Caso a solução corrente apresente menor custo, definido como o menor número de operações, ela substitui a melhor solução encontrada (linhas 28–29). Após a conclusão de todas as iterações, a melhor solução acumulada é retornada (linha 29).

ISSN: 2448-0916

Figura 6. Pseudocódigo código do algoritmo PRC_CONSTRUCTIVE.

Algoritmo 2 : PRC_CONSTRUCTIVE(*pilhas*, *localizacao*, *qtd*)

Saída: Melhor solução encontrada após as iterações

```
1  melhor_solucao ← criar_lista()
2  melhor_custo ← ∞
3  iteracoes ← 100
4  para i ← 1 até iteracoes faça
5      container_atual ← 1
6      solucao ← criar_lista()
7      pilhas_copia ← copia(pilhas)
8      localizacao_copia ← copia(localizacao)
9      enquanto container_atual ≤ qtd faça
10         indice_container ← localizacao_copia[container_atual]
11         container_topo ← desempilhar(pilhas_copia[indice_container])
12         se container_topo = container_atual então
13             remover(localizacao_copia, container_atual)
14             adicionar(solucao, 1, container_atual, indice_container)
15             container_atual ← container_atual + 1
16         senão se existe_pilha_vazia(pilhas_copia) então
17             indice_pilha ← buscar_pilha_vazia(pilhas_copia)
18             localizacao_copia[container_topo] ← indice_pilha
19             empilhar(pilhas_copia[indice_pilha], container_topo)
20             adicionar(solucao, 2, container_topo, indice_container, indice_pilha)
21         senão
22             lista_candidatas ← atualizar_lista_candidatas(pilhas_copia)
23             indice_pilha ← metodo_roleta(lista_candidatas)
24             localizacao_copia[container_topo] ← indice_pilha
25             empilhar(pilhas_copia[indice_pilha], container_topo)
26             adicionar(solucao, 2, container_topo, indice_container, indice_pilha)
27     se custo(solucao) < melhor_custo então
28         melhor_solucao ← solucao
29         melhor_custo ← custo(solucao)
30 retorne melhor_solucao
```

Fonte: Elaborada pelos autores.

4.3 PASSO 3 – RESULTADOS COMPUTACIONAIS

Como prova de conceito, foi implementado o *framework* e os algoritmos PRC_GREEDY e PRC_CONSTRUCTIVE com uso da linguagem de programação Python. Foi realizado um experimento para verificar os resultados obtidos em cada algoritmo ao receber as instâncias do problema como entrada. A descrição do experimento e o resultado da implementação do *framework* serão descritos na Seção 5.

ISSN: 2448-0916

5. RESULTADOS COMPUTACIONAIS

Para verificar o comportamento dos algoritmos, os mesmos receberam como entrada o conjunto de instâncias utilizado na literatura. Os experimentos realizados nesse trabalho foram executados em um computador Intel Core i5-13420H bits: 64, memória RAM de 8 GB e com o sistema operacional Pop!_OS 22.04 LTS. Os algoritmos foram implementados na linguagem de programação Python. Para a geração dos números pseudoaleatórios, foi utilizado o algoritmo Mersenne Twister. Para a criação das sementes, foram utilizadas as casas decimais de π = 3.14159265358979323846..., ou seja, a cada 6 casas decimais uma nova semente foi gerada, por exemplo: s_1 = 141592, s_2 = 653589, s_3 = 793238, ... e s_{30} = 105559.

Com o intuito de comparar os algoritmos de otimização, os mesmos receberam como entrada um *benchmark* com 1500 instâncias, classificadas como: pequena (5 pilhas e 5 níveis), média (7 pilhas e 5 níveis) e alta (10 pilhas e 5 níveis). Na Tabela 2, observam-se os resultados ao executar os algoritmos, onde foram agrupados a cada 30 instâncias e calculada a média aritmética. Como o algoritmo PRC_GREEDY não usa números aleatórios, cada instância foi executada uma vez. Já o algoritmo PRC_CONSTRUCTIVE possui o método da roleta que faz uso de números aleatórios, sendo assim cada instância foi executada 30 vezes com sementes distintas usando as casas decimais de π.

Ao comparar os resultados obtidos pelos algoritmos, é possível notar que o algoritmo PRC_CONSTRUCTIVE conseguiu apresentar soluções com uma quantidade menor de operações, quando comparado ao algoritmo PRC_GREEDY. No entanto, ao comparar o tempo de processamento, nota-se que PRC_GREEDY retornou uma solução em um tempo médio de 1 segundo. Para cada mudança de classe de instância, nota-se um aumento abrupto na quantidade total de operações em ambos os algoritmos. Extraindo a quantidade de operações da última instância da classe pequena com a primeira instância da classe média, nota-se um aumento aproximado de 36% e 37,5%, respectivamente para os algoritmos PRC_GREEDY e PRC_CONSTRUCTIVE. Com a transição da classe média para alta, extraindo os valores da última instância da classe média e os valores do primeiro

ISSN: 2448-0916

valor da classe alta, é perceptível um aumento aproximado de 38% e 40% respectivamente para os algoritmos PRC_GREEDY e PRC_CONSTRUCTIVE.

Tabela 2. Resultados do experimento com os algoritmos.

Instâncias	Pilhas-Níveis	PRC_GREEDY			PRC_CONSTRUCTIVE		
		Movimentações	Realocações	Tempo	Movimentações	Realocações	Tempo
000.dat - 029.dat	5-5	52	27	0.55427	48	23	16.85142
030.dat - 059.dat	5-5	51	26	0.52946	48	23	16.61669
060.dat - 089.dat	5-5	53	28	0.54915	49	24	17.20200
090.dat - 119.dat	5-5	51	26	0.53405	48	23	16.96614
120.dat - 149.dat	5-5	53	28	0.56023	50	25	17.66818
150.dat - 179.dat	5-5	53	28	0.50990	49	24	17.52708
180.dat - 209.dat	5-5	52	27	0.55647	48	23	17.81022
210.dat - 239.dat	5-5	52	27	0.50116	48	23	17.21503
240.dat - 269.dat	5-5	54	29	0.54094	48	23	17.55125
270.dat - 299.dat	5-5	51	26	0.49082	46	21	16.57448
300.dat - 329.dat	5-5	52	27	0.50095	47	22	16.86445
330.dat - 359.dat	5-5	53	28	0.49600	47	22	17.44142
360.dat - 389.dat	5-5	52	27	0.49610	46	21	16.51847
390.dat - 419.dat	5-5	53	28	0.50292	48	23	17.57043
420.dat - 449.dat	5-5	51	26	0.53402	47	22	16.66299
450.dat - 499.dat	5-5	52	27	0.51155	48	23	17.29186
000.dat - 029.dat	7-5	71	36	0.66607	66	31	27.75836
030.dat - 059.dat	7-5	70	35	0.62572	64	29	26.46718
060.dat - 089.dat	7-5	70	35	0.60382	65	30	26.82238
090.dat - 119.dat	7-5	73	38	0.65153	67	32	28.80746
120.dat - 149.dat	7-5	70	35	0.77488	64	29	26.23032
150.dat - 179.dat	7-5	72	37	0.67200	65	30	27.41939
180.dat - 209.dat	7-5	71	36	0.78873	65	30	27.05040
210.dat - 239.dat	7-5	69	34	0.67021	63	28	25.70227
240.dat - 269.dat	7-5	73	38	0.67347	66	31	28.16134
270.dat - 299.dat	7-5	71	36	0.88943	65	30	27.20058
300.dat - 329.dat	7-5	71	36	0.75266	65	30	26.99809
330.dat - 359.dat	7-5	73	38	0.69958	66	31	28.45629
360.dat - 389.dat	7-5	72	37	0.75185	66	31	27.69315
390.dat - 419.dat	7-5	69	34	0.61209	63	28	26.14841
420.dat - 449.dat	7-5	71	36	0.84498	65	30	27.39403
450.dat - 499.dat	7-5	70	35	0.67319	64	29	26.36002
000.dat - 029.dat	10-5	97	47	0.83939	90	40	43.80946
030.dat - 059.dat	10-5	100	50	0.84734	91	41	45.07389
060.dat - 089.dat	10-5	101	51	0.84590	94	44	47.62361
090.dat - 119.dat	10-5	98	48	0.97068	92	42	46.72265
120.dat - 149.dat	10-5	98	48	0.97653	92	42	45.81803
150.dat - 179.dat	10-5	98	48	0.84588	91	41	45.56545
180.dat - 209.dat	10-5	96	46	0.94660	90	40	42.98980
210.dat - 239.dat	10-5	98	48	0.81006	91	41	44.30045
240.dat - 269.dat	10-5	98	48	0.91443	91	41	44.99934
270.dat - 299.dat	10-5	100	50	0.84452	92	42	45.12400

ISSN: 2448-0916

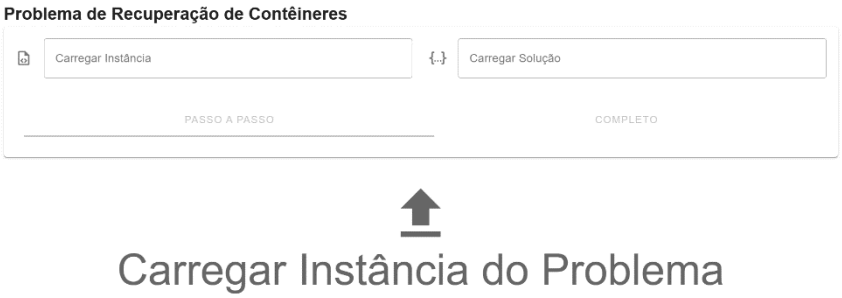
300.dat - 329.dat	10-5	99	49	0.84753	92	42	44.72323
330.dat - 359.dat	10-5	99	49	0.85633	93	43	45.13082
360.dat - 389.dat	10-5	98	48	0.91703	92	42	46.07631
390.dat - 419.dat	10-5	97	47	1.09601	90	40	44.11366
420.dat - 449.dat	10-5	101	51	1.01115	94	44	46.80196
450.dat - 499.dat	10-5	101	51	1.21267	94	44	46.70318

Fonte: Elaborada pelos autores.

5.1 EXEMPLO DE SOLUÇÃO NO *FRAMEWORK*

De posse das soluções, é importante obter uma estratégia capaz de auxiliar na visualização da sequência de operações encontrada com o intuito de verificar se a mesma é uma solução viável. Nesta subseção, será visto um exemplo de instância e uma solução para a mesma, assim será possível observar o funcionamento do *framework*. Na Figura 7, nota-se a tela inicial do *framework*.

Figura 7. Tela inicial do *framework*.



Fonte: Elaborada pelos autores.

Em seguida, será carregado o arquivo de configurações, neste exemplo, será utilizada a instância 0.dat do conjunto de entrada do experimento. Na Figura 8, nota-se um exemplo de Configuração da baia, conforme as informações de entrada que seguem a estrutura de instância utilizada na literatura, visto que esse padrão de organização das informações no arquivo já é adotado nas comparações de algoritmos.

Na Figura 9, é possível observar a tela de carregamento da solução. É importante destacar que, ao fazer o carregamento da solução, são apresentadas três regiões: a Matriz

ISSN: 2448-0916

inicial com o estado da baia de entrada; a região Atual mostrando qual a operação será realizada; e a região Próxima destacando qual será a movimentação seguinte.

Figura 8. Tela de carregamento do arquivo de configuração do *framework*.

Fonte: Elaborada pelos autores.

Figura 9. Tela de carregamento do arquivo com a solução do *framework*.

Fonte: Elaborada pelos autores.

Na Figura 10, é possível observar a tela do final da solução, dessa forma o especialista poderá verificar a cada passo se a solução está correta, ou seja, se a operação movimenta sempre os contêineres do topo da pilha e se após todas as operações a baia ficará vazia conforme o exemplo da figura.

De posse do *framework*, o especialista, seja ele um gestor, um operador do guindaste ou um desenvolvedor de uma nova solução, entre outros. Ele será capaz de observar de forma intuitiva se a solução encontrada para o problema é factível. Assim, ele não precisará validar uma solução manualmente ou por meio de simulação em papéis, que

ISSN: 2448-0916

fica cada vez mais trabalhoso conforme o aumento do tamanho das instâncias. Sendo assim, o *framework* vem como um facilitador para os especialistas tanto na construção, validação e tomada de decisão com relação às soluções para o Problema de Recuperação de Contêineres.

Figura 10. Tela destacando as últimas operações de uma solução no *framework*.

Fonte: Elaborada pelos autores.

6. CONSIDERAÇÕES FINAIS

Com a elaboração deste trabalho, foi possível observar uma alternativa capaz de auxiliar os especialistas na construção de soluções para o Problema de Recuperação de Contêineres. Através do *framework*, os especialistas podem construir suas soluções e verificar o impacto na solução final, bem como observar a solução de uma forma mais intuitiva. Adicionalmente, o *framework* possibilitou a comunicação com outros problemas de otimização em terminais portuários que podem fornecer novos dados para serem incorporados a novas técnicas de otimização, bem como podem consumir as informações geradas pelo *framework* para usar em outros problemas que tenham ligação com o PRC.

Nesse trabalho, foi possível construir dois algoritmos: o PRC_GREEDY e o PRC_CONSTRUCTIVE, onde o primeiro decide qual pilha um contêiner que irá sofrer a operação de realocação irá se movimentar, já o segundo algoritmo usa o método da roleta que seleciona a próxima operação aleatoriamente, no entanto, ele atribui maior probabilidade para as pilhas menores ou com contêineres com valor de recuperação

ISSN: 2448-0916

menor. Após os experimentos, notou-se que o primeiro algoritmo tem um tempo de processamento menor e o segundo algoritmo consegue encontrar soluções com uma quantidade menor de operações.

Durante a revisão da literatura, foi possível identificar os principais trabalhos relacionados, o que possibilitou a construção do *framework*, de forma que ele fosse genérico a ponto de integrar com outras técnicas de otimização ou sistemas legados, além de padronizar os arquivos necessários para a comunicação com outros sistemas.

Quanto às propostas de trabalhos futuros, pode-se citar: aplicar outras heurísticas no Problema de Recuperação de Contêiner para obter melhores resultados; analisar o impacto das soluções geradas pelo *framework* em um contexto global, ou seja, com outros problemas, em relação ao terminal portuário; realizar um estudo para verificar a experiência de usuário com os especialistas.

REFERÊNCIAS

COMEX. **Balança Comercial e Estatísticas de Comércio Exterior**. 2025. Disponível em: https://balanca.economia.gov.br/balanca/pg_principal_bc/principais_resultados.html. Acesso em: 01 jun. 2025.

DIAZ, C.; RIFF, M. C. New bounds for large container relocation instances using grasp. In: **2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)**. IEEE, 2016. p. 343-349.

ĐURASEVIĆ, M.; ĐUMIĆ, M.; GIL-GALA, F. J. Constructing ensembles of automatically designed relocation rules for the container relocation problem. In: 2024 **IEEE Congress on Evolutionary Computation (CEC)**. IEEE, 2024. p. 1-8.

ĐURASEVIĆ, M.; ĐUMIĆ, M.; GIL-GALA, F. J. Designing relocation rules with genetic programming for the online container relocation problem. In: **2024 IEEE Congress on Evolutionary Computation (CEC)**. IEEE, 2025. p. 1-8.

FENG, Y. **Yard Operations Optimisation at Maritime Terminals Under Uncertain Truck Arrivals: Container Stacking, Retrieval, and Relocation**. The University of Liverpool (United Kingdom), 2021.

FIRMINO, A. S.; SILVA, R. M. A.; TIMES, V. C. An exact approach for the container retrieval problem to reduce crane's trajectory. In: **2016 IEEE 19th international conference on intelligent transportation systems (ITSC).** IEEE, 2016. p. 933-938.

ISSN: 2448-0916

HOTTUNG, A.; TANAKA, S.; TIERNEY, K. Deep learning assisted heuristic tree search for the container pre-marshalling problem. **Computers & Operations Research**, v. 113, p. 104781, 2020.

LERSTEAU, C. *et al.* Solving the problem of stacking goods: mathematical model, heuristics and a case study in container stacking in ports. **IEEE Access**, v. 9, p. 25330-25343, 2021. doi: 10.1109/ACCESS.2021.3052945.

LI, J. *et al.* Optimising outbound scheduling in shipyard steel stockyard. **Ships and Offshore Structures**, p. 1-13, 2024.

OLIVEIRA, R. D.; OLIVEIRA, L. A. de; PAIVA, I. M.. Robótica colaborativa na logística: humanos e robôs trabalhando em conjunto nos armazéns inteligentes. **Cadernos Cajuína**, *[S. l.]*, v. 11, n. 1, p. e1565, 2026. DOI: 10.52641/cadcajv11i1.1565.

QUEIROZ, M. D. de; SILVA, R. M. de A. Framework para apoiar especialistas no problema de sequenciamento de contêiner em terminais portuários. **Revista Principia**, [S. l.], v. 61, n. 1, p. 31–57, 2024. DOI: 10.18265/1517-0306a2022id6730.

QUEIROZ, M. D.; *et al.* Um Framework para Apoiar Especialistas no Estudo da Eficiência Energética em Trens Urbanos. In: **Simpósio Brasileiro de Sistemas de Informação (SBSI)**, 15., 2019, Aracajú. Anais [...]. Porto Alegre. p. 32-39.

RATHI, P.; UPADHYAY, A. Container retrieval and wagon assignment planning at container rail terminals. **Computers & Industrial Engineering**, v. 172, p. 108626, 2022.

SAID, M. **Résolution du problème de stockage de conteneurs dans un port par un algorithme d'optimisation du coronavirus**. 2021. 38 f. Tese - Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie, Université Mohamed Khider – BISKRA, Algérie, 2021.

ZHANG, Z. *et al.* A data-driven approach to solving the container relocation problem with uncertainties. **Advanced Engineering Informatics**, v. 65, p. 103112, 2025.

ZHENG, S. *et al.* An improved path optimum algorithm for container relocation problems in port terminals worldwide. **Journal of Coastal Research**, v. 34, n. 3, p. 752-765, 2018.

ZHENG, S. *et al.* Research on artificial intelligence-driven container relocation problem for green ports. **Frontiers in Marine Science**, v. 12, p. 1614356, 2025.

ISSN: 2448-0916