PROGRAMAÇÃO GENÉTICA E EVOLUTIVA APLICADAS AO CARREGAMENTO E DESCARREGAMENTO DE CONTÊINERES EM TERMINAIS PORTUÁRIOS

Mateus Roberge Warmling[1], Merisandra Côrtes de Mattos Garcia[2]

Resumo: O transporte marítimo é a base do comércio internacional e da economia global, sendo que parte desse transporte é realizado por meio de navios porta-contêineres, área que cresce a cada ano e deve se tornar cada vez mais relevante. Dentre a navegação por contêineres, existe o problema de carga e descarga de contêineres nos navios, em que planos ineficientes aumentam o custo da operação de manejo desses contêineres, gerando trocas desnecessárias durante o curso do navio e sua parada nos portos, visto que só é possível acessar um contêiner pela parte superior da pilha onde ele se encontra. Este trabalho tem como objetivo implementar dois algoritmos evolucionários, o algoritmo genético e o baseado em estratégias evolutivas, e comparar a sua performance quando aplicados a este problema, também é proposta uma implementação em que as operações de evolução dos algoritmos são aplicadas diretamente ao planejamento de cargas do navio. As implementações demonstram uma superioridade do algoritmo genético aplicado ao problema em números de trocas desnecessárias e tempo de execução, além de uma superioridade da implementação por regras encontrada na literatura ao problema de carregamento e descarregamento de contêineres.
Palavras-chave: Transporte marítimo. Algoritmos evolucionários. Heurística.

ABSTRACT: Maritime transport is the basis of international trade and the global economy, and part of this transport is carried out by container ships, an area that grows every year and is expected to become increasingly relevant. Among container shipping, there is the problem of loading and unloading containers on ships, where inefficient plans increase the cost of the operation of handling these containers, generating unnecessary exchanges during the course of the ship and its stops in ports, since it is only possible to access a container from the top of the stack where it is

1 Curso de Ciência da Computação, Grupo de Pesquisa em Inteligência Artificial Aplicada, Universidade do Extremo Sul Catarinense (Unesc), Criciúma - Santa Catarina - Brasil. mateuswarmling@unesc.net.
2 Orientadora, Curso de Ciência da Computação, Grupo de Pesquisa em Inteligência Artificial Aplicada, Universidade do Extremo Sul Catarinense (Unesc), Criciúma - Santa Catarina - Brasil. mem@unesc.net.

placed. This work aims to implement two evolutionary algorithms, the genetic algorithm and the evolution strategies, and compare their performance when applied to this problem, it is also proposed an implementation where the evolutionary operations of the algorithms are applied directly to the ship's cargo planning. The implementations show a superiority of the genetic algorithm applied to the problem in numbers of unnecessary changes and execution time, and a superiority of the rule-based implementation found in the literature to the container loading and unloading problem.

Keywords: Maritime transport. Evolutionary algorithms. Heuristics.

1 INTRODUÇÃO

O estudo *Review of Maritime Transport* realizado em 2021 pela *United Nations Conference on Trade and Development* (UNCTAD), aponta que o transporte marítimo é a base do comércio internacional e da economia global, tendo-se mais de 80% do volume do comércio internacional de mercadorias transportado pelo mar, e esta porcentagem é superior na maioria dos países em desenvolvimento (UNCTAD, 2021, tradução nossa). Além disso, dentre as movimentações no próprio país tem-se a cabotagem, prática de navegação entre portos nacionais realizada visualizando-se a costa, que é responsável por aproximadamente 11% das movimentações no Brasil (BNDES, 2022), sendo que uma parte considerável deste transporte é realizado por meio de navios porta-contêineres.

O transporte de contêineres vem crescendo significativamente desde 1967, projetando-se que nos próximos anos essa forma de transporte se torne ainda mais relevante e automatizada (SAXON; STONE, 2017), porém, ao analisar a automatização dos navios porta-contêineres, esbarra-se no Problema de Carga e Descarga de Contêiner (*Container Stowage Problem*).

Em um navio porta-contêineres, os contêineres são transportados em formato de pilha, sendo assim, só podem ser retirados pela parte de cima do navio, e em ordem contrária a que foram colocados, o que ocasionalmente gera a necessidade da mudança de posição de um contêiner que esteja atrapalhando o desembarque de outro. Ademais, essa movimentação atrasa a operação de carga e descarga, não atendendo o interesse dos portos que visam minimizar o tempo desta atividade a fim de maximizar o número de navios atendidos e os lucros. Ao mesmo tempo, as

companhias marítimas também têm interesse de diminuir o tempo gasto em portos e os custos envolvidos (AVRIEL; PENN; SHPIRER, 2000; BILICAN; EVREN; KARATAS, 2020, tradução nossa), sendo assim, o problema de carga e descarga de contêiner ilustra o cenário de um navio que passa por diversos portos, descarregando os contêineres destinados ao porto atual, e carregando contêineres para portos futuros. De acordo com Benedito e Santos (2015), são necessárias alternativas que busquem uma solução para esse problema, minimizando-se o número de trocas durante a operação. Este problema é considerado NP-Completo, pois a identificação da sequência otimizada para o carregamento de contêineres é uma tarefa difícil computacionalmente (SAIKIA *et al*., 2018, tradução nossa).

Na resolução de problemas deste tipo, empregam-se meta-heurísticas, que propõem uma solução próxima da ideal em pouco tempo, enquanto os algoritmos exatos buscam a solução, exigindo um tempo de execução ou recursos computacionais impraticáveis para problemas complexos (DOKEROGLU *et al*., 2019, tradução nossa). Dentre os tipos de meta-heurísticas, comumente são utilizados os algoritmos evolucionários. Os algoritmos evolucionários são entendidos como algoritmos de busca direta estocástica baseada em população, que em certo nível, imitam a evolução natural (BARTZ-BEIELSTEIN *et al*., 2014, tradução nossa). Os algoritmos mantêm uma população de pontos de busca (conhecidos como candidatos a solução, indivíduos, cromossomos ou agentes), que são gerados aleatoriamente, e evoluem iterativamente ao longo de uma série de gerações ao aplicar operadores de variação e seleção. Os operadores de variação geram mudanças para os membros da população, produzindo movimentações no campo de busca. A cada geração o valor objetivo dos pontos de busca são calculados, aqueles de menor valor são removidos pelos operadores de seleção, restando apenas os melhores e, a partir destes, derivando-se novos pontos (CORNE; LONES, 2018, tradução nossa).

Diversas variantes dos algoritmos evolucionários são empregadas em múltiplos trabalhos relacionados ao Problema de Carga e Descarga de Contêiner, seja no uso combinado entre algoritmos de colônia de formigas e algoritmos genéticos (BENEDITO; SANTOS, 2015, tradução nossa), na comparação de algoritmos genéticos com colônia de abelhas (CARRARO; CHIWIACOWSKY; GÓMEZ, 2013) ou relacionados a algoritmos evolucionários e aprendizagem por reforço (SAIKIA *et al*., 2018; TIJJANI; OZKAYA, 2014, tradução nossa). No entanto, são raros os trabalhos que comparam a performance de dois algoritmos evolucionários voltados ao problema

de carga e descarga de contêiner, o que demonstra uma lacuna nessa área de pesquisa.

O problema de carga e descarga de contêineres tange a área de sistemas produtivos, pois os terminais portuários atendem navios continuamente e buscam maximizar a quantidade desses atendimentos. A resolução desse problema relaciona-se com os Objetivos de Desenvolvimento Sustentável (ODS) definidos pela Organização das Nações Unidas (ONU, 2022), pois pode influenciar em diferentes ODS, como por exemplo, no ODS 8 de Trabalho Decente e Crescimento Econômico e no ODS 9 de Indústria, Inovação e Infraestrutura, pois esta pesquisa volta-se à modernização e otimização de um importante processo da indústria marítima.

Mediante o exposto, esta pesquisa visa avaliar a performance entre duas das variações, consideradas por Corne e Lones (2018), como das mais populares da abordagem evolucionária, o algoritmo genético e o algoritmo baseado em estratégias evolutivas. A análise da performance de ambos os algoritmos aplicados ao problema de carga e descarga de contêineres pode proporcionar uma solução ideal para o problema, buscando a otimização do processo de planejamento de contêineres em navios porta-contêineres.

2 TRABALHOS CORRELATOS

Devido a extensa literatura focada no problema de carga e descarga de contêineres, existem diversas formas de modelagem do problema para representação de um navio porta-contêineres.

Lima (2011) aplicou a heurística *Beam Search* para o carregamento e descarregamento de contêineres junto de uma representação por regras. Foram criadas ao todo quatro regras, duas para carregamento e duas para descarregamento. O programa foi desenvolvido em Java e gerou um resultado satisfatório, retornando soluções que eram capazes de suprir o problema proposto, junto de uma interface gráfica para facilitar a interação com o usuário.

Azevedo *et al*. (2011) apresenta uma solução via *Simulated Annealing* aplicada ao carregamento de contêineres comparada a uma solução por meio de *Beam Search*. Foi utilizada a representação por regras, e foram criadas 12 combinações, juntando quatro regras para o carregamento e três para o descarregamento. Foi utilizada a linguagem C para a criação do programa, sendo

observada uma superioridade da solução por *Simulated Annealing*, tendo apresentado resultados iguais ou superiores a solução por *Beam Search*

Carraro, Chiwiacowsky e Gómez (2013) aplicaram meta-heurísticas por meio de algoritmos genéticos e de colônia de abelhas no problema de carregamento de navios contêineres. A representação por regras foi utilizada, combinando-se uma regra de carregamento com uma de descarregamento, o que gerou dezoito combinações. O desenvolvimento foi na ferramenta *Matlab*, constatando-se uma superioridade do algoritmo genético quando utilizadas quatro combinações de regras e resultados semelhantes quando ambos os algoritmos empregaram dezoito combinações. Neste último caso, o algoritmo genético apresenta resultados levemente superior e necessita de um menor esforço computacional.

Benedito e Santos (2015) aplicam algoritmos genéticos e de colônia de formigas ao problema de carga e descarga de contêineres. Para isso, utilizam a representação por regras, unindo-se novas regras as já presentes na literatura, sendo que estas regras novas consideram as informações da situação atual do navio, como o número de contêineres nas pilhas ou o destino dos contêineres no topo. De acordo com os resultados obtidos, o trabalho demonstrou sucesso ao aplicar novas regras aos algoritmos utilizados na resolução do problema, que apesar de tornar a execução do algoritmo mais lenta, demonstrou ganhos em relação ao uso apenas das regras antes propostas, visto que os resultados encontrados pelos autores são superiores aos da literatura. Além disso, o uso da heurística de colônia de formigas mostrou que existe a possibilidade de exploração de novos algoritmos a esta problemática, visto que teve resultados comparáveis em tempo e número de trocas, quando relacionados aos algoritmos já utilizados.

Saikia *et al*. (2018) propõem a aplicação de algoritmos evolucionários e da aprendizagem por reforço para o carregamento de contêineres. No desenvolvimento empregaram a biblioteca *OpenAI Gym* e o algoritmo foi treinado com soluções boas, porém não ótimas, com base em uma política evolucionária de recompensa. No estudo concluíram que o agente treinado com aprendizagem por reforço teve resultados superiores as soluções heurísticas encontradas na literatura, adaptando-se melhor a problemas inéditos do que o agente treinado sem exemplos ou das soluções heurísticas (SAIKIA *et al*., 2018).

Apesar do trabalho de Saikia *et al*. (2018) sugerir uma superioridade das soluções de aprendizagem por reforço quando comparada a soluções heurísticas,

esse trabalho optou por comparar dois algoritmos evolucionários, pois como citado anteriormente, há carência de estudos que realizassem este tipo de comparação, levando ao questionamento se as soluções da literatura estão considerando diferentes vertentes que possam levar a solução deste problema de otimização.

O quadro 1 apresenta uma comparação entre os trabalhos citados e esta pesquisa.

Quadro 1 – Comparação entre as pesquisas

Trabalho	**Algoritmos**	**Plataforma ou Linguagem**	**Hardware utilizado**	**Resultados**
Lima (2011)	*Beam Search* (representação por regras)	Java	N/A	Resultados satisfatório, junto de interface gráfica
Azevedo *et al*. (2011)	*Simulated Annealing* (representação por regras)	C	Intel Core 2Duo 2.20 GHz (E4500), 2GB RAM e Sistema Operacional Windows XP	Resultados superiores ou iguais a implementação de *Beam Search*
Carraro, Chiwiacowsky e Gómez (2013)	Algoritmo genético e Colônia de abelhas (representação por regras em ambos)	MatLab	Dual Core 2,0GHz, memória RAM de 3GB e Sistema Operacional Windows 7	Resultados superiores do algoritmo genético com 4 regras, ou levemente superior com 18 regras
Benedito e Santos (2015)	Algoritmo genético e Colônia de formigas (representação por regras em ambos)	C++	Intel(R) Core(TM) i7 CPU 2.83GHz, 8 GB de RAM e sistema operacional Windows 7	Resultados superiores aos da literatura, com uma execução mais lenta, porém com melhores soluções
Saikia *et al*. (2018)	Algoritmos evolucionários e de aprendizagem por reforço	OpenAI Gym	N/A	Superioridade da aprendizagem por reforço em comparação com as soluções heurísticas
Esta pesquisa	Algoritmo genético e baseado em estratégias evolutivas	Python	AMD Ryzen™ 5 7600X 4.7GHz, memória RAM de 16GB e sistema operacional Windows 10	Resultados inferiores a representação por regras (maior esforço computacional). Superioridade do algoritmo genético.

Fonte: Do autor (2023).

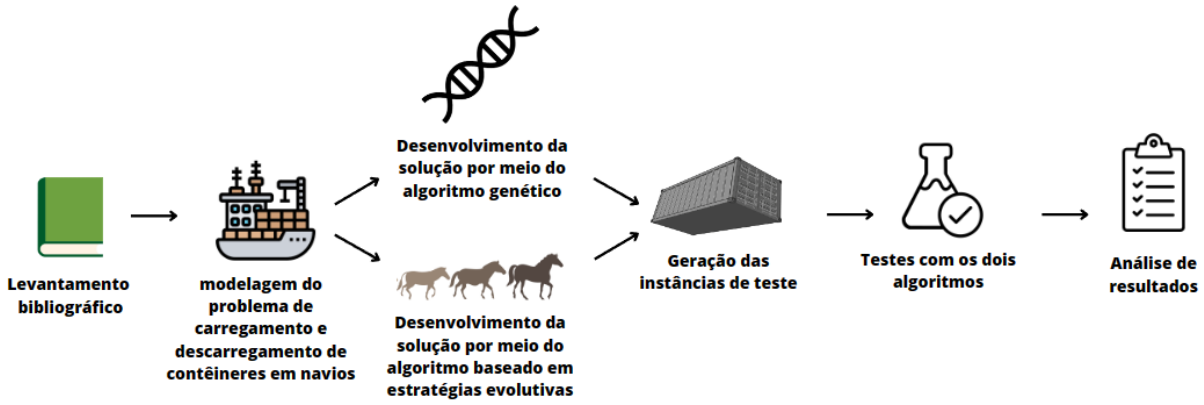
3 MATERIAIS E MÉTODOS

Esta pesquisa compreende a aplicação de programação genética e evolutiva ao carregamento e descarregamento de contêineres em navios nos terminais portuários, avaliando a performance do algoritmo genético e do algoritmo baseado em estratégias evolutivas na resolução deste problema. Para isso, analisou-se o tempo necessário para realizar o planejamento de cargas e o número de trocas de carga no descarregamento em rotas com dez ou quinze portos, e diferentes tipos de matrizes de transporte, sendo estas matrizes com distância curta, mista e longa.

A abordagem da pesquisa é descrita como quantitativa, sendo quanto à natureza aplicada e de base tecnológica, visto que gera o conhecimento relacionado a análise da aplicação das duas meta-heurísticas ao problema estudado. Em relação aos objetivos é caracterizada como uma pesquisa descritiva, pois possibilita novas visões acerca de uma realidade conhecida (NUNES; NASCIMENTO; LUZ, 2016), delimitando-se métodos, modelos e teorias para a coleta e interpretação dos dados (TRIVIÑOS, 2011). Quanto aos procedimentos, compreende a pesquisa bibliográfica, visto que conhece as contribuições científicas realizadas sobre o assunto, e a pesquisa experimental, já que manipula diversos dados referentes ao modelo do problema estudado em diversas condições.

A figura 1 apresenta as etapas realizadas no desenvolvimento desta pesquisa, sendo elas: levantamento bibliográfico, modelagem do problema de carregamento e descarregamento de contêineres em navios; desenvolvimento da solução por meio do algoritmo genético; desenvolvimento da solução por meio do algoritmo baseado em estratégias evolutivas, geração das instâncias de teste, testes com os dois algoritmos e análise de resultados. No desenvolvimento da solução por ambos os algoritmos se utilizou a linguagem de programação *Python* em sua versão 3.11.2.

Figura 1 – Etapas de desenvolvimento da pesquisa



Fonte: Do autor (2023).

3.1 Modelagem do problema de carregamento e descarregamento de contêineres

O problema de carga e descarga de contêiner consiste em um navio que passa por diferentes terminais portuários, descarregando os contêineres destinados ao terminal atual, e carregando contêineres para os portos futuros (BILICAN; EVREN; KARATAS, 2020). Segundo os mesmos autores, consiste em um problema de atribuição que visa diminuir o tempo gasto no terminal e os custos deste processo, sendo que para isto, conforme Euchi *et al*. (2016), é necessário organizar no navio os contêineres em lugares ideais.

A literatura aborda diferentes modelagens do problema para a representação de um navio porta-contêineres, sendo que nesta pesquisa, foi utilizada a descrição do problema de Avriel *et al*. (1998), uma das mais utilizadas em diferentes estudos da área, como por exemplos nos trabalhos de Lima (2011), Azevedo *et al*. (2011), Carraro, Chiwiacowsky e Gómez (2013) e Benedito e Santos (2015), nesta definição o navio contêiner é composto por linhas ($r = 1,...,R$), sendo o valor 1 a base do navio e R o topo, e colunas ($c = 1,...,C$), em que 1 representa a primeira coluna da esquerda e C a última da direita. Assim, o navio é capaz de comportar, simultaneamente, $X = R \times C$ contêineres.

Dada a capacidade do navio, ele deve viajar do porto i até o porto $i+x$, começando e terminando vazio no último porto, onde irá descarregar os contêineres restantes. A cada ponto de parada serão adicionados contêineres para portos futuros, que só podem ser carregados pela parte de cima das pilhas, e quando chegarem a

seus destinos, devem ser descarregados e não voltarem para o navio, sendo que a operação deve conter o mínimo de trocas desnecessárias possível (CARRARO; CHIWIACOWSKY; GÓMEZ, 2013).

Na reprodução do modelo, considerando-se que o navio passa por diversos portos, carregando e descarregando contêineres, esta pesquisa adotou a utilização de uma lista para representar o planejamento das cargas, contendo objetos de uma classe criada para os contêineres, composta pelo destino e pela posição atual do contêiner no navio. Também foram adicionadas algumas restrições em relação ao posicionamento dos contêineres no navio e ao cálculo do número de trocas desnecessárias de contêineres de cada planejamento.

3.1.1 Posicionamento dos contêineres

A fim de que exista uma verossimilhança com a realidade e os planejamentos das cargas sejam fisicamente possíveis, as seguintes restrições são aplicadas ao posicionamento dos contêineres no navio:

a) **restrição 1:** garante que nenhum contêiner seja posicionado em uma coordenada que já foi preenchida. Para isso, mantém-se uma lista das posições ocupadas no navio, e anteriormente ao carregamento de qualquer contêiner é feita a validação, verificando se a nova posição já está em uso. Caso a resposta seja positiva, gera novas coordenadas, caso contrário o contêiner é alocado na nova posição;
b) **restrição 2:** garante que o contêiner seja carregado no nível mais baixo da coluna selecionada. Ao carregar um contêiner no navio, são verificadas se as posições abaixo da gerada já estão preenchidas, alocando um contêiner na coordenada referente a posição mais inferior disponível. Isso é feito para que o contêiner não esteja posicionado sem uma sustentação.

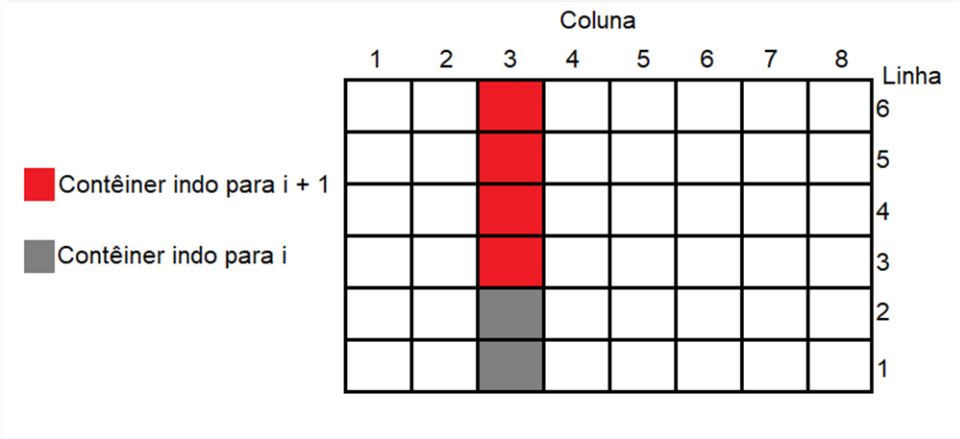
3.1.2 Número de trocas desnecessárias de contêineres

O cálculo do número de trocas desnecessárias de contêineres realizado é baseado em quantas operações desnecessárias são realizadas para o

descarregamento de um contêiner, com base em mudanças de posição de contêineres que não serão descarregados no porto em que o navio se encontra.

Na figura 2 é possível observar uma situação em que serão necessárias quatro trocas devido ao posicionamento dos contêineres do porto futuro *i+1*, que deverão ser reorganizados para o descarregamento no porto atual *i*.

Figura 2 – Representação de contêineres que terão que ser reorganizados



Fonte: Do autor (2023).

Ao chegar em um terminal portuário, deve-se analisar se será necessária a troca de posição de contêineres. Para isso, verifica-se quais contêineres devem ser descarregados e sua posição atual, bem como o estado atual do navio, sendo este as posições onde todos os contêineres estão alocados. Posteriormente, para cada uma das cargas que serão retiradas, são analisadas se as posições acimas estão sendo ocupadas. No caso de a resposta ser positiva, é verificado se os contêineres que estão preenchendo essas coordenadas também são destinados ao porto atual. Se sim, tanto a carga que iniciou a verificação, quanto as que estão bloqueando a passagem são descarregadas sem aumentar o número de trocas.

Caso exista algum contêiner bloqueando a operação e que não deve ser descarregado no porto atual, ele é retirado e realocado para outra posição do navio, permitindo que o descarregamento seja concluído. A operação de reorganização completa de um contêiner, que está obstruindo o processo de descarregamento, conta como uma troca, repetindo-se quantas vezes se fizerem necessárias. Este processo se repete em todos os portos em que o navio passar, sendo que a soma final de todas as operações é considerada o número de trocas do planejamento de cargas.

3.2 Desenvolvimento da solução por meio do algoritmo genético

Nos algoritmos genéticos os indivíduos são comumente chamados de cromossomos e costumam empregar *strings* binárias, com o valor real sendo codificado por símbolos binários, atualmente, as variáveis normalmente são mapeadas a um domínio de otimização, com as variáveis tendo valores reais que devem ser otimizados (CORNE; LONES, 2018). Na figura 3 tem-se o pseudocódigo de um algoritmo genético, que compreende o fluxo básico do funcionamento clássico dessa meta-heurística, iniciando com uma população, que é avaliada para verificar a sua aptidão, após isso é criado um laço em que são realizadas as operações de mutação, cruza e seleção dos descendentes com base na aptidão, e após o laço encerrar, é retornada a melhor solução. A implementação do algoritmo neste trabalho é detalhada a seguir:

Figura 3 – Pseudocódigo de um algoritmo genético

```
1   pop = InicializaPopulação(tamanhoPopulação, dimensãoDoProblema);
2   avaliaPopulação(população);
3   melhorSolução = getMelhorSolução(população);
4   enquanto testeParaEncerrar == falso faça  {
5       descendente = 0;
6       progenitores = seleçãoDaCruza(população);
7       para progenitor1, progenitor2 ∈ progenitores faça {
8           (descendente1, descendente2) = cruza(progenitor1, progenitor2);
9           descendente1 = mutação(descendente1)'
10          descendente2 = mutação(descendente2);
11          descendente = {descendente} ∪ descendente1 ∪ descendente2;
12      }
13      avaliaPopulação(descendente);
14      melhorSolução = getMelhorSolução(descendente);
15      população = substituir(população, descendente);
16  }
17  Retorna(melhorSolução)
```

Fonte: Adaptado de Bartz-Beielstein *et al*. (2014)

Os cromossomos, que também podem ser chamados de indivíduos, são os planejamentos das cargas nos navios, assim, todas as operações relacionadas aos algoritmos genéticos são feitas com base na performance de cada planejamento, que juntos compõe a população do algoritmo.

A primeira geração da população é criada com base na lista de destinos de contêineres do primeiro carregamento do navio. Cada contêiner recebe um posicionamento aleatório no planejamento, posteriormente, aplica-se a operação de

cruzamento, gerando os descendentes que passam pelo processo de mutação, substituindo a antiga população.

Após cada ciclo a população é avaliada, se algum dos cromossomos alcançou a solução ideal (zero trocas) o algoritmo é encerrado. Caso contrário, o ciclo de gerações continua, com o máximo definido em 100 gerações. Os métodos empregados de cruzamento e mutação são descritos a seguir.

3.2.1 Cruzamento

Na seleção dos progenitores a serem utilizados na fase de cruzamento, foi aplicada a seleção por torneio, em que se escolhe uma quantidade de cromossomos aleatórios da população. Na sequência, define-se o cromossomo com a melhor aptidão, conforme as trocas totais do planejamento de cargas em que o cromossomo com a menor quantidade de trocas desnecessárias tem a melhor aptidão, após definir qual o cromossomo mais apto, ele é considerado o progenitor desta fase de cruzamento. O processo se repete até existirem progenitores suficientes para uma nova população (BARTZ-BEIELSTEIN *et al*., 2014).

O método de cruzamento empregado no algoritmo foi o *one-point crossover*, em que é selecionado um ponto dos cromossomos, e a partir disso ambos os progenitores trocam segmentos, gerando dois novos descendentes (BARTZ-BEIELSTEIN *et al*., 2014).

Nesta pesquisa, os cromossomos consistem no planejamento das cargas do navio, com a posição de cada contêiner, após cada cruzamento é realizada uma validação para garantir que nenhuma posição está sendo ocupada mais de uma vez, realocando um dos contêineres caso ocorra um conflito. Não há necessidade de validação dos destinos, visto que a lista de posições é sempre ordenada com base neles. Assim, muda-se apenas a coordenada de cada contêiner entre os planejamentos, e não o destino.

3.2.2 Mutação

De acordo com Bartz-Beielstein *et al*. (2014), a mutação é uma estratégia utilizada para que novas informações sejam adicionadas a população, fazendo com

que as buscas não fiquem estagnadas, um método de mutação para cromossomos baseados em *bits* é a inversão de um bit em uma posição aleatória.

Considerando-se que o indivíduo neste trabalho é uma lista com posições definidas, a operação de mutação utilizada é uma adaptação deste método, em que são sorteados dois contêineres aleatórios do cromossomo, e após isso a posição de ambos é invertida. A taxa de mutação foi definida em 20% de chance para que o indivíduo seja modificado.

3.3 Desenvolvimento da solução por meio do algoritmo baseado em estratégias evolutivas

A implementação padrão das estratégias evolutivas se aproxima dos algoritmos genéticos, utilizando populações e cruzamentos, mas ainda existem diferenças, como na forma que o baseado em estratégias evolutivas aplica as mutações, em que as variáveis de decisão também são incluídas na mutação (CORNE; LONES, 2018). Nas estratégias evolutivas é utilizado mais de um indivíduo para gerar uma nova solução, sendo os progenitores escolhidos de maneira determinística, fazendo com que os pontos de busca mais aptos sejam usados para gerar novos indivíduos (BARTZ-BEIELSTEIN *et al*., 2014).

Na figura 4 é possível observar o pseudocódigo de um algoritmo baseado em estratégias evolutivas, em que é iniciado uma população, que é avaliada para verificar a sua aptidão, após isso é criado um laço que passa por todos os indivíduos da população, selecionando os com melhor aptidão para a cruza, então são realizadas as operações de recombinação e mutação, assim que o laço encerra, é retornada a melhor solução pós reavaliações. A implementação do algoritmo neste trabalho é detalhada a seguir:

Figura 4 – Pseudocódigo de um algoritmo de Estratégias Evolutivas

```
1   pop = InicializaPopulação(tamanhoPopulação, dimensãoDoProblema);
2   avaliaPopulação(população);
3   melhorSolução = getMelhorSolução(população);
4   enquanto testeParaEncerrar == falso faça {
5       descendente = 0;
6       para i = 0 até quantidadeDeDescendentes faça {
7           populaçãoParaCruza = seleçãoDaCruza(população);
8           descendente¡ = recombinar(populaçãoParaCruza);
9           descendente¡ = mutação(descendente¡);
10          descendente = {descendente} ∪ descendente¡;
11      }
12      avaliaPopulação(descendente);
13      população = seleçãoPorAmbiente(população);
14      melhorSolução = getMelhorSolução(população);
15  }
16  Retorna(melhorSolução)
```

Fonte: Adaptado de Bartz-Beielstein *et al*. (2014)

Diante da proximidade entre as duas vertentes de algoritmos evolucionários empregados nesta pesquisa, a implementação do algoritmo baseado em estratégias evolutivas compartilha algumas semelhanças com a do algoritmo genético. Em ambos, a primeira população é gerada da mesma forma, em que são alocadas posições aleatórias no navio para os contêineres do primeiro carregamento, com exceção de que cada cromossomo das estratégias evolutivas também contém sua própria taxa de mutação, junto do planejamento de cargas do navio.

Da segunda geração em diante, a implementação passa pelo mesmo ciclo de cruzamento, mutação e avaliação, mas contando com particularidades, como o número de progenitores e os parâmetros envolvidos na mutação, referente a essas operações.

3.3.1 Cruzamento

Para a seleção dos progenitores, os algoritmos baseados em estratégias evolutivas costumam utilizar mais do que dois indivíduos para criar um descendente, além disso, outra particularidade deste algoritmo em relação ao genético é a tendência de utilizar métodos de seleção determinístico, ao invés de probabilístico, fazendo com que os melhores indivíduos sejam sempre utilizados (CORNE; LONES, 2018).

Com o intuito de simular esse comportamento, neste trabalho antes da seleção dos parentes, o algoritmo faz uma avaliação de aptidão em todas as soluções da população, retornando os três melhores resultados para serem utilizados como progenitores.

Após a seleção, é realizado o procedimento de cruzamento, reunindo os planejamentos de carga dos três cromossomos e selecionando dois pontos de cada planejamento, em que será realizada a divisão dos planejamentos, em contraste ao cruzamento de um ponto implementando no algoritmo genético. Também é considerada a taxa de mutação de cada indivíduo, com cada descendente sendo capaz de herdar a taxa de mutação de um dos progenitores, sendo que quanto maior a aptidão do progenitor, maior a chance dessa característica ser passada para um descendente.

Após o cruzamento, é feita a avaliação dos progenitores junto dos descendentes, retornando-se os três melhores indivíduos para a população.

3.3.2 Mutação

Nesta implementação é mantida a mutação do planejamento de cada indivíduo com base na inversão de posições entre dois contêineres. No entanto, visto que neste modelo a taxa de mutação é um parâmetro específico de cada cromossomo, ela também é influenciada pela operação de mutação. A mutação ocorre com a geração de um número aleatório em uma faixa entre +0.1 e –0.1, referente ao valor original.

No algoritmo baseado em estratégia evolutiva, diferente do algoritmo genético, a taxa de mutação é particular de cada cromossomo, não sendo possível definir a chance de um indivíduo ser modificado pela operação de mutação.

3.4 Testes

Para os testes de ambos os algoritmos, genético e baseado em estratégias evolutivas, foram geradas seis instâncias de teste de maneira aleatória, como no trabalho de Azevedo *et al*. (2011), cada instância é formada por uma matriz de transporte, que é gerada de maneira em que a capacidade total do navio não seja excedida em nenhum porto, considerando os carregamentos e descarregamentos em cada terminal portuário.

Foram gerados três tipos diferentes de matrizes de transporte, como descritas no trabalho de Avriel *et al*. (1998), sendo estes tipos: curta distância, em que os contêineres percorrem poucos portos antes de serem descarregados; Longa distância, em que os contêineres percorrem diversos portos antes de serem descarregados; E distância mista, que é uma junção dos dois tipos.

A implementação deste trabalho mostrou exigir um grande esforço computacional, devido a aleatoriedade envolvida no carregamento do navio, o que fez com que a dimensão das matrizes utilizada para os testes seja composta por problemas menores, devido a inviabilidade de testes em dimensões mais amplas.

Por isso, foram considerados navio porta-contêineres com capacidade máxima de trezentos contêineres, além disso, foram considerados matrizes com dez e quinze portos. Cada instância de teste foi executada trinta vezes por cada tipo de algoritmo, em um computador com AMD Ryzen™ 5 7600X 4.7GHz, memória RAM de 16GB e sistema operacional Windows 10.

4 RESULTADOS E DISCUSSÃO

Após os testes realizados, todos os dados obtidos de quantidade de trocas desnecessárias de contêineres e tempo de execução durante cada uma das execuções foi exportado para a ferramenta *IBM SPSS Statistics 21*, na qual foram realizadas as análises estatísticas para a comparação da performance dos dois algoritmos, sendo que para a análise geral entre os dois algoritmos, como vista no quadro 2, foi utilizado o teste não paramétrico U de Mann Whitney, e para o teste de diferença entre matrizes, que pode ser vista no quadro 3, foi utilizado o teste não paramétrico de Kruskal Wallis. Todos os testes estatísticos foram realizados considerando um nível de significância de 5% e Intervalo de confiança de 95%.

No quadro 2, é possível observar uma superioridade do algoritmo genético, representado como GA no quadro, sobre o algoritmo baseado em estratégias evolutivas, representado como ES, em ambos os casos apresentados, de 10 portos e 15 portos, sendo que o primeiro algoritmo se mostra melhor do que o segundo tanto em número de trocas, quanto em tempo de execução.

Nas instâncias com 10 portos, foi possível observar uma diferença de 250,4 entre as médias de trocas desnecessárias, além da diferença de 1358,6 segundos na média de tempo de execução dos dois algoritmos. Essa diferença de tempo pode ser explicada pela forma com que ambos os algoritmos funcionam, visto que o algoritmo baseado em estratégias evolutivas é obrigado a avaliar a aptidão de toda a população a cada vez que vai escolher os progenitores da geração posterior.

Essa diferença no tempo de execução fica ainda maior nos casos com 15 portos, aumentando para 2211,1 segundos, devido à maior complexibilidade das matrizes de transporte, que resulta em uma demora maior para a análise de aptidão. Porém, também é possível observar que ao contrário da diferença no tempo de execução, a média da diferença nas trocas desnecessárias diminui se comparado as instâncias de 10 portos, com a diferença de 126 trocas desnecessárias, ainda com uma superioridade do algoritmo genético.

Quadro 2 – Comparação de tempo em segundos e trocas entre os algoritmos genético e baseado em estratégias evolutivas em matrizes de 10 e 15 portos.

Trocas desnecessárias considerando 10 portos

Algoritmos	Média	Diferença	Desvio padrão	Mínimo	Máximo
GA	712,5	250,4	78,8	585	816
ES	962,9		92,1	836	1.104

Tempo (s) considerando 10 portos

Algoritmos	Média	Diferença	Desvio padrão	Mínimo	Máximo
GA	746,7	1.358,6	113,7	518,2	879,2
ES	2.105,3		433,5	1.401,3	2.709,5

Trocas desnecessárias considerando 15 portos

Algoritmos	Média	Diferença	Desvio padrão	Mínimo	Máximo
GA	1.236,5	126,0	159,3	1.013	1.439
ES	1.362,5		120,8	1.182	1.507

Tempo (s) considerando 15 portos

Algoritmos	Média	Diferença	Desvio padrão	Mínimo	Máximo
GA	1.656,7	2.211,1	41,9	1.596,5	1.793,7
ES	3.867,8		537,7	3.088,0	4.884,0

Fonte: Do autor (2023).

No quadro 3 é possível observar a performance de ambos os algoritmos de acordo com os tipos diferentes de matriz: longa (l), mista (m) e curta (c), em que é verificável a superioridade do algoritmo genético em todas as variações.

Quadro 3 – Comparação de tempo em segundos e trocas entre os algoritmos genético e baseado em estratégias evolutivas em diferentes matrizes.

GA

Trocas desnecessárias considerando 10 portos

Matriz	Média	Desvio padrão	Mínimo	Máximo
c	606,3	10,4	585	637
l	741,7	10,7	716	762
m	789,5	11,8	761	816

Tempo (s) considerando 10 portos

Matriz	Média	Desvio padrão	Mínimo	Máximo
c	592,3	15,5	518,2	607,4
l	807,1	6,4	790,3	817,0
m	840,7	42,7	684,0	879,2

Trocas desnecessárias considerando 15 portos

Matriz	Média	Desvio padrão	Mínimo	Máximo
l	1.031,3	10,6	1.013	1.049
m	1.263,4	14,4	1.243	1.288
c	1.414,9	16,5	1.389	1.439

Tempo (s) considerando 15 portos

Matriz	Média	Desvio padrão	Mínimo	Máximo
l	1.618,6	14,6	1.596,5	1.663,7
c	1.650,7	19,3	1.603,2	1.676,2
m	1.700,7	35,7	1.662,8	1.793,7

ES

Trocas desnecessárias considerando 10 portos

Matriz	Média	Desvio padrão	Mínimo	Máximo
c	865,3	11,6	836,0	883,0
l	939,5	14,4	901,0	975,0
m	1.084,0	10,6	1.060,0	1.104,0

Tempo (s) considerando 10 portos

Matriz	Média	Desvio padrão	Mínimo	Máximo
c	1.568,2	83,0	1.401,3	1.648,2
l	2.190,6	146,8	1.837,5	2.354,4
m	2.557,1	176,4	2.098,7	2.709,5

Trocas desnecessárias considerando 15 portos

Matriz	Média	Desvio padrão	Mínimo	Máximo
l	1.203,5	12,6	1.182	1.226
m	1.392,9	17,6	1.358	1.418
c	1.491,1	9,7	1.475	1.507

Tempo (s) considerando 15 portos

Matriz	Média	Desvio padrão	Mínimo	Máximo
c	3.266,1	184,1	3.088,0	3.500,2
l	3.866,8	201,6	3.516,0	4.102,0
m	4.470,4	250,6	4.139,2	4.884,0

Fonte: Do autor (2023).

Quando comparando os resultados de tempo de execução da implementação dos algoritmos deste trabalho, em que as operações de seleção,

cruzamento e mutação são realizadas no próprio planejamento de cargas, com o tempo de execução da comparação por regras, é clara a vantagem da implementação por regras.

Essa diferença pode ser demonstrada na capacidade da implementação por regras conseguir solucionar instâncias mais complexas de matrizes de transporte em tempo reduzido, como no trabalho de Azevedo *et al*. (2011), em que o algoritmo de *simulated annealing* com regras é capaz de solucionar instâncias utilizando navios com capacidade máxima de trezentos contêineres passando por 30 portos com um tempo médio de 162,16 segundos.

Parte dessa diferença no tempo de execução se dá pela necessidade dessa implementação ter que verificar todas as posições das cargas no navio e gerar novas posições aleatoriamente diversas vezes para manter a consistência do posicionamento dos contêineres, exigindo um grande esforço computacional, o que não é necessário na implementação por regras, visto que o processo de evolução acontece nas regras que definem como os contêineres serão carregados no navio.

5 CONCLUSÃO

Essa pesquisa teve o objetivo de implementar o algoritmo genético e um algoritmo baseado em estratégias evolutivas ao problema de carregamento e descarregamento de contêineres em terminais portuários.

Após o desenvolvimento de ambos os algoritmos, em uma implementação que aplica as operações de evolução no planejamento de cargas dos navios, e o teste de ambos os algoritmos em diferentes instâncias, com diferentes tipos de matrizes e quantidade de portos, foi constatada uma superioridade do algoritmo genético, tanto na quantidade inferior de trocas desnecessárias, quanto no tempo de execução.

Também foi possível observar que a implementação proposta exige um grande esforço computacional, tornando inviável a sua utilização em matrizes maiores do que quinze portos.

Com base no conhecimento adquirido na confecção dessa pesquisa, para trabalhos futuros sugere-se: comparação entre o algoritmo genético e o baseado em estratégias evolutivas com a implementação por regras; implementação de novos algoritmos evolucionários, como a evolução diferencial; implementação de outras soluções heurísticas.

REFERÊNCIAS

AVRIEL, Mordecai; PENN, Michal; SHPIRER, Naomi. Container ship stowage problem: complexity and connection to the coloring of circle graphs. en. Discrete Applied Mathematics, v. 103, n. 1–3, p. 271–279, jul. 2000. ISSN 0166218X. DOI: https://doi.org/10.1016/S0166-218X(99)00245-0.

AVRIEL, Mordecai; PENN, Michal; SHPIRER, Naomi; WITTEBOON, Smadar. Stowage planning for container ships to reduce the number of shifts. Annals of Operations Research, v. 76, n. 0, p. 55–71, jan. 1998. ISSN 1572-9338. DOI: https://doi.org/10.1023/A:1018956823693.

AZEVEDO, Anibal T; SENA, Galeno J; CHAVES, Antônio A; RIBEIRO, Cassilda M. O PROBLEMA DE CARREGAMENTO E DESCARREGAMENTO DE CONTÊINERES EM TERMINAIS PORTUÁRIOS: UMA ABORDAGEM VIA SIMULATED ANNEALING. Disponível em :<http://ws2.din.uem.br/~ademir/sbpo/sbpo2011/pdf/87127.pdf>. Acesso em: 19 abr. 2023.

BARTZ-BEIELSTEIN, Thomas *et al*. Evolutionary Algorithms. en. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, v. 4, n. 3, p. 178–195, mai. 2014. ISSN 19424787. DOI: https://doi.org/10.1002/widm.1124

BENEDITO, Marcelo; SANTOS, André. Evolutionary Algorithm and Ant Colony Optimization for a Container Stowage Problem. en, p. 12, 2015

BILICAN, Mevlut Savas; EVREN, Ramazan; KARATAS, Mumtaz. A Mathematical Model and Two-Stage Heuristic for the Container Stowage Planning Problem With Stability Parameters. IEEE Access, v. 8, p. 113392–113413, 2020. ISSN 2169-3536. DOI: 10.1109/ACCESS.2020.3003557.

BNDES. A Cabotagem no Brasil. [S.l.: s.n.]. Disponível em: <https://www.bndes.gov.br/wps/portal/site/home/conhecimento/noticias/noticia/cabotagem>. Acesso em: 24 maio 2023.

CARRARO, Luziana F; CHIWIACOWSKY, Leonardo D; GÓMEZ, Arthur T. UMA APLICAÇÃO DAS METAHEURÍSTICAS ALGORITMO GENÉTICO E COLÔNIA ARTIFICIAL DE ABELHAS ATRAVÉS DA CODIFICAÇÃO POR REGRAS PARA RESOLVER O PROBLEMA DE CARREGAMENTO DE NAVIOS- CONTÊINERES. pt, p. 12, 2013. Disponível em: <http://www.din.uem.br/sbpo/sbpo2013/pdf/arq0230.pdf>. Acesso em: 18 maio 2022.

CORNE, David; LONES, Michael A. Evolutionary Algorithms. In: HANDBOOK of Heuristics. [S.l.]: Springer International Publishing, 2018. P. 1–22. DOI: 10.1007/978-3-319-07153-4_27-1.

DOKEROGLU, Tansel *et al*. A survey on new generation metaheuristic algorithms. en. Computers Industrial Engineering, v. 137, p. 106040, nov. 2019. ISSN 03608352. DOI: 10.1016/j.cie.2019.106040.

EUCHI, Jalel *et al*. Ant Colony Optimization for Solving the Container Stacking Problem: set. 2016.

LIMA, Fábio Mascagna Bittencourt. Resolução do problema de carregamento e descarregamento de contêineres em terminais portuários via Beam Search. 2011. 1 CD-ROM. Trabalho de conclusão de curso - (bacharelado - Engenharia Elétrica) – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2011. Disponível em: <http://hdl.handle.net/11449/119641>.

NUNES, Ginete; NASCIMENTO, Maria Cristina; LUZ, Maria Aparecida. Pesquisa científica: conceitos básicos. **Revista Multidisciplinar e de Psicologia**, v.10, n.29, p. 144-151, fev. 2016. ISSN 1981-1179.

ONU. Objetivos de Desenvolvimento Sustentável | As Nações Unidas no Brasil. pt-br. [S.l.: s.n.]. Disponível em: <https://brasil.un.org/pt-br/sdgs>. Acesso em: 27 nov. 2022.

SAIKIA, S *et al*. Evolutionary RL for Container Loading. en. Computational Intelligence, p. 6, 2018.

SAXON, Steve; STONE, Max. Container shipping: The next 50 years. [S.l.: s.n.], 2017. Disponível em: <https://www.mckinsey.com/~/media/mckinsey/industries/travel%20logistics%20and%20infrastructure/our%20insights/how%20container%20shipping%20could%20reinvent%20itself%20for%20the%20digital%20age/container-shipping-the-next-50-years-103017.pdf>. Acesso em: 22 maio 2023.

TIJJANI, Sani; OZKAYA, Armagan. A Comparison of Reinforcement Learning and Evolutionary Algorithms for Container Loading Problem. In.

TRIVIÑOS, Augusto Nibaldo Silva. **Introdução à pesquisa em ciências sociais**: a pesquisa qualitativa em educação. São Paulo: Atlas, 2011.

UNCTAD. Review of Maritime Transport 2021. en. REVIEW OF MARITIME TRANSPORT, p. 177, 2021. Acesso em: 15 nov. 2022.