UNIVERSIDADE FEDERAL DO MARANHÃO

SAMIR SILVA FAHD

MINIMIZAÇÃO DE MOVIMENTOS IMPRODUTIVOS NO PÁTIO DE CONTÊINERES DO PORTO DO ITAQUI UTILIZANDO ALGORITMO GENÉTICO

São Luís – MA

2022

SAMIR SILVA FAHD

MINIMIZAÇÃO DE MOVIMENTOS IMPRODUTIVOS NO PÁTIO DE CONTÊINERES DO PORTO DO ITAQUI UTILIZANDO ALGORITMO GENÉTICO

Trabalho de conclusão de curso apresentado ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Francisco Glaubos Nunes Climaco

São Luís – MA

2022

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Fahd, Samir Silva.
Minimização de Movimentos Improdutivos no Pátio de Contêineres do Porto do Itaqui Utilizando Algoritmo Genético / Samir Silva Fahd. - 2022.
62 f.

Orientador(a): Francisco Glaubos Nunes Climaco.
Curso de Ciência da Computação, Universidade Federal do Maranhão, Videoconferência, 2022.

1. Algoritmos genéticos. 2. Contêiner. 3. Otimização. 4. Porto. I. Climaco, Francisco Glaubos Nunes. II. Título.

SAMIR SILVA FAHD

MINIMIZAÇÃO DE MOVIMENTOS IMPRODUTIVOS NO PÁTIO DE CONTÊINERES DO PORTO DO ITAQUI UTILIZANDO ALGORITMO GENÉTICO

Trabalho de conclusão de curso apresentado ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 04 de fevereiro de 2022

BANCA EXAMINADORA

Profº Dr. Francisco Glaubos Nunes Climaco
Universidade Federal do Maranhão

Profº Me. Italo Francyles Santos Da Silva
Universidade Federal do Maranhão

Profº Dr. Anselmo Cardoso De Paiva
Universidade Federal do Maranhão

São Luís
2022

AGRADECIMENTOS

A Deus,
à minha esposa, Gabriela, aos meus pais e a QQ pelo apoio irrestrito, incondicional e sempre presente,
ao professor Glaubos pelas orientações, sugestões e correções no desenvolvimento do trabalho e
aos professores Anselmo, Simara e Geraldo pela paciência e incentivo constantes.

RESUMO

Com o provável aumento dos fluxos de entrada e saída de contêineres no Porto do Itaqui, que são decorrentes dos investimentos que vêm sendo realizados por sua administração, surgirá a necessidade de utilizar ferramentas computacionais de suporte à tomada de decisões que otimizem as operações do pátio de contêineres. Este trabalho aborda o problema específico da minimização de movimentos improdutivos, ou seja, visa evitar, ao máximo, que contêineres sejam movimentados pela simples necessidade de reorganização do pátio. Como solução para este problema, será elaborado um modelo que represente o pátio de contêineres e suas operações, bem como uma ferramenta de suporte à tomada de decisões aplicada a este modelo, baseada em algoritmos genéticos.

Palavras chave: Porto. Contêiner. Algoritmos genéticos. Otimização.

ABSTRACT

Considering the likely increase of inflows and outflows of containers at Itaqui Port, which are a result of the investments made by its administration, it is necessary to use a Decision Making Support System that optimizes the operations of container yards. This work tackles the specific problem of the minimization of unproductive movements, in other words, it aims to avoid as much as possible that containers are moved by the simple necessity of reorganizing the yard. As a solution for this problem, a model that represents the yard and its operation will be created, as well as a Decision Making Support System applied to this model and based on a genetic algorithm.

Key-words: Port. Container. Genetic algorithms. Optimization.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE PSEUDOCÓDIGOS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Em 2018, o Porto do Itaqui movimentou 22,3 milhões de toneladas de carga (ANUÁRIO ANTAQ, 2018), ultrapassando seu próprio recorde histórico de movimentação. No terceiro trimestre do mesmo ano, apresentou crescimento de 21,6% em carga movimentada, sendo essa a maior taxa de crescimento registrada dentre os portos públicos nesse período. O Terminal Marítimo da Ponta da Madeira, por sua vez, movimentou 198,1 milhões de toneladas em 2018 e, no terceiro trimestre, apresentou crescimento de 8,5%, o que representa a terceira maior taxa de crescimento registrada dentre os terminais de uso privado (BOLETIM 3º SEMESTRE ANTAQ, 2018). Com esses números, o complexo portuário da Baía de São Marcos em São Luís - MA, que é composto pelos dois portos já citados e pelo porto da Alumar, reafirma sua posição como maior do Brasil em movimentação de carga.

Essa liderança se deve, em parte, à sua posição estratégica, próxima geograficamente aos mercados norte-americano e europeu e à largura e profundidade do seu canal de acesso, que são as maiores dentre as áreas portuárias brasileiras (AMARAL; ALFREDINI, 2010), podendo comportar os maiores navios existentes no mundo. Estas características são bastante exploradas nas operações de transbordo de granéis líquidos, nas quais navios tanque de grande porte (com valores de calados acima da média) atracam no Porto do Itaqui e transferem suas cargas para navios menores que, por sua vez, se destinam a terminais regionais.

Tais características também permitiriam que o Porto do Itaqui se estabelecesse como terminal concentrador de contêineres (*hub ports*), que, segundo Paiva (2006), têm por principal função o recebimento de navios de grande porte, dos quais os contêineres são transbordados para navios menores, que seguem viagem em direção a terminais de contêineres regionais. Esse potencial, entretanto, ainda é inexplorado, visto que, atualmente, a linha regular de contêineres para a cidade de São Luís atende apenas a mercados regionais.

Visando colocar o Itaqui no mapa das linhas de transportes marítimas de contêineres, a Empresa Maranhense de Administração Portuária – EMAP – investiu, em 2018, R$ 19 milhões em obras de engenharia e de reestruturação elétrica que viabilizaram a reativação da linha regular de contêineres para a cidade de São Luís. A infraestrutura portuária para armazenagem de contêineres contempla 20.250 m² com capacidade estática de 1.341 TEUs (sigla para *twenty-foot equivalent unit*) e

potencial para movimentar 1.800 contêineres / mês (EMAP, 2019). Porém, a exemplo de outros portos, é sabido que, com o aumento do fluxo de contêineres e a consequente maior ocupação do pátio, surgirá a necessidade de utilizar ferramentas computacionais de suporte à tomada de decisões, inclusive as aplicadas à estratégia de empilhamento de contêineres no pátio, categoria na qual o objeto de estudo deste trabalho se inclui.

Segundo Borgman, Asperen e Dekker (2006), os principais objetivos de uma estratégia de empilhamento são:

- Maximização do aproveitamento do pátio: Uso eficiente do espaço disponível no pátio;
- Otimização dos processos de transporte: Eficiência no tempo de transporte do contêiner do cais para a pilha e destino seguinte;
- Minimização dos movimentos improdutivos: Minimizar o rearranjo de contêineres, reduzindo desta forma o custo operacional.

No que se refere ao objetivo de minimizar movimentos improdutivos, entende-se que um único movimento é necessário para armazenar um contêiner no pátio e um único movimento é necessário para retirar um contêiner do pátio, qualquer outra movimentação ocasionada por necessidade de reorganização do pátio será considerado um **movimento improdutivo**. O deslocamento de um contêiner X1 de uma pilha Y1 para outra pilha Y2 ocasionado pela necessidade de retirar outro contêiner X2 que se localizava abaixo do contêiner X1 na pilha Y1 é um exemplo de movimento improdutivo.

Este trabalho tem por objetivo abordar o problema da minimização de movimentos improdutivos. Será apresentada uma solução utilizando algoritmo genético e demonstrado o seu desempenho através de testes em simulações, bem como o comparativo com outras soluções para o mesmo problema disponíveis na literatura.

1.1 DEFINIÇÃO DO TEMA

Na literatura, é provado que o problema da minimização de movimentos improdutivos no empilhamento de contêineres é NP-completo (AVRIEL; PENN; SHPIRER, 2000), o que justifica a utilização de heurística para solucioná-lo. Algumas abordagens simples são a heurística da pilha mais baixa (LEE; WANG; MIAO, 2008),

na qual um contêiner, no momento de sua chegada ou após rearranjo, é colocado sobre a menor pilha disponível, e a heurística do índice de rearranjo (MURTY at al., 2005), na qual é atribuído a cada pilha do pátio um índice diretamente proporcional à quantidade de contêineres nessa pilha que sairão antes do contêiner a ser rearranjado. O contêiner é então posto sobre a pilha de menor índice.

Para solucionar o referido problema, este trabalho optou pela utilização de algoritmos genéticos, que são técnicas de busca baseadas no processo biológico de evolução natural. Esta escolha se deve aos bons resultados apresentados por Piva (2008), que utilizou esta heurística na resolução de problema semelhante ao abordado neste trabalho e também às seguintes vantagens dos algoritmos genéticos enumeradas por Haupt (2004):

- Podem ser usados na otimização com variáveis contínuas ou discretas;
- Lidam com grande número de variáveis;
- Podem fugir dos mínimos locais;
- Proveem uma lista de soluções ótimas, não apenas uma solução única;
- Podem trabalhar com dados gerados numericamente, dados experimentais e funções analíticas.

1.2 JUSTIFICATIVA

O tempo necessário e os custos inerentes às operações de um porto são fatores determinantes para a sua competitividade. Portos com operações eficientes, nos quais os navios precisem ficar menos tempo atracados e cujos custos para os seus utilizadores são mais baixos, costumam ser mais atrativos.

Em se tratando de operações de contêineres, melhorias podem ser obtidas através do aumento das dimensões do pátio, da aquisição de equipamentos mais eficientes ou de investimentos em infraestrutura de modo geral. A exemplo dos realizados pela EMAP em 2018, citados anteriormente neste trabalho, tais investimentos costumam ser bastante onerosos.

A aplicação de técnicas de otimização, como a proposta neste trabalho, apresenta resultados semelhantes (até um certo limite) aos obtidos através de investimentos em infraestrutura, porém, demandam investimento financeiro muito menor. Portanto, a utilização de ferramentas computacionais que otimizem processos operacionais deve ser fator comum a todos os portos que visem se tornar competitivos no mercado.

1.2 OBJETIVO

1.3.1 Geral

Aplicar algoritmo genético ao problema da minimização de movimentos improdutivos no pátio de contêineres do Porto do Itaqui e, desta forma, diminuir os custos e o tempo das operações tornando o Itaqui mais competitivo no mercado.

1.3.1 Específicos

- Modelar o pátio de contêineres e sua operação abstraindo as informações irrelevantes e gerar um modelo no qual possa ser aplicada a heurística proposta;
- Desenvolver o algoritmo genético de forma que se adeque ao problema tratado, definindo critério de seleção dos pais, metodologia de *crossover*, metodologia de mutação, função de avaliação e condição de parada;
- Avaliar o desempenho do algoritmo desenvolvido através de simulações;
- Comparar os resultados obtidos com os de outras técnicas aplicadas ao mesmo problema.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 OPERAÇÕES DE CONTÊINERES

Nesta seção serão abordados alguns aspectos gerais inerentes às operações de descarga de navios de contêineres e armazenagem de contêineres em portos. As características específicas do Porto do Itaqui serão destacadas.

2.1.1 Estivagem de contêineres e operações em navios

Segundo Neto et al. (2021), os navios celulares, que são destinados ao transporte de contêineres, são divididos em células chamadas de *slots*, que são utilizados para estivagem de contêineres padronizados de 20 pés (ocupa 1 *slot*) ou 40 pés (ocupa 2 *slots*). A capacidade de um navio deste tipo é dada em TEUs (sigla para *twenty-foot equivalent unit*), que corresponde à quantidade de contêineres de 20 pés que o navio comporta. Estes navios podem ser classificados como *Self Sustained*, que possuem recursos próprios de operação ou *Non Self Sustained,* que não possuem equipamentos de bordo e, portanto, são dependentes dos equipamentos de costado.

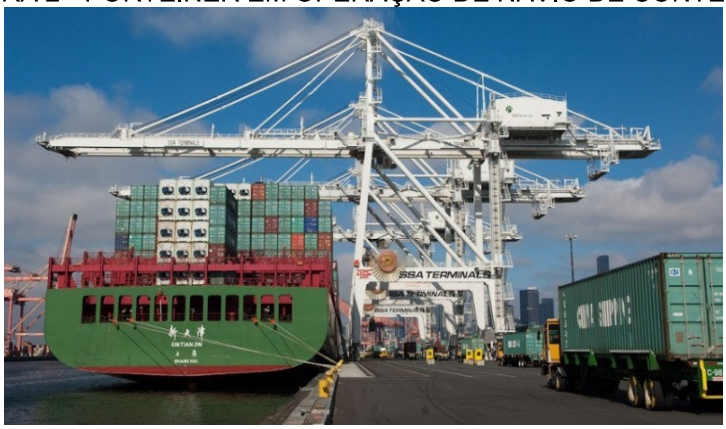
Os navios que operam na linha regular de contêiner do Porto do Itaqui pertencem à primeira categoria, porém costumam ser operados tanto com equipamentos de bordo, quanto com equipamento de costado do tipo guindaste móvel sobre pneus (FIGURA 1). Alguns Portos possuem equipamentos de cais especializados denominados portêineres (FIGURA 2), que tornam as operações de navios de contêineres, significativamente, mais rápidas. Este não é o caso do Porto do Itaqui.

A configuração dos contêineres dentro do navio também é um aspecto relevante para sua operação. O desafio de planejar esta configuração é similar e está interligado ao problema abordado neste trabalho. O *ship planner*, profissional responsável por esta tarefa, precisa analisar a sequência em que o navio atracará nos Portos que possui escala, considerando os contêineres que serão embarcados e desembarcados em cada um. Existem diversos softwares que auxiliam nesta tarefa, alguns exemplos são o *Cube-IQ* da empresa *MagicLogic* (MAGICLOGIC, 2021) e o *SimpleStow* da empresa *AMT Marine Software* (AMT MARINE SOFTWARE, 2021).

FIGURA 1 - GUINDASTE MÓVEL SOBRE PNEUS EM OPERAÇÃO DE NAVIO DE CONTÊINERES NO PORTO DO ITAQUI

FONTE: EMAP (2019).

FIGURA 2 - PORTÊINER EM OPERAÇÃO DE NAVIO DE CONTÊINER



FONTE: Engeplus (2021).

No Itaqui, o navio Aliança Leblon tem escala regular todas as segundas-feiras. Durante sua estadia, são desembarcados os contêineres ovados (cheios), contendo mercadorias destinadas a mercados regionais. Também são embarcados os contêineres vazios que estavam ovados quando foram desembarcados no Itaqui em viagens anteriores do Aliança Leblon e já sofreram o processo de desova, trata-se da devolução de contêineres vazios. Também ocorre, com menor frequência, o embarque de contêineres ovados, contendo mercadorias de exportação.

2.1.2 Entrada, saída e armazenamento de contêineres no pátio

Seguindo o fluxo de um contêiner no Porto do Itaqui, após o desembarque, um equipamento do tipo *reach stacker* é utilizado para pôr o contêiner sobre a carroceria de um veículo do tipo caminhão prancha (FIGURA 3). O contêiner é então

transportado pelo veículo da faixa de cais até o pátio de contêiner, onde outra *reach stacker* é utilizada para retirar o contêiner da carroceria do veículo e empilhá-lo sobre a pilha de contêineres (FIGURA 4).

No processo de retirada de um contêiner do pátio, também é utilizada uma *reach stacker* para retirar o contêiner da pilha e pô-lo sobre a carroceria do veículo que o conduzirá até o destino final. Entretanto, se o contêiner a ser retirado não estiver no topo de uma pilha, todos os contêineres que estiverem sobre ele ou à frente dele na pilha precisam ser realocados para outra pilha do pátio que não esteja cheia. Trata-se do movimento improdutivo, objeto de estudo deste trabalho.

Alguns Portos possuem equipamentos especializados nos pátios de contêineres. Não é o caso do Porto do Itaqui. Steenken, Voss e Stahlbock (2004) apresentam 3 categorias: *rail mounted gantry cranes* (RMG), que operam sobre trilhos (FIGURA 5), *rubber tired gantries* (RTG), que operam sobre pneus e *over-head bridge cranes* (OBC), que são montados em pilares de concreto ou aço. A utilização de equipamentos especializados destes tipos torna as operações mais rápidas e facilita, consideravelmente, a retirada de contêineres do pátio, uma vez que possibilita a retirada de contêineres de uma coluna sem que seja necessária a retirada de todos os contêineres das colunas a frente na mesma pilha.

FIGURA 3 - EQUIPAMENTO REACH STACKER PONDO UM CONTÊINER SOBRE A CARROCERIA DE UM CAMINHÃO PRANCHA

FONTE: ASM Treinamentos (2021).

FIGURA 4 - EQUIPAMENTO REACH STACKER PONDO UM CONTÊINER SOBRE UMA PILHA

FONTE: Nautic Expo (2021).

FIGURA 5 - RAIL MOUNTED GANTRY CRANES (RMG)

FONTE: Sea News (2021).

As diversas características específicas do pátio de contêineres do Itaqui não serão abordadas neste trabalho, uma vez que a solução desenvolvida será aplicada sobre um modelo simplificado desse pátio. Tal modelo segue as seguintes regras de simplificação:

- Restrições que definem armazenamento de contêineres específicos em *slots* específicos serão desconsideradas. Esse é o caso dos contêineres refrigerados, que precisam ser armazenados em locais próximos a tomadas, e dos contêineres com cargas perigosas, que precisam ser armazenados em locais adequados ao tipo de carga;
- O pátio será tratado com um bloco único formado por pilhas com o mesmo limite de tamanho que, por sua vez, são formadas por colunas com o mesmo limite de altura.
- Será considerado que todos os contêineres são do tipo padrão (*dry box*);

- Será considerado que um contêiner só poderá ser alocado em uma coluna se todas as outras colunas atrás dela na pilha já estiverem cheias (com contêineres até a altura limite). Isso já é praticado no Itaqui. Caso contrário, haveria acréscimo de movimentos improdutivos ou inutilização de posições no pátio, visto que os equipamentos disponíveis no Itaqui somente são capazes de retirar contêineres de uma coluna após a retirada de todos os contêineres das colunas à frente na mesma pilha;
- Será considerado que todos os contêineres são do mesmo tamanho, inclusive, não levando em conta o empilhamento de contêineres de 20' sobre contêineres de 40' (ocupa duas pilhas) e vice-versa.

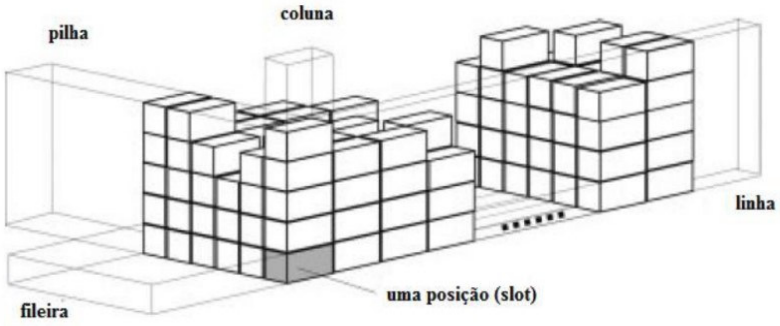
Contêineres com restrições de armazenamento ou de tipos diferentes do padrão representam uma pequena parcela da totalidade de contêineres do pátio e são armazenados em locais segregados e destinados a eles. Desta forma, o armazenamento da maioria dos contêineres é compatível com o modelo de pátio simplificado apresentado. O empilhamento de 1 contêiner de 40' sobre dois contêineres de 20', que não é compatível com o modelo, ocorre raramente no Porto do Itaqui.

Quanto à nomenclatura, será utilizada a apresentada por Carraro (2012), que define os seguintes termos, todos ilustrados na FIGURA 6:

- ***Slot*** (ou posição) - menor unidade de armazenamento. Local exata de um contêiner;
- **Coluna** - vários *slots* colocados uns sobre os outros;
- **Pilhas** - várias colunas colocadas uma à frente da outra;
- **Bloco** - grupo de pilhas colocadas lado-a-lado;
- **Fileira** - *slots* do mesmo nível (altura) em uma pilha;
- **Linha** - sequência de colunas lado a lado.

Desta forma, uma coordenada **(fileira, pilha, linha)** define a posição exata de um *slot*.

FIGURA 6 - MODELO DO PÁTIO DE CONTÊINERES E NOMENCLATURA



FONTE: Carraro (2012).

2.2 ALGORITMOS GENÉTICOS

Algoritmos evolucionários são técnicas heurísticas que têm como característica definidora a busca por soluções para problemas computacionais por meio de simulação da teoria da evolução das espécies (LINDEN, 2012). Esses algoritmos recebem como entrada uma população, cujos indivíduos são soluções possíveis para o problema. Através dos processos de seleção, reprodução (*crossover*) e mutação, uma solução evoluída é encontrada.

Algoritmos genéticos (AGs) são uma subcategoria dos algoritmos evolucionários, que combinam a simulação da evolução das espécies com uma forma estruturada de troca de informações genéticas através dos operadores de mutação, seleção e *crossover*. Se caracterizam, principalmente, pela busca de soluções globais evitando cair em ótimos locais e pela capacidade de encontrar soluções satisfatórias sem que seja necessário avaliar todas as soluções possíveis. Estas características tornam os AGs adequados para a resolução do problema apresentado neste trabalho, porém, assim como todas as outras técnicas heurísticas, não podem garantir a entrega de um resultado ótimo nem a repetição da mesma resposta em execuções diferentes.

A seguinte sequência de passos, apresentada por Linden (2012), descreve, em uma visão de alto nível, o funcionamento de um AG:

1. Inicialize a população de cromossomos;
2. Avalie cada cromossomo na população;
3. Selecione os pais para gerar novos cromossomos. Aplique os operadores de *crossover* e mutação a esses pais de forma a gerar os indivíduos da nova geração;

4. Apague os velhos membros da população;
5. Avalie todos os novos cromossomos e insira-os na população;
6. Se o tempo acabou, ou o melhor cromossomo satisfaz os requerimentos de performance, retorne-o, caso contrário volte para o passo 3.

Nas próximas seções, serão explicados os conceitos necessários para a compreensão da sequência de passos descrita acima.

2.2.1 Cromossomos, inicialização da população e regra de tradução

Cada solução possível para o problema deve ser representada de forma estruturada para possibilitar a aplicação dos operadores de mutação, seleção e *crossover*, que serão explicados nas próximas seções. Em alusão à genética, dá-se o nome de cromossomo a essa forma estruturada. O cromossomo, por sua vez, é formado por partes chamadas genes, que podem ter um valor dentre diversos possíveis, chamados de alelos (LINDEN, 2012). Também é necessário definir regras de tradução do cromossomo para a solução real e vice-versa.

Existem diversas representações possíveis, algumas das mais comuns são a binária, em que cada cromossomo é formado por uma *string* de zeros e uns, a permutação, que é comum na resolução de problemas de ordenação, tal qual o problema do caixeiro viajante (PCV), e por valor, na qual o cromossomo é a própria solução do problema, dispensando regra de tradução (GAD, 2020).

O PCV será utilizado como exemplo para demonstrar a utilização de algoritmos genéticos para resolver problemas de permutação, visto que o problema abordado neste trabalho é deste tipo. Outros tipos de AGs baseados em outras representações de cromossomos não serão abordados. O PCV pode ser definido como o problema de encontrar o roteiro de menor distância ou custo que passa por um conjunto de cidades, sendo cada cidade visitada exatamente uma vez (REYNOSO, 2017).

Neste problema, a representação do cromossomo é uma lista ordenada, em que cada item da lista está associado a uma cidade, conforme definido pela regra de tradução. Por exemplo, se um caixeiro viajante precisa visitar as cidades de São Luís, Bom Jardim, Carolina e Balsas, a regra de tradução pode associar as cidades aos números ‘1’, ‘2’, ‘3’ e ‘4’, respectivamente. Desta forma, se o cromossomo com melhor função de avaliação for o 1-3-4-2, significa que a melhor rota encontrada para o

caixeiro é começar a viagem pela cidade de São Luís, depois ir para Carolina, depois Balsas e, por último, Bom jardim.

Para inicializar a população, serão gerados N cromossomos de forma aleatória, sendo N o tamanho definido para a população. Uma possível população inicial de tamanho ‘5’ para o exemplo citado seria a formada pelos cromossomos: ‘1-3-4-2’, ‘1-2-3-4’, ‘4-3-2-1’, ‘1-4-3-2’ e ‘1-2-4-3’. É importante ressaltar que este exemplo é ilustrativo e, em situações reais de aplicação de algoritmos genéticos, normalmente, existe uma quantidade significativamente maior de combinações possíveis e a geração de cromossomos repetidos não é uma preocupação, de modo que não é necessário criar uma restrição neste sentido.

2.2.2 Avaliação dos Cromossomos

A avaliação do cromossomo é feita pela função de avaliação. É essa função que define a qualidade (ou nível de adaptação) do cromossomo (LINDEN, 2012), ou seja, em um problema de minimização, se o valor encontrado ao aplicar a função de avaliação no cromossomo A for menor que o valor encontrado para o cromossomo B, significa que a solução indicada por A é melhor que a indicada por B. A função de avaliação depende bastante da natureza do problema, devendo-se observar o que se deseja, maximizar ou minimizar. No exemplo do PCV, uma função de avaliação possível apresentada por Linden (2012) é o somatório das distâncias entre as cidades e as suas sucessoras (de acordo com sequência definida pelo cromossomo). Uma outra abordagem poderia incluir no somatório a distância entre a posição inicial do caixeiro e a primeira cidade da sequência, bem como, definir que a distância a ser considerada entre uma cidade e outra é a menor possível utilizando rodovias (em vez da distância em linha reta).

Para exemplificar, comparando-se os cromossomos 1-2-3-4 e 3-1-2-4, supondo que a posição inicial do caixeiro viajante é na cidade de São Luís. O primeiro se traduz em um percurso iniciado em São Luís, seguido por Bom Jardim e Carolina e finalizado em Balsas, totalizando 1.088 km. O segundo se traduz em um percurso iniciado em São Luís, seguido por Carolina, São Luís e Bom Jardim e finalizado em Balsas, totalizando 2.608 km. Isto significa que o primeiro cromossomo é mais bem avaliado que o segundo.

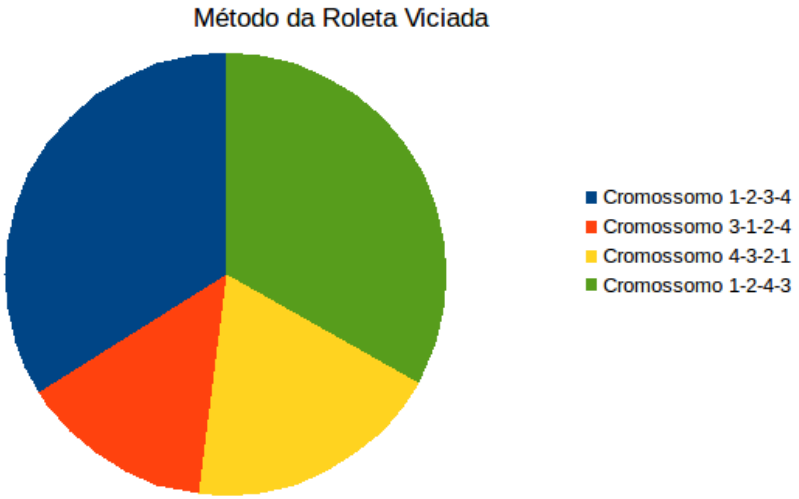
2.2.3 Seleção dos Pais (progenitores) - Método da Roleta Viciada

Uma vez que a população inicial foi gerada e todos os indivíduos (cromossomos) foram avaliados pela função de avaliação, inicia-se a criação de uma nova geração, a começar pela seleção dos seus progenitores. Nessa etapa, o propósito é simular a característica da evolução das espécies que determina que indivíduos mais bem adaptados têm maior probabilidade de se reproduzirem e passarem suas características genéticas adiante. Para esse fim, no contexto do AG, é preciso utilizar um método de seleção no qual indivíduos mais bem avaliados tenham maior probabilidade de serem selecionados como progenitores da próxima geração.

Um método comum entre pesquisadores de AGs, denominado Método da Roleta Viciada, é apresentado por Linden (2012). Descrevendo de modo ilustrativo: neste método, cada cromossomo recebe um pedaço da roleta proporcional à sua avaliação, sendo que a soma dos pedaços equivale à totalidade da mesma. A roleta é girada e o indivíduo sobre o qual ela parar será selecionado.

Para exemplificar, supondo que, além dos cromossomos 1-2-3-4 (de 1.088 km) e 3-1-2-4 (de 2.608 km) também estão presentes em uma população inicial de 4 indivíduos outros dois cromossomos, 4-3-2-1 e 1-2-4-3, que mapeiam percursos de 1.964 km e 1.112 km, respectivamente. Primeiramente, para que o método da roleta viciada funcione, a função de avaliação precisa ser de maximização, portanto, a função definida na Seção 2.2.2, que é de minimização, precisa ser adaptada. Desta forma, sendo f (x) o somatório das distâncias entre as cidades, a função g (x) = 1/f (x) será utilizada no método da roleta viciada. Isto significa que quanto menor o tamanho do percurso definido pelo cromossomo, maior o valor de g (x). No exemplo dado, para o cromossomo 1-2-3-4, g (x) = 0,000919118, para o cromossomo 3-1-2-4, g (x) = 0,000383436, para o cromossomo 4-3-2-1, g (x) = 0,000509165 e para o cromossomo 1-2-4-3, g (x) = 0,000899281. A FIGURA 7 demonstra a roleta viciada associada a situação descrita, pode-se observar, por exemplo, que o cromossomo com menor percurso, 1-2-3-4, tem 2,397 vezes mais chances de ser selecionado que o cromossomo com maior percurso, 3-1-2-4.

FIGURA 7 – ILUSTRAÇÃO DO MÉTODO DA ROLETA VICIADA



FONTE: O autor (2021).

2.2.4 Crossover

Uma vez que os progenitores foram selecionados para gerar integrantes da nova geração, é necessário utilizar um método que transmita informações genéticas de ambos os progenitores para o novo integrante. Este integrante, provavelmente, apresentará um padrão novo, porém manterá algumas semelhanças com seus pais. Esta etapa é denominada *crossover.*

Existem diversos métodos de *crossover* disponíveis na literatura, cada um adequado a tipos de problemas específicos. A título de exemplo, será explicado a seguir o funcionamento do *single-point crossover* apresentado por Amjad (2018).

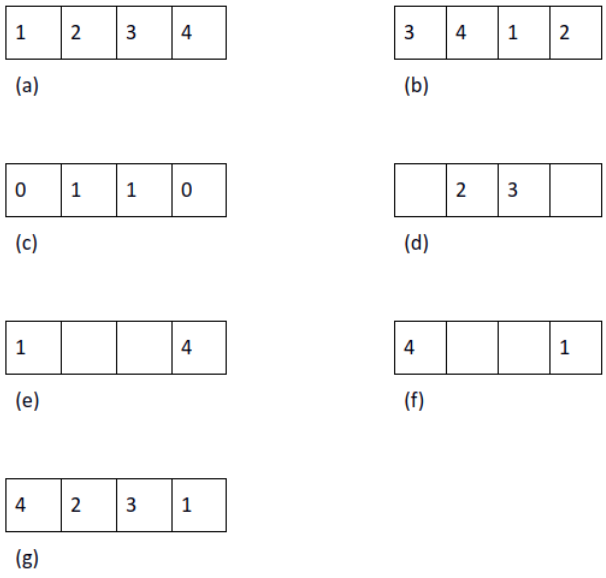
Sendo os progenitores denominados de “progenitor 1” e “progenitor 2”, o primeiro filho (integrante da nova geração) é formado pela união da primeira metade do progenitor 1 com a segunda metade do progenitor 2. O segundo filho é formado pela primeira metade do progenitor 2 e a segunda metade do progenitor 1. Este método, e outros semelhantes a ele, não são adequados para o problema do caixeiro viajante. A título de exemplo, um *single-point crossover* entre os cromossomos 1-2-3-4 e 4-3-2-1 geraria os filhos 1-2-2-1 e 4-3-3-2, ambos são soluções inadequadas para o problema, pois o caixeiro passaria por cidades repetidas e não passaria por todas as cidades.

Linden (2012) apresenta um método adequado para o PCV e que também se adequa ao problema apresentado neste trabalho. Trata-se do operador de crossover baseado em ordem. Seu funcionamento é descrito através dos seguintes passos:

1 Gere uma *string* de bits aleatória do mesmo tamanho que os elementos;
2 Copie para o filho 1 os elementos do pai 1 referente àquelas posições onde a *string* de bits possui um 1;
3 Faça uma lista dos elementos do pai 1 referentes a zeros da *string* de bit;
4 Permute esta lista de forma que os elementos apareçam na mesma ordem que no pai 2;
5 Coloque estes elementos nos espaços do filho 1 na ordem gerada no passo anterior;
6 Repita o processo para gerar o filho 2, substituindo o pai 1 pelo 2 e vice-versa.

A FIGURA 8 ilustra um possível resultado da aplicação do método descrito na sequência acima a dois progenitores denominados “progenitor 1” (FIGURA 8.a) e “progenitor 2” (FIGURA 8.b), que são cromossomos referentes ao PCV. A *string* de bits gerada no passo 1 é mostrada na FIGURA 8.c. Os resultados dos passos 2, 3 e 4 são mostrados na FIGURA 8 ‘d’, ‘e’ e ‘f’, respectivamente. A FIGURA 8.g mostra o filho 1, membro da próxima geração, gerado no passo 5.

FIGURA 8 - CROSSOVER BASEADO EM ORDEM



FONTE: O Autor (2021).

2.2.5 Mutação

Após alguns ciclos, as características genéticas de indivíduos bem avaliados se tornam predominantes na população e, gradativamente, os indivíduos vão se tornando mais parecidos uns com os outros, chegando ao ponto em que a variabilidade genética é pequena e não é mais possível evoluir apenas com o *crossover*, visto que progenitores muito parecidos entre si geram filhos que são,

geneticamente, suas cópias ou quase isso. Este problema pode ser solucionado, acrescentando-se mutações aos indivíduos da população.

A mutação consiste em realizar, com algum grau de aleatoriedade, modificações em partes dos cromossomos. Isto traz a diversidade genética de volta e possibilita novos ciclos de evolução. Existem diversos métodos de mutação disponíveis. Linden (2012) apresenta a permutação de elementos como método adequado para algoritmos genéticos baseados em ordem. Neste método, dois genes são selecionados aleatoriamente e suas posições são trocadas no cromossomo. Por exemplo, dado um cromossomo do PCV 1-2-3-4, se na seleção aleatório o primeiro e o último gene fossem selecionados, o mesmo cromossomo, após a mutação, seria alterado para 4-2-3-1.

2.2.6 Condição de parada

Os passos de 3 a 6 da sequência de passos para o algoritmo genético descrita na Seção 2.2 se repetem continuamente (criando novas gerações) até que uma condição de parada seja atendida. Tal condição é definida de acordo com as necessidades da aplicação. Algumas abordagens são:

- Definição inicial de um número limite de iterações, interrompendo o ciclo quando este limite for atingido;
- Definição de um valor de função de avaliação como meta, ou seja, assim que um cromossomo qualquer for avaliado melhor ou igual à meta, o ciclo é interrompido. Ressalta-se que, nesta abordagem, uma segunda condição de parada precisa ser definida, pois há a possibilidade de o ciclo se repetir indefinidamente;
- Definição de limite para o número de ciclos sem evolução. Entende-se que não houve evolução quando nenhum cromossomo de uma geração é melhor que o melhor cromossomo gerado nas gerações anteriores. Se a ausência de evolução se repetir consecutivamente, a quantidade de vezes estabelecida como limite, o ciclo é interrompido.

3 TRABALHOS RELACIONADOS

Junqueira (2015) aborda o planejamento conjunto da estivagem de contêineres em um navio e do empilhamento e desempilhamento de contêineres nos portos pelos quais esse navio passa. Ou seja, em vez de analisar isoladamente um pátio ou navio de contêineres específico, a técnica desenvolvida no referido trabalho avalia a forma como as decisões tomadas em determinado pátio afetarão a estivagem do navio. Além disso, também avalia os impactos nas operações dos pátios de contêineres dos demais portos em que este navio possui escala. São utilizados algoritmos genéticos e representação por regras para obtenção de soluções que diminuam a quantidade de movimentos improdutivos.

Carraro (2012) propõe a utilização de um algoritmo de Seleção Clonal como ferramenta de obtenção de soluções para o problema do empilhamento e desempilhamento de contêineres em pátios de contêineres. Esse algoritmo se enquadra na categoria dos algoritmos evolutivos e sua metodologia para a evolução de soluções é baseada em comportamentos presentes no sistema imunológico dos animais. Os resultados obtidos em simulações dos casos estáticos e dinâmicos, que serão explicados em seções posteriores deste trabalho, foram comparados aos obtidos pelas heurísticas do índice de rearranjo e da pilha mais baixa, dentre outras técnicas. O algoritmo de seleção clonal apresentou menor número de rearranjos se comparado às outras técnicas, porém para pilhas mais baixas (pátios menos cheios) todas as técnicas apresentaram resultados semelhantes.

As heurísticas da pilha mais baixa (WAN; LIU; TSAI, 2009) e do índice de rearranjo (MURTY et al., 2005) serão utilizadas para comparação de desempenho com a técnica desenvolvida neste trabalho, portanto, ambas serão explicadas de forma mais detalhada a seguir.

A heurística da pilha mais baixa é uma abordagem simples que, como o próprio nome sugere, determina que, sempre que houver necessidade de rearranjo de um contêiner, este será movido para a pilha com a menor quantidade de contêineres, se houver mais de uma pilha com esta mesma quantidade, o contêiner rearranjado poderá ser movido para qualquer uma delas.

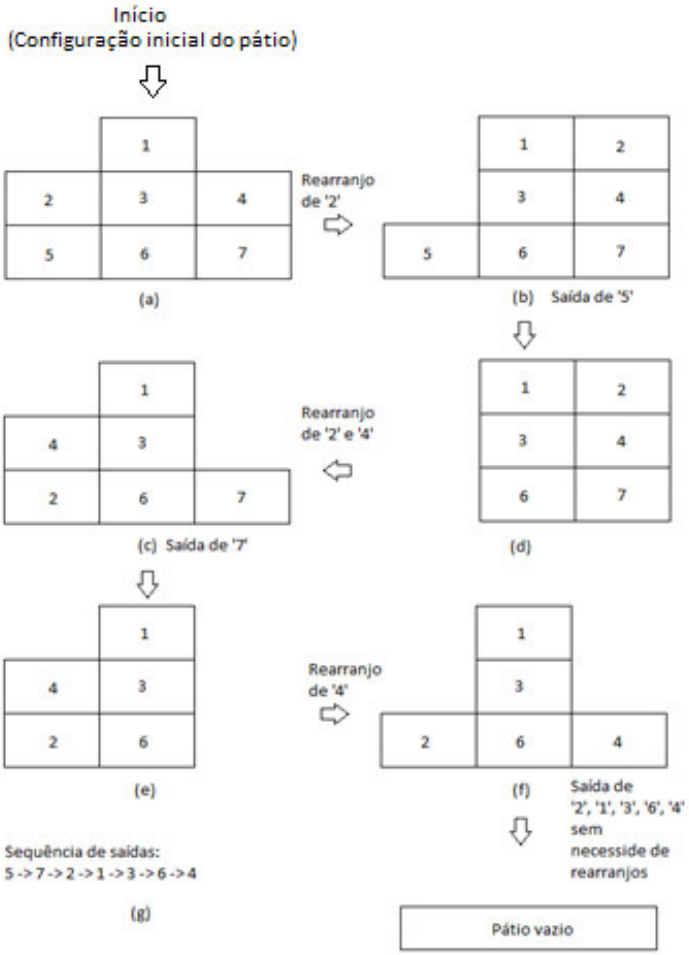
A FIGURA 9 ilustra o funcionamento desta heurística. Neste exemplo, os contêineres, representados por retângulos e identificados por números, alocados no pátio de 3 pilhas mostrados em FIGURA 9.a devem ser removidos conforme

sequência mostrada em FIGURA 9.g. Para que o contêiner ‘5’ seja removido, o contêiner ‘2’ precisa ser rearranjado. Conforme heurística da pilha mais baixa, esse rearranjo será da 1ª pilha para a 3ª, que é menor que a 2ª (FIGURA 9.b). Após rearranjo, o contêiner ‘5’ poderá ser removido (FIGURA 9.d). Em seguida, os contêineres ‘2’ e ‘4’ serão rearranjados para a 1ª pilha (FIGURA 9.c) e o contêiner ‘7’ poderá então ser removido (FIGURA 9.e). O contêiner ‘4’ será novamente rearranjado para a 3ª pilha (FIGURA 9.f), permitindo a retirada do contêiner ‘2’. Os contêineres ‘1’, ‘3’, ‘6’ e ‘4’ serão então removidos sem necessidade de rearranjos e o pátio estará então vazio. Ao todo, após esvaziamento do pátio, foram necessários 4 movimentos improdutivos.

A heurística do índice de rearranjo, por sua vez, utiliza uma métrica melhor para selecionar a pilha de destino, sendo essa a única diferença entre as duas heurísticas. Trata-se do índice de rearranjo, que pode ser definido da seguinte forma: para um dado contêiner a ser rearranjado, o índice de rearranjo de uma pilha para a qual ele pode ser rearranjado consiste na quantidade de contêineres nesta pilha que sairão antes do contêiner a ser rearranjado (conforme sequência de retiradas).

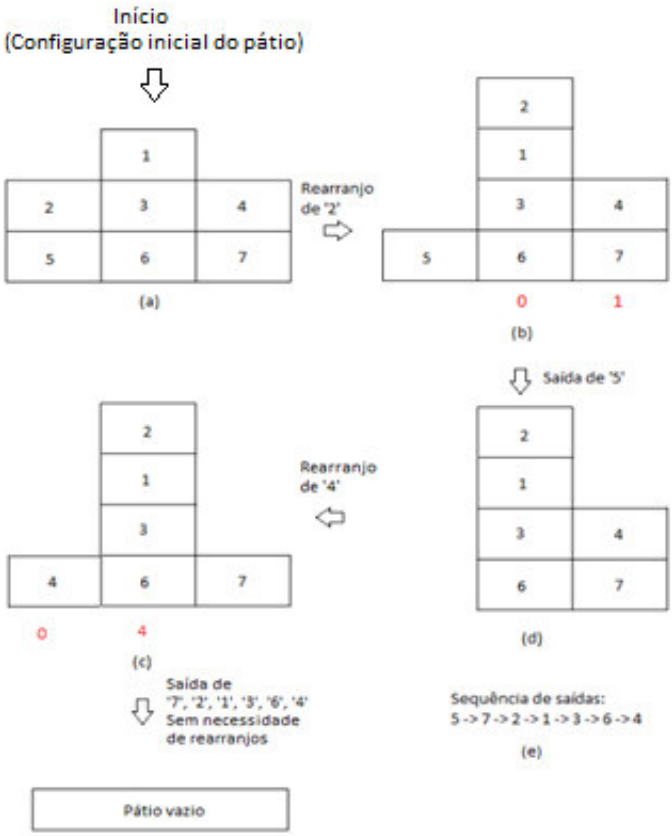
A FIGURA 10 ilustra a aplicação da heurística do índice de rearranjo à mesma situação na qual foi aplicada a heurística da pilha mais baixa no exemplo anterior. As duas heurísticas divergem no primeiro rearranjo (FIGURA 10.b), visto que a 2ª pilha, apesar de maior que a 3ª, não possui nenhum contêiner a ser removido antes do contêiner 4 na sequência de remoções (a 3ª pilha possui 1), desta forma, a 2ª pilha é a que possui menor índice de rearranjo e o contêiner 4 será rearranjado para ela nesta etapa. Ao todo, após esvaziamento do pátio, foram necessários 2 movimentos improdutivos.

FIGURA 9 - HEURÍSTICA DA PILHA MAIS BAIXA



FONTE: O autor (2021).

FIGURA 10 - HEURÍSTICA DO ÍNDICE DE REARRANJO



FONTE: O autor (2021).

4 METODOLOGIA

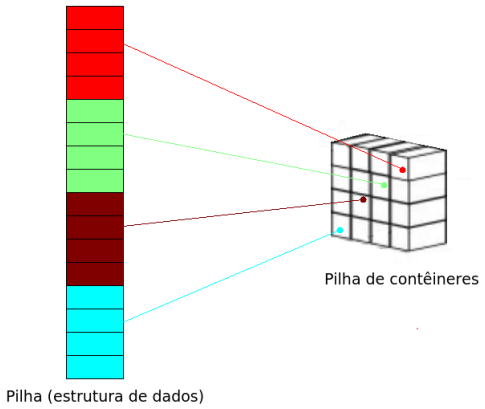
A solução baseada em algoritmos genéticos desenvolvida neste trabalho segue as mesmas etapas do AG descritas na Seção 2.2. Para cada uma dessas etapas e de acordo com os critérios definidos em Linden (2012), o método mais adequado ao tipo de problema foi selecionado e, quando necessário, adaptado. As próximas seções descrevem as entradas do problema e a metodologia desenvolvida para cada uma das etapas citadas.

4.1 ENTRADAS DO PROBLEMA

As informações de entrada do problema são modeladas nas seguintes estruturas de dados: uma sequência de saídas, informando a ordem em que os contêineres serão retirados do pátio, e uma lista de pilhas, informando a configuração atual do pátio, ou seja, em qual *slot* cada contêiner se encontra. Nesta Seção, a palavra ‘pilha’ se refere à estrutura de dados e a expressão ‘pilha de contêineres’ se refere à pilha no contexto de pátio de contêineres, conforme nomenclatura apresentada na Seção 2.1.2.

Visto que a retirada de um contêiner de uma determinada coluna envolve a realocação de todos os contêineres das colunas a sua frente na mesma pilha de contêineres, pode-se modelar cada pilha de contêineres como uma única pilha, sendo que as colunas à frente na pilha de contêineres ficam acima na pilha (FIGURA 11). Essa modelagem possibilita que todo o pátio de contêineres seja representado pela lista de pilhas anteriormente citada.

FIGURA 11 - MODELAGEM DA PILHA DE CONTÊINER COMO UMA ÚNICA PILHA (ESTRUTURA DE DADOS)



FONTE: O autor (2021).

4.2 CROMOSSOMO E REGRA DE TRADUÇÃO

Na solução desenvolvida, o cromossomo consiste em uma sequência de movimentos, que podem ser de saída do pátio ou de rearranjo de uma pilha para a outra. Para que o cromossomo seja considerado válido, as seguintes restrições precisam ser atendidas:

- A sequência dos movimentos de saída do pátio precisa ser a mesma definida na lista de saídas recebida como entrada;
- Um contêiner pode ser movimentado somente se estiver no topo de uma pilha, levando em conta a lista de pilhas atualizada com as movimentações anteriores;
- Um contêiner só pode ser rearranjado para uma pilha que não esteja cheia, levando em conta a lista de pilhas atualizada com as movimentações anteriores.

A FIGURA 9, utilizada para ilustrar o funcionamento da heurística da pilha mais baixa, apresenta uma sequência de movimentos válidos para a configuração de pátio mostrada na FIGURA 9.a e a sequência de saídas mostrada na FIGURA 9.g. Um cromossomo que define esta mesma sequência de movimentos é mostrado na FIGURA 12 (considerando que na FIGURA 9 as pilhas são numeradas de 1 a 3 da esquerda para a direita).

FIGURA 12 - REPRESENTAÇÃO DE UM CROMOSSOMO

Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6
Tipo: Rearranjo Contêiner: 2 Pilha de destino: 3	Tipo: Saída Contêiner: 5	Tipo: Rearranjo Contêiner: 2 Pilha de destino: 1	Tipo: Rearranjo Contêiner: 4 Pilha de destino: 1	Tipo: Saída Contêiner: 7	Tipo: Rearranjo Contêiner: 4 Pilha de destino: 3

Gene 7	Gene 8	Gene 9	Gene 10	Gene 11
Tipo: Saída Contêiner: 2	Tipo: Saída Contêiner: 1	Tipo: Saída Contêiner: 3	Tipo: Saída Contêiner: 6	Tipo: Saída Contêiner: 4

FONTE: O autor (2021).

Para garantir que um cromossomo seja sempre válido, cada modificação do valor de um gene deve levar em conta as restrições descritas anteriormente. Para tal, a lista de pilhas atualizada precisa sempre ser consultada antes de qualquer modificação. Esta lista é definida como aquela obtida após a execução, sobre a lista de pilhas de entrada, dos movimentos definidos pelos genes anteriores ao gene cujo valor pretende-se modificar. Desta forma, é possível verificar se o contêiner a ser movimentado está no topo da pilha antes de definir o movimento como do tipo “saída” ou, em caso de rearranjo, sempre selecionar a pilha de destino dentro do subconjunto das pilhas que não estão cheias e são diferentes da pilha na qual o contêiner a ser rearranjado se encontra.

A FIGURA 12 apresenta, na verdade, o resultado da aplicação da regra de tradução sobre o cromossomo, já fornecendo instruções inteligíveis, do ponto de vista dos operadores do pátio. O cromossomo em si consiste na estrutura de dados utilizada na implementação. Em decorrência da utilização de linguagem orientada a objetos, a tradução do cromossomo para a informação inteligível é trivial.

4.3 INICIALIZAÇÃO DA POPULAÇÃO

Inicializar a população consiste em inserir nela cromossomos aleatórios até que seja atingido o tamanho da população definido por meio de parâmetro.

O PSEUDOCÓDIGO 1 define a abordagem utilizada para a criação de um cromossomo aleatório durante a inicialização da população. Ressalta-se que a estrutura de dados descrita como “lista_de_pilhas_atualizada” está sempre atualizada conforme descrito na Seção 4.2, apesar de o pseudocódigo suprimir a forma como isto é feito na implementação real.

A FIGURA 13 demonstra a aplicação de parte dessa sequência de passos em uma situação ilustrativa, com “lista de saídas” e “lista de pilhas” definidas na figura. Nesse exemplo, considera-se que a altura máxima da pilha é de 10 contêineres. Ressalta-se que a explicação fará referência à FIGURA 13 e às linhas do pseudocódigo apresentado em PSEUDOCÓDIGO 1. Inicialmente, o “contêiner 2” é definido como “conteiner_a_ser_retirado” (linha 1). Em seguida, a pilha 2 é definida como “pilha” e é percorrida a partir do topo (linhas 2, 3 e 4), definindo “contêiner 4” como o primeiro “contêiner_a_ser_rearranjado”. Nesta etapa, as pilhas para as quais o “contêiner 4” pode ser rearranjado são “pilha 1”, “pilha 2” e “pilha 3”. A “pilha_de_destino” é selecionada, aleatoriamente, dentre elas (linha 5).

Nesse exemplo, a pilha selecionada foi a “pilha 3”. Em seguida, o “gene 1” é incluído no cromossomo (linha 6). O processo então se repete com a definição do “contêiner 3” como o próximo “contêiner_a_ser_rearranjado”. Nessa etapa, observa-se que a “pilha 3” não está entre as pilhas para os quais o “contêiner 3” pode ser rearranjado, visto que, com a execução do movimento definido pelo gene 1, a pilha 3 atingiu a altura máxima de 10 contêineres. No exemplo, a pilha selecionada foi a “pilha 4” e o “gene 2” foi inserido no cromossomo. Finalmente, o “gene 3”, que define a saída do “contêiner 2” do pátio, é inserido no cromossomo (linha 7). Cada contêiner presente em “lista_de_saídas” será definido como “contêiner_a_ser_retirado”, seguindo a sequência definida por esta lista. Todo o processo descrito se repetirá para todos estes contêineres.

PSEUDOCÓDIGO 1 - CRIAÇÃO DE CROMOSSOMO ALEATÓRIO E INICIALIZAÇÃO DA POPULAÇÃO

```
1   for contêiner_a _ser_retirado in senquência_de_saídas:
2       pilha = contêiner_a_ser_retirado .retorna_pilha (lista_de_pilhas) // Encontra a pilha em que contêiner_a_ser_retirado se encontra em lista_de_pilhas
3       pilha = pilha [contêiner_a_ser_retirado.posição() +1 : ] //pilha passa a ser a sub lista que vai da primeira posição superior ao contêiner_a_ser_retirado até o topo da pilha.
4       for contêiner_a_ser_rearranjado in reverso (pilha) //percorre a pilha de cima para baixo a partir do topo
5           pilha_de_destino = selecionar_pilha_válida (lista_de_pilhas_atualizada) // seleciona aleatoriamente uma pilha_de_destino para o contêiner_a_ser_rearranjado dentre as pilhas não cheias e diferentes de pilha
```

Continua

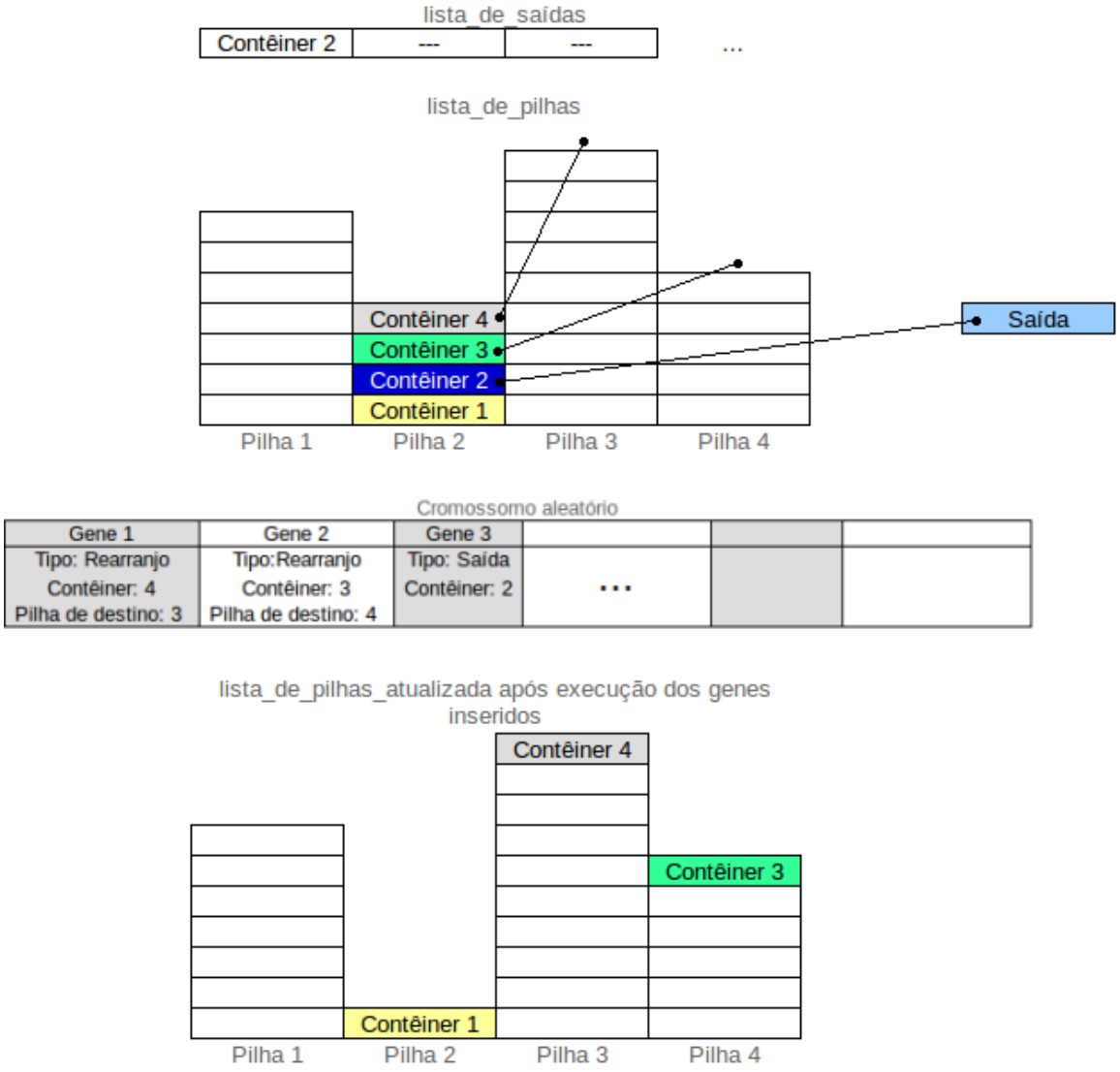
```
6                   Incluir_gene_no_cromossomo (
                        REARRANJO, // tipo de movimento
                        contêiner_a_ser_rearranjado,
                        pilha_de_destino)

7               incluir_gene_no_cromossomo (
                    SAÍDA, // tipo de movimento
                    contêiner_a_ser_retirado)
```

FONTE: O autor (2021).

Observa-se que, no passo 3, pode ocorrer de não haver nenhuma pilha válida para a qual o contêiner possa ser rearranjado. Neste caso, seguir com a criação do cromossomo se torna impossível e o mesmo é considerado natimorto, sendo descartado. Quanto maior o número de contêineres no pátio em relação à sua capacidade máxima, maior a probabilidade de genes natimortos serem gerados e, para determinadas configurações de pátio (listas de pilhas) e sequências de saídas, a criação de cromossomos que não sejam natimortos é impossível. Desta forma, um número máximo de tentativas de criação de cromossomos é definido para eliminar a possibilidade de o programa executar indefinidamente. Quando esse número de tentativas é atingido, a execução é interrompida e o cromossomo mais bem avaliado é retornado como solução. Se nenhum cromossomo chegou a ser criado, considera-se que não foi possível encontrar uma solução.

FIGURA 13 - CRIAÇÃO DE CROMOSSOMO ALEATÓRIO E INICIALIZAÇÃO DA POPULAÇÃO



FONTE: O autor (2021).

4.4 FUNÇÃO DE AVALIAÇÃO E SELEÇÃO DOS PAIS

A avaliação de um cromossomo está diretamente ligada à quantidade de movimentos do tipo “rearranjo” que ele possui. Quanto menor a quantidade de movimentos deste tipo, mais eficiente é a solução e mais bem avaliado deve ser o cromossomo. Sendo “r” o número de movimentos de rearranjos, a função de avaliação seria definida como f(r) = 1/(1+r). A adição de 1 ao divisor evita a tentativa de divisão por zero em soluções sem movimentos de rearranjo. Entretanto, esta função, utilizada juntamente com o método da roleta viciada, não seria uma abordagem eficaz. O motivo é que, em situações reais, a diferença entre as quantidades de movimentos

improdutivos de uma solução para a outra é pequena se comparada aos totais de movimentos improdutivos destas soluções. Isto resulta em valores de f(r) muito próximos e, consequentemente, baixa eficácia do método da roleta viciada.

A título de exemplo, se um cromossomo A definisse 1100 rearranjos e outro cromossomo B definisse 1000 rearranjos. O cromossomo B teria, aproximadamente, 1,1 vezes mais chance de ser selecionado no método da roleta viciada do que A, apesar de ser uma solução significativamente melhor.

Linden (2012) apresenta como solução para este problema a **normalização não linear**, que consiste em passar os valores da função de avaliação por uma função não linear que faça com que as avaliações sejam mais discriminatórias das diferenças entre os indivíduos. Dessa forma, com o objetivo de aumentar as diferenças entre as soluções e, consequentemente, tornar a seleção através do método da roleta viciada mais eficaz, a função de avaliação é definida como:

$$\mathbf{f(r) = 1/(1+r^3)}$$

Onde:

r = número de movimentos do tipo “rearranjo”

Com esta função de avaliação, o cromossomo B, do exemplo citado anteriormente, passaria a ter 1,3 vezes mais chances de ser selecionado no método da roleta viciada do que A.

4.5 CROSSOVER

A metodologia de *crossover* desenvolvida se baseia no *crossover* voltado para problema de ordenação descrito na Seção 2.2.4. O PSEUDOCÓDIGO 2 demonstra o funcionamento do método desenvolvido, sendo que “progenitor_0” e “progenitor_1” são os dois progenitores selecionados através do método da roleta viciada e “lista_binária” é uma lista com o mesmo comprimento da entrada “lista_de_saídas” e formada por zeros e uns distribuídos de modo aleatório. Ressalta-se também que cada cromossomo possui sua própria “lista_de_pilhas_atualizada”, que é atualizada de acordo com a execução dos movimentos definidos pelos genes do cromossomo. Inicialmente, antes de se levar em conta a execução de qualquer movimento, todas as “listas_de_pilhas_atualizadas” são iguais à entrada “lista_de_pilhas”.

Esta sequência de passos é parecida com a utilizada para criação do gene aleatório, a diferença é que, em vez de selecionar aleatoriamente uma pilha de destino dentre as pilhas possíveis para os movimentos de rearranjo, o *crossover* utiliza um dos progenitores como referência para determinar a pilha de destino (linhas 6 a 16 do PSEUDOCÓDIGO 2). O progenitor de referência é definido pelo valor contido em "lista_binária" (0 para "progenitor_0", 1 para "progenitor_1"). Uma vez que ele é selecionado, é necessário verificar em que pilha da sua "lista_de_pilhas_atualizada" (atualizada até o gene que define a saída do "contêiner_a_ser_retirado") o "contêiner_a_ser_rearranjado" se encontra (linhas 18 a 21 do PSEUDOCÓDIGO 2) e, se possível, definir esta pilha como a "pilha_de_destino" do gene a ser inserido no cromossomo filho. Se não for possível, ou seja, se a pilha estiver cheia na "lista_de_pilhas_atualizada_do_filho", o outro progenitor é tomado como referência. Se também não for possível, uma "pilha_de_destino" é selecionada aleatoriamente.

Nesta etapa, pode ocorrer de não haver nenhuma pilha válida para a qual o contêiner possa ser rearranjado, portanto, o cromossomo filho será dado como natimorto e será descartado. A tentativa de *crossover* entre os progenitores selecionados será interrompida e o processo será então reiniciado a partir da seleção de novos progenitores. Um número máximo de tentativas consecutivas e malsucedidas de *crossovers* é definido para eliminar a possibilidade de o programa executar indefinidamente. Quando esse número de tentativas é atingido, a execução é interrompida e o cromossomo mais bem avaliado (dentre todos os cromossomos de todas as gerações) é retornado como solução.

PSEUDOCÓDIGO 2 - CROSSOVER

```
1   for contêiner_a_ser_retirado in sequência_de_saídas:
2       progenitor_de_referência = lista_binária
        [contêiner_a_ser_retirado.retornar_índice()] //atribui à variável
        progenitor_de_referência o valor (0 ou 1) que consta em lista_binária no
        mesmo índice do contêiner_a_ser_retirado em sequência_de_saídas
3       pilha = contêiner_a_ser_retirado.retorna_pilha
        (lista_de_pilhas_atualizada_do_filho) // Encontra a pilha em que
        contêiner_a_ser_retirado se encontra em lista_de_pilhas_atualizada_do_filho
4       pilha = pilha [contêiner_a_ser_retirado.posição() +1 : ] // pilha passa a ser a
        sub lista que vai da primeira posição superior ao contêiner_a_ser_retirado até
        o topo da pilha.
```

Continua

```
5           for contêiner_a_ser_rearranjado in reverso (pilha): //percorre a pilha de
            cima para baixo a partir do topo
6               if progenitor_de_referência == 0
7                   pilha_de_destino = encontra_pilha_de_destino
                    (lista_de_pilhas_atualizada_do_progenitor_0,
                    lista_de_pilhas_atualizada_do_filho,
                    contêiner_a_ser_rearranjado)
8                   if pilha_de_destino == FALSE
9                       pilha_de_destino = encontra_pilha_de_destino
                        (lista_de_pilhas_atualizada_do_progenitor_1,
                        lista_de_pilhas_atualizada_do_filho,
                        contêiner_a_ser_rearranjado)
10              else // progenitor_de_referência == 1
11                  pilha_de_destino = encontra_pilha_de_destino
                    (lista_de_pilhas_atualizada_do_progenitor_1,
                    lista_de_pilhas_atualizada_do_filho,
                    contêiner_a_ser_rearranjado)
12                  if pilha_de_destino == FALSE
13                      pilha_de_destino = encontra_pilha_de_destino
                        (lista_de_pilhas_atualizada_do_progenitor_0,
                        lista_de_pilhas_atualizada_do_filho,
                        contêiner_a_ser_rearranjado)
14              if pilha_de_destino == FALSE
15                  pilha_de_destino = selecionar_pilha_válida
                    (lista_de_pilhas_atualizada_do_filho) // seleciona
                    aleatoriamente uma pilha_de_destino para o
                    contêiner_a_ser_rearranjado dentre as pilhas não cheias do
                    filho e diferentes de pilha
16              Incluir_gene_no_cromossomo (
                    REARRANJO, // tipo de movimento
                    contêiner_a_ser_rearranjado,
                    pilha_de_destino)

17          incluir_gene_no_cromossomo (
                SAÍDA, // tipo de movimento
                contêiner_a_ser_retirado)
```

Continua

```
18      function encontra_pilha_de_destino (lista_de_pilhas_atualizad_do_progenitor,
        lista_de_pilhas_atualizada_do_filho, contêiner_a_ser_rearranjado)
19              pilha_de_destino =
                lista_de_pilhas_atualizada_do_progenitor.encontra_pilha
                (contêiner_a_ser_rearranjado) //encontra em que pilha o
                contêiner_a_ser_rearranjado se encontra na
                lista_de_pilhas_atualizada_do_progenitor, sendo esta lista atualizada até a
                execução do movimento definido pelo gene do cromossomo do progenitor
                que define a saída do contêiner_a_ser_retirado
20                      if (lista_de_pilhas_atualizada_do_filho.verifica_se_válida
                        (pilha_de_destino, contêiner_a_ser_rearranjado)) //a função retorna
                        FALSE caso, na lista_de_pilhas_atualiza_do_filho, a pilha_de_destino
                        esteja cheia ou seja a mesma pilha na qual o
                        contêiner_a_ser_rearranjado se encontra
                                return FALSE
21                      else
                                return pilha_de_destino
```

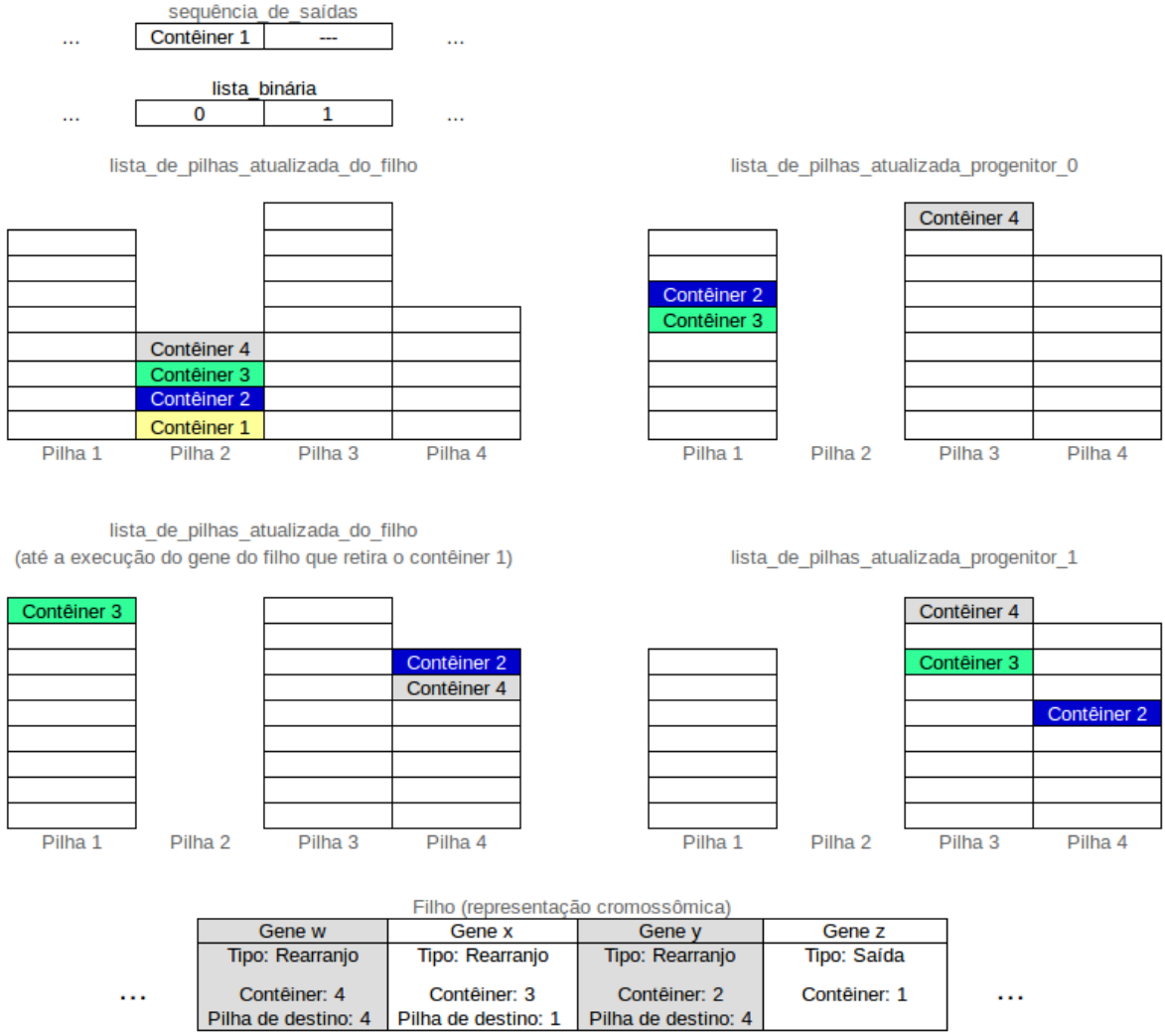
FONTE: O autor (2021).

A FIGURA 14 exemplifica uma etapa da criação de um novo indivíduo através do método do *crossover*. Neste exemplo, o contêiner a ser retirado da lista_de_pilhas_atualizada_do_filho é o contêiner 1, conforme definido em sequência_de_saídas. Para isto será necessário rearranjar os contêineres 4, 3 e 2, nesta ordem. Conforme lista_binária, o progenitor de referência será o progenitor_0.

Para o contêiner 4, verifica-se a situação em que ambos os progenitores não podem ser utilizados como referência, pois em suas listas de pilhas atualizadas o contêiner 4 se encontra na pilha 3, que está cheia em lista_de_pilhas_atualizada_do_filho (considerando 9 a altura máxima da pilha neste exemplo). Nesse caso, a pilha de destino para contêiner 4 precisa ser gerada aleatoriamente (gene w).

Para o contêiner 3, progenitor_0 é tomado como referência e a pilha de destino é definida como a pilha 1 (gene x). Para contêiner 2, progenitor_0 não pode ser tomado como referência, pois, após execução do gene x, a pilha 1 estará cheia em lista_de_pilhas_atualizada_do_filho, nesse caso, progenitor_1 é tomado como referência e o contêiner 2 é realocado para a pilha 4 (gene y). Finalmente, o contêiner 1 é removido (gene z).

FIGURA 14 - CROSSOVER



FONTE: O autor (2021).

4.6 MUTAÇÃO

A mutação é implementada através de uma pequena alteração na metodologia de *crossover* apresentada na Seção 4.5. Este *crossover* modificado é utilizado na criação de um percentual dos indivíduos da nova geração, percentual este que é definido pelo parâmetro “percentual de mutação”.

A modificação no método *crossover*, que cria o método *crossover_mutação* é a seguinte: para cada gene, existe uma probabilidade (definida pelo parâmetro “grau de mutação”) de ele ser criado sem tomar como referência nenhum dos progenitores, da mesma forma que ocorre na criação de cromossomos aleatórios descrita na Seção 4.3. O parâmetro “condição de mutação”, por sua vez, define quantas iterações são necessárias para que a mutação seja ativada.

Para exemplificar, configurando-se os parâmetros desta forma: “condição de mutação” = 40, “fator de mutação” = 30 e “grau de mutação” = 20, significa que, sempre que ocorrerem 40 iterações consecutivas sem evolução, a geração seguinte será criada com mutação em 30% dos seus indivíduos (cromossomos). Na criação de cada um dos genes dos cromossomos selecionados para mutação, haverá 20% de chance de ele ser criado sem tomar como referência nenhum dos progenitores, ou seja, aleatoriamente. O método “*crossover_mutação”* lida com a ocorrência de cromossomos natimortos da mesma forma que o método *crossover* (Seção 4.5).

4.7 CONDIÇÃO DE PARADA E SOLUÇÃO RETORNADA

A execução é interrompida quando ocorre uma sequência de iterações consecutivas sem que haja evolução e a quantidade de iterações nesta sequência se iguala ao valor definido pelo parâmetro “condição de parada”. Entende-se que há evolução em uma iteração quando ao menos um dos cromossomos criados nesta iteração é mais bem avaliada que o cromossomo mais bem avaliado dentre os criados nas gerações anteriores.

Atrelar o encerramento da execução à ocorrência de uma sequência de iterações consecutivas e sem evoluções é uma abordagem adequada, pois a ocorrência desta sequência indica que a população convergiu para uma condição de pouca variabilidade genética, ou seja, é formada por cromossomos muito parecidos entre si, o que não favorece a evolução. Por outro lado, se alguma evolução for identificada, significa que continuar com a execução pode gerar mais evoluções, o que se alinha com o objetivo de encontrar a melhor solução possível.

No momento em que a execução é interrompida, é retornado, como solução, o cromossomo mais bem avaliado dentre todos os cromossomos gerados em todas as iterações. As Seções 4.3, 4.5 e 4.6 apresentaram outras condições de parada relacionadas à quantidade de tentativas malsucedidas e consecutivas de criar novos cromossomos (cromossomos natimortos).

5 EXPERIMENTOS E RESULTADOS

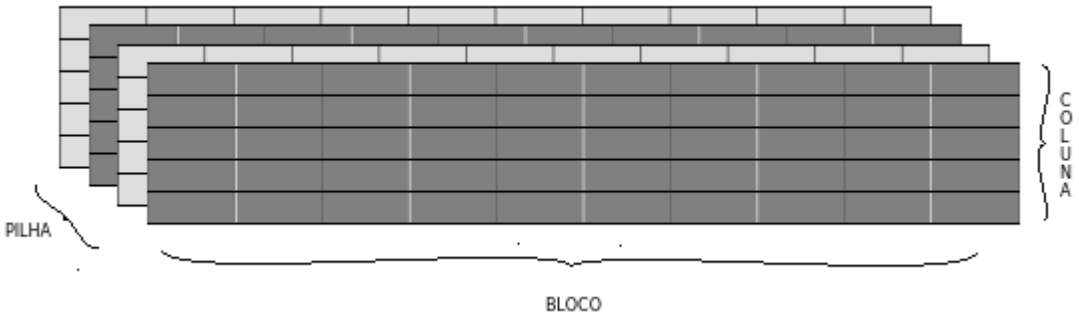
A implementação computacional da solução proposta foi desenvolvida na linguagem de programação C++. Os experimentos foram realizados em computador com as seguintes configurações: processador Intel core i7, 16 Gb de memória RAM e HD de 1T. Nas seções seguintes, serão descritos os casos de testes utilizados nas simulações, será apresentado um estudo de sensibilidade paramétrica e também um comparativo dos resultados das simulações com os resultados apresentados por soluções da literatura aplicadas aos mesmos casos de teste.

5.1 CASOS DE TESTES

Os casos de testes utilizados nas simulações se baseiam no caso estático descrito por Carraro (2012), no qual, a partir de uma configuração de pátio de contêineres e de uma sequência de saídas de contêineres gerados aleatoriamente, a implementação deve apresentar como resultado uma sequência de movimentos que resultem no total esvaziamento do pátio. Carraro (2012) também descreve um caso dinâmico, no qual é levada em conta a chegada de novos contêineres durante o processo de esvaziamento do pátio. Este caso não será abordado.

Neste trabalho, todas as simulações foram realizadas tendo como entradas pátios de contêineres formados por blocos de tamanho 10, que, por sua vez, são formados por pilhas de tamanho 20 (pátio 10x20). A implementação é indiferente à altura das colunas. A FIGURA 15 ilustra essa configuração levando em conta colunas com cinco contêineres de altura, o que é prática comum em situações reais. Quatro colunas enfileiradas, com cinco contêineres de altura cada, formam uma pilha de tamanho 20. Ressalta-se que o tamanho da pilha se refere à quantidade de contêineres e não à quantidade de colunas.

FIGURA 15 - PÁTIO DE CONTÊINERES 10X20



FONTE: O autor (2021).

O método implementado em C++ “geraPatioAleatorio”, utilizado para gerar esses pátios, é demonstrado no PSEUDOCÓDIGO 3. O método recebe como parâmetros os números mínimo e máximo de contêineres que o pátio a ser gerado pode ter (linha 1). Na linha 3, é gerado, aleatoriamente, um valor entre “numeroMinimoDeConteineres” e “numeroMaximoDeConteienres”, que representa a quantidade de contêineres que o pátio deverá ter. Isto possibilita que os testes sejam feitos com diferentes níveis de lotações dos pátios e que sejam evitados os pátios muito vazios, nos quais identificar a melhor sequência de movimentos seria uma tarefa trivial, e os muito cheios, nos quais não há sequência de movimentos válida para a retirada dos contêineres na sequência de saídas predefinida.

PSEUDOCÓDIGO 3 - GERAÇÃO DO PÁTIO DE CONTÊINERES PARA CASOS DE TESTES IMPLEMENTADO EM C++

```
1  patio::geraPatioAleatorio (numeroMinimoDeConteineres, numeroMaximoDeConteineres)
2  {
3     numeroDeConteineres = valor_aleatório_entre ( numeroMinimoDeConteineres, numeroMaximoDeConteineres);
4     //quantidade de conteineres. Valor aleatorio
5
6     for (numeroDeConteineres; numeroDeConteineres >0;numeroDeConteineres--)
7     {
8          indicePilha = pilha_aleatoria (this->pilhasPossiveis);
9          // gera pilha aleatoria para o conteiner a ser colocado
10
11         c = new conteiner (numeroDeConteineres-1);
12         //recebe o codigo do conteiner como entrada
13
14         armazena (c, indicePilha);
15    }
16 }
```

FONTE: O autor (2021).

O laço de repetição entre as linhas 6 e 15 construirá o pátio inserindo um contêiner por vez. Para cada novo contêiner, será escolhida, aleatoriamente, dentre as pilhas possíveis, a pilha na qual o novo contêiner será alocado (linha 8). Na linha 14 é utilizada o método armazena, que recebe como entrada um contêiner e uma pilha, e aloca esse contêiner no topo dessa pilha. Descrevendo em termo de colunas,

pode-se dizer que o contêiner será alocado no topo da coluna não cheia que se encontra mais atrás na pilha.

Todos os pátios criados através deste método obedecem às seguintes restrições:

- Uma pilha não recebe mais contêineres que o limite que comporta;
- Não há “contêineres flutuantes”, ou seja, nenhum contêiner é alocado em uma posição da pilha sem que todas as posições abaixo estejam ocupadas;
- Uma coluna recebe contêineres somente se todas as colunas atrás dela na mesma pilha estiverem cheias.

Dentre as restrições acima, apenas a última não é uma restrição em situações reais, porém, em pátios nos quais não é possível retirar contêineres de uma coluna sem retirar todos os contêineres das colunas a sua frente, obedecer a esta restrição sempre será a melhor opção se o objetivo for minimizar movimentos improdutivos.

As outras entradas das simulações, as “sequências de saídas” de cada pátio são obtidas através da criação de listas que contenham todos os contêineres do pátio e são ordenadas de modo aleatório.

5.2 SENSIBILIDADE PARAMÉTRICA

Os seguintes parâmetros serão analisados:

- Tamanho da população. Valores experimentados para este parâmetro: 25, 50, 75, 100 e 125;
- Condição de mutação. Valores experimentados para este parâmetro: 20, 40, 60, 80 e 100;
- Percentual de mutação. Valores experimentados para este parâmetro: 10, 20, 30 e 40;
- Grau de mutação. Valores experimentados para este parâmetro: 20, 40 e 60;
- Condição de parada. Valores experimentados para este parâmetro: 40, 60, 80, 100 e 120.

Para cada valor de parâmetro, foram realizadas 50 execuções do AG. Para cada execução, são geradas as entradas, conforme descrito na Seção 5.1, sendo que o número de contêineres de cada pátio 10x20 é fixado em 140 (70% de lotação). Em cada execução são aplicadas sobre as entradas, a técnica desenvolvida neste trabalho e a solução da literatura “índice de rearranjo”. Cada uma tentará gerar uma sequência de movimentos que resulte no total esvaziamento do pátio. Para cada valores de parâmetros, serão apresentadas graficamente as diferenças entre as quantidades médias de movimentos improdutivos das soluções geradas por cada técnica, bem como o tempo médio de execução da técnica desenvolvida neste trabalho.

Optou-se por avaliar o desempenho utilizando como métrica a diferença para uma outra técnica em vez de considerar, simplesmente, a quantidade de movimentos improdutivos, visto que, pelo fato de cada execução ser aplicada a um pátio diferente gerado de forma aleatório, a variância entre os valores absolutos de movimentos improdutivos é grande, o que não ocorre com a métrica escolhida.

5.2.1 Tamanho da população

Para avaliação deste parâmetro, os demais parâmetros são fixados nos seguintes valores:

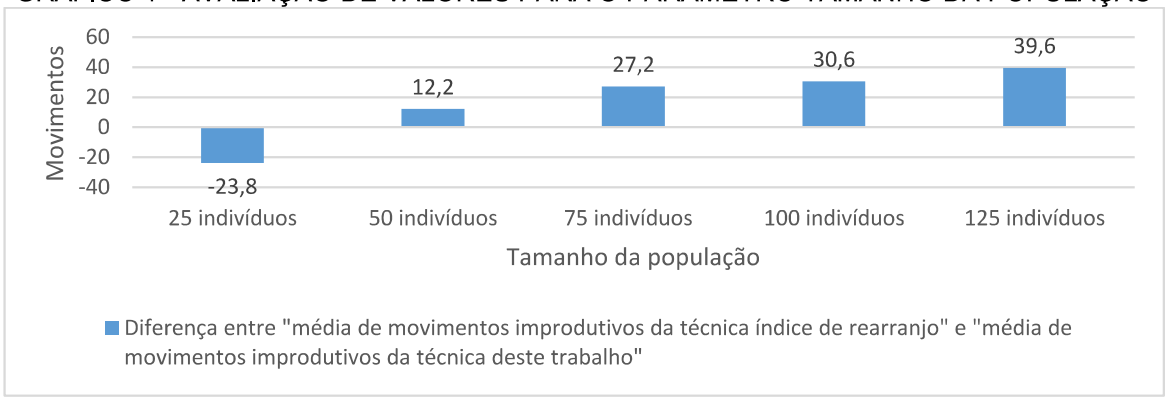
- Condição de parada em 120;
- Condição de mutação em 40;
- Percentual de mutação em 20;
- Grau de mutação em 40.

O GRÁFICO 1 mostra o aumento no desempenho diretamente proporcional ao aumento da população. Não é possível identificar um pico ou uma estabilização, o que indica que os valores do parâmetro avaliado poderiam continuar sendo aumentados para verificar uma possível melhora no desempenho, entretanto, conforme mostrado no GRÁFICO 2, o tempo de execução com o parâmetro “tamanho da população” definido como “125” chega a, aproximadamente, 1h18min por execução do AG, de

modo que o teste como valores maiores para este parâmetro seria inviável por conta do tempo de execução.

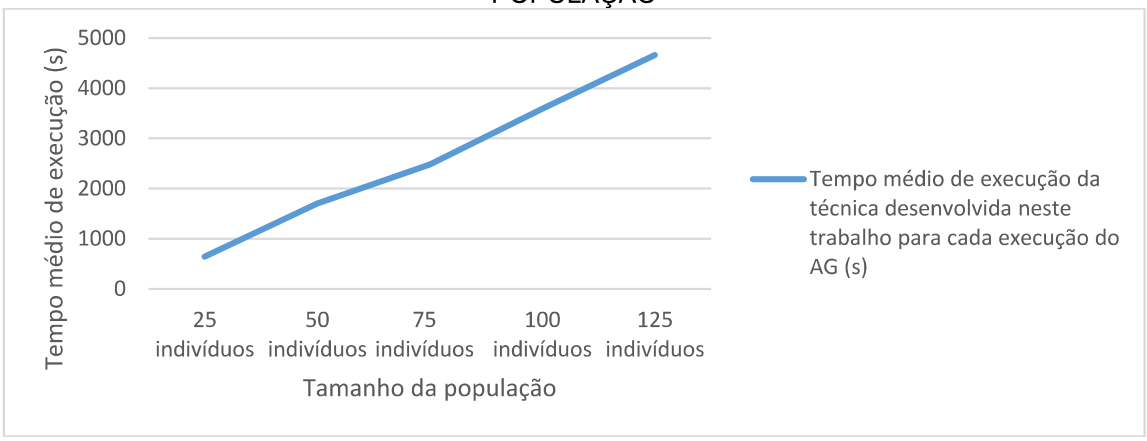
O melhor valor encontrado para o parâmetro tamanho da população é 125.

GRÁFICO 1 - AVALIAÇÃO DE VALORES PARA O PARÂMETRO TAMANHO DA POPULAÇÃO



FONTE – O Autor (2021).

GRÁFICO 2 - TEMPO DE EXECUÇÃO PARA VALORES DO PARÂMETRO TAMANHO DA POPULAÇÃO



FONTE: O autor (2021).

5.2.2 Condição de mutação

Para avaliação deste parâmetro, os demais parâmetros são fixados nos seguintes valores:

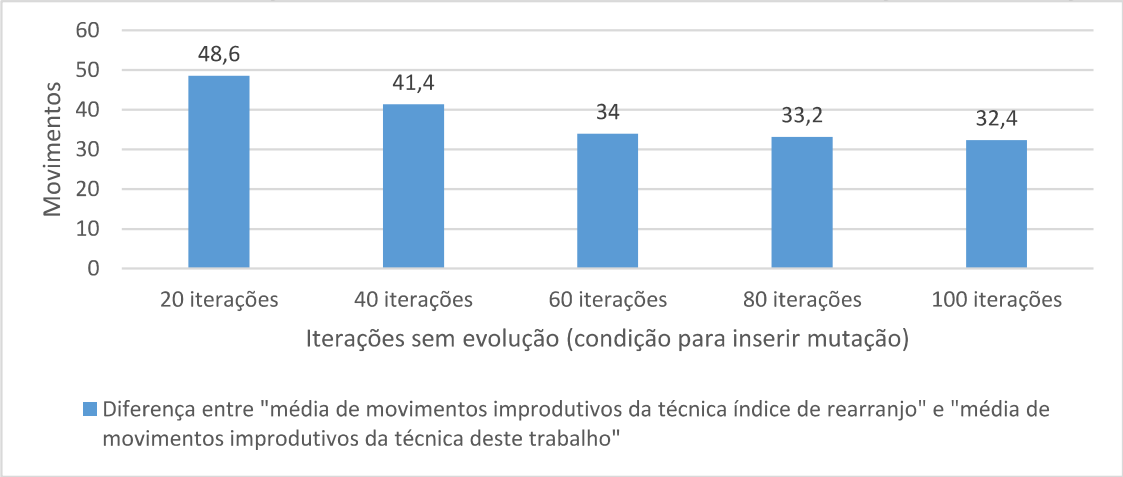
- Condição de parada em 120;
- Tamanho da população em 125;
- Percentual de mutação em 20;
- Grau de mutação em 40.

O GRÁFICO 3 mostra uma queda no desempenho, conforme o valor do parâmetro "condição de mutação" é aumentado. Isso indica que, quanto maior a demora para inserção de diversidade na população, mais rápido ela converge, ou seja, passará a ser formada por indivíduos quase idênticos. Consequentemente, as evoluções deixam de ocorrer e a condição de parada é atingida mais rapidamente, conforme mostrado no GRÁFICO 4.

Não foram realizados testes para valores deste parâmetro menores que 20 em decorrência do tempo de execução, o que tornaria os testes inviáveis.

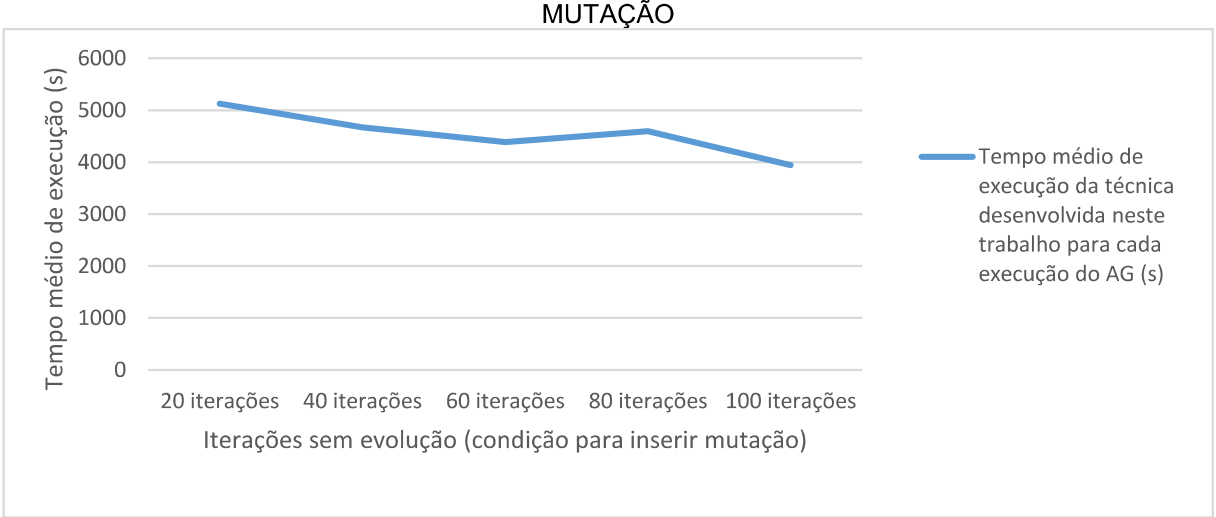
O melhor valor identificado para este parâmetro é 20.

GRÁFICO 3 - AVALIAÇÃO DE VALORES PARA O PARÂMETRO CONDIÇÃO DE MUTAÇÃO



FONTE: O autor (2021).

GRÁFICO 4 - TEMPO DE EXECUÇÃO PARA VALORES DO PARÂMETRO CONDIÇÃO DE MUTAÇÃO



FONTE: O autor (2021).

5.2.3 Percentual de mutação

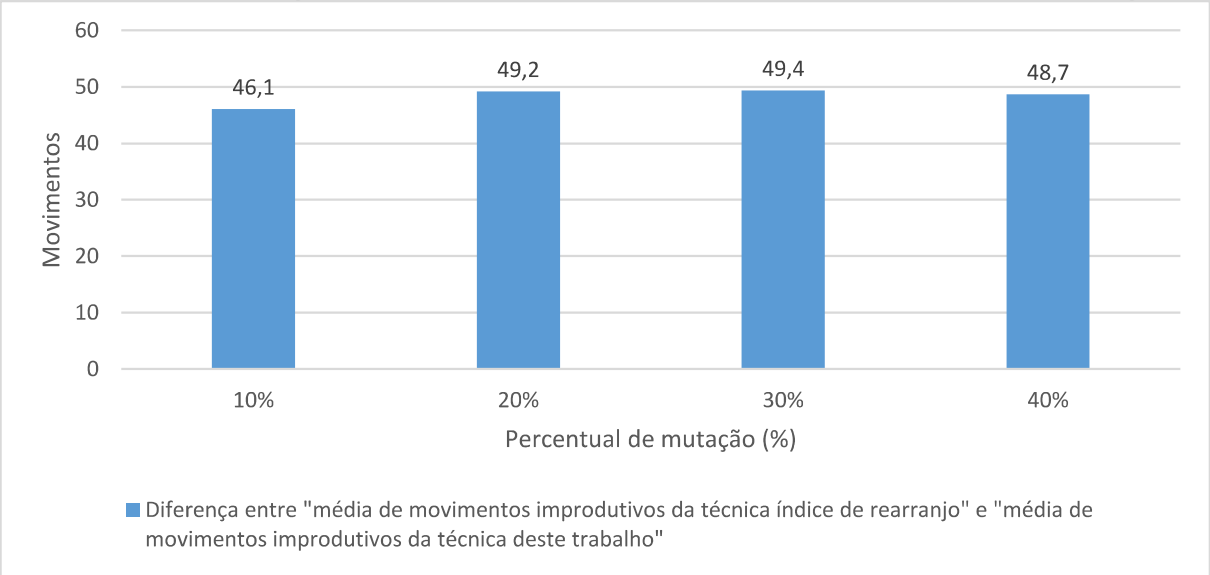
Para avaliação deste parâmetro, os demais parâmetros são fixados nos seguintes valores:

- Condição de parada em 120;
- Tamanho da população em 125;
- Condição de mutação em 20;
- Grau de mutação em 40.

O GRÁFICO 5 mostra uma estabilização entre 20% e 30% e uma pequena queda no desempenho em 40%. O GRÁFICO 6 mostra um aumento significativo no tempo de execução para cada aumento no valor do parâmetro “fator de mutação”. A explicação para isso é que a inserção de mutação em muitos indivíduos da população se assemelha a um *restart* desta, resultando em vários ciclos de *restarts* e evoluções até que a condição de parada seja atingida. Esses ciclos não representam uma melhora no desempenho.

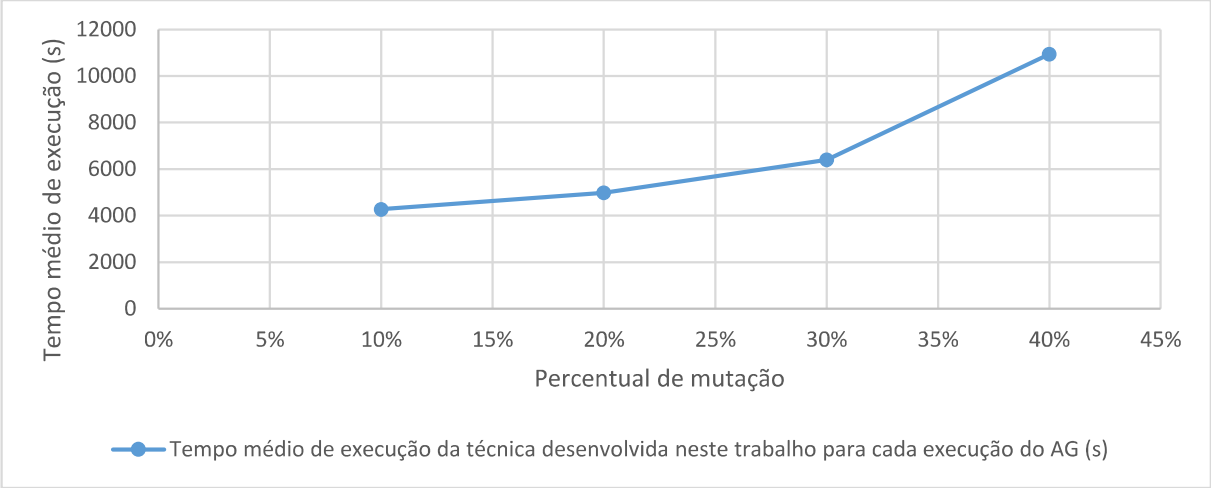
O melhor valor identificado para este parâmetro é 20%, levando em conta os desempenhos semelhantes e os tempos de execução, significativamente, diferentes.

GRÁFICO 5 - AVALIAÇÃO DE VALORES PARA O PARÂMETRO PERCENTUAL DE MUTAÇÃO



FONTE: O autor (2021).

GRÁFICO 6 - TEMPO DE EXECUÇÃO PARA VALORES DO PARÂMETRO PERCENTUAL DE MUTAÇÃO



FONTE: O autor (2021).

5.2.4 Condição de parada

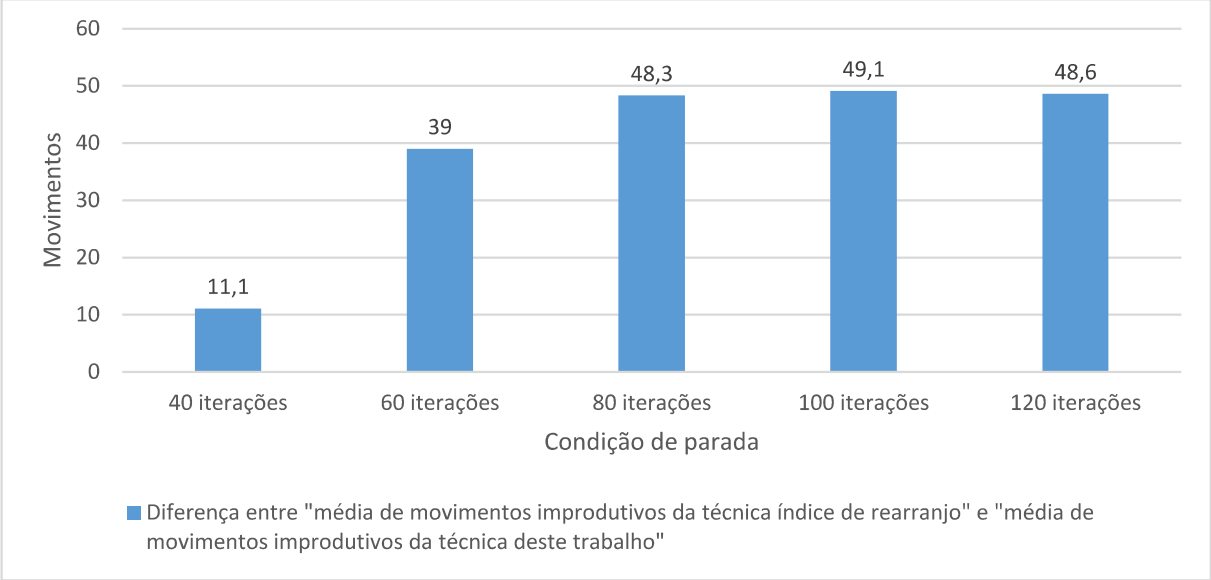
Para avaliação deste parâmetro, os demais parâmetros são fixados nos seguintes valores:

- Percentual de mutação em 20;
- Tamanho da população em 125;
- Condição de mutação em 20;
- Grau de mutação em 40.

O GRÁFICO 7 mostra uma estabilização no desempenho a partir de 80 iterações como condição de parada, para quantidades menores de iterações, o desempenho apresentado é, significativamente, menor. O que indica que, nessas condições, a execução é interrompida sem que a população tenha a chance de evoluir tanto quanto possível.

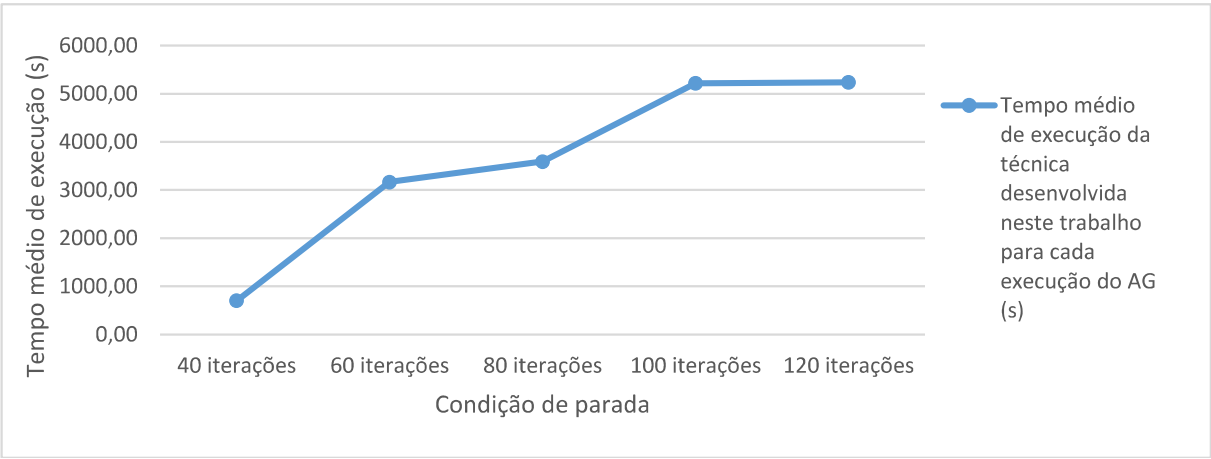
Levando em conta a estabilização no desempenho a partir de 80 iterações sem que haja a mesma estabilização no tempo de execução (GRÁFICO 8), considerou-se que 80 iterações é o melhor valor para o parâmetro condição de parada.

GRÁFICO 7 - AVALIAÇÃO DE VALORES PARA O PARÂMETRO CONDIÇÃO DE PARADA



FONTE: O autor (2021).

GRÁFICO 8 - TEMPO DE EXECUÇÃO PARA VALORES DO PARÂMETRO CONDIÇÃO DE PARADA



FONTE: O autor (2021).

5.2.5 Grau de mutação

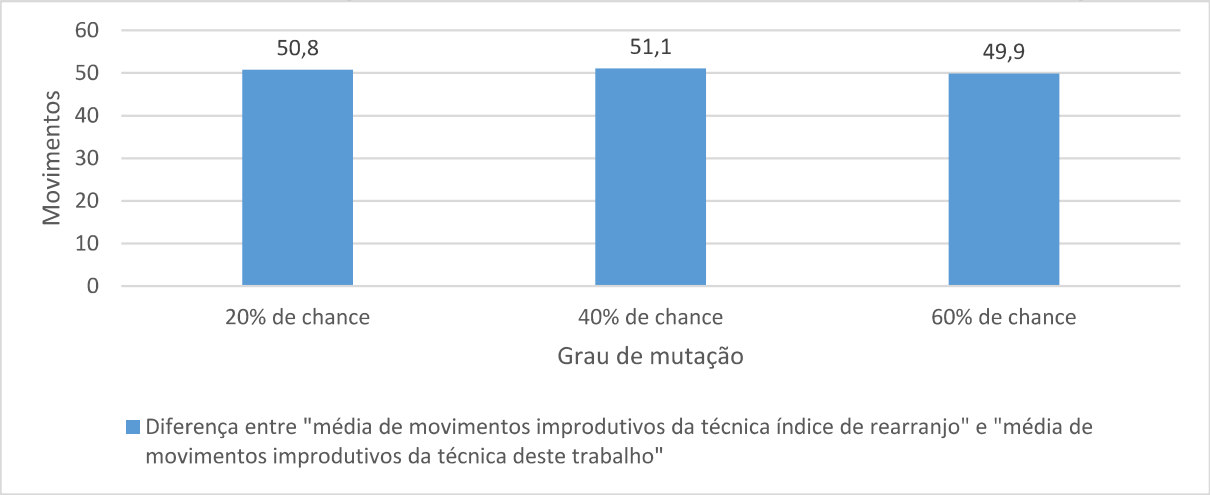
Para avaliação deste parâmetro, os demais parâmetros são fixados nos seguintes valores:

- Percentual de mutação em 20;
- Tamanho da população em 125;
- Condição de mutação em 20;
- Condição de parada em 80.

O GRÁFICO 9 demonstra que, para os valores avaliados para o parâmetro grau de mutação, não há diferença significativa no desempenho. O GRÁFICO 10, por sua vez, demonstra um aumento significativo no tempo de execução decorrente do aumento do valor do parâmetro avaliado, tornando inviável a realização de teste para valores deste parâmetro maiores que “60% de chance”. Isto indica que valores altos deste parâmetro causam o mesmo *restart* da população descrito na Seção 5.2.3.

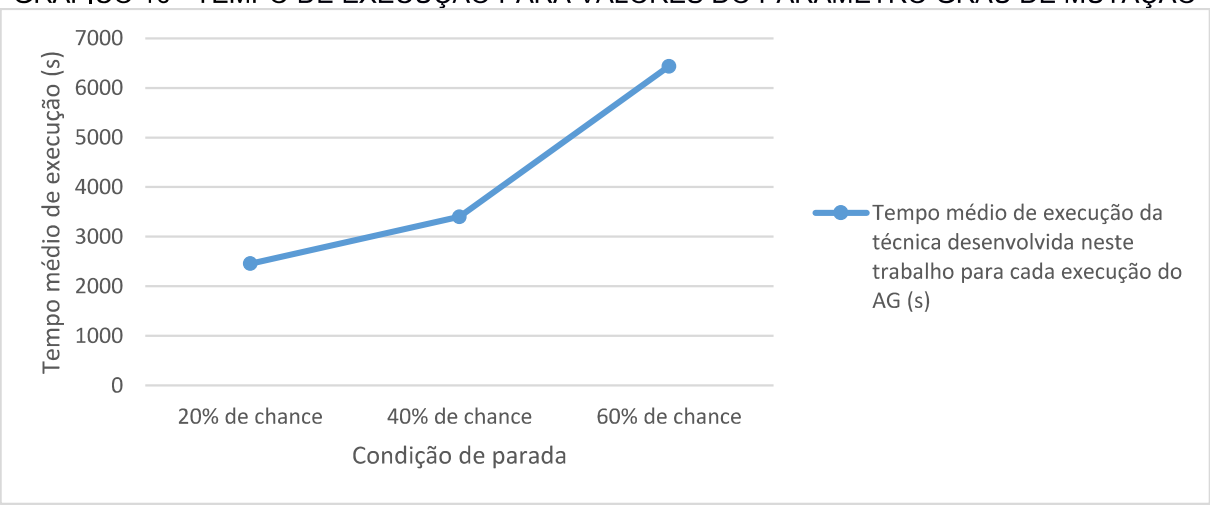
Levando-se em conta o tempo de execução menor, sem diferença significativa no desempenho, considerou-se que o melhor valor para o parâmetro grau de mutação é “20% de chance”.

GRÁFICO 9 - AVALIAÇÃO DE VALORES PARA O PARÂMETRO GRAU DE MUTAÇÃO



FONTE: O autor (2021).

GRÁFICO 10 - TEMPO DE EXECUÇÃO PARA VALORES DO PARÂMETRO GRAU DE MUTAÇÃO



FONTE: O autor (2021).

5.2 RESULTADOS

Os testes descritos nesta seção foram realizados com a seguinte configuração de parâmetros:

- Percentual de mutação em 20;
- Tamanho da população em 125;
- Condição de mutação em 20;
- Condição de parada em 80;
- Grau de mutação em 20.

Assim como em Carrara (2012), neste trabalho, serão realizados testes para os diferentes estados de armazenamento apresentados por Wan, Liu e Tsai (2009), que são: leve (20% da capacidade), médio (50% da capacidade) e pesado (80% da capacidade).

Para cada estado de armazenamento foram realizadas 50 execuções do AG. Em cada execução, é gerado, aleatoriamente, um pátio de tamanho 10x20 com uma quantidade predefinida de contêineres (conforme o estado de armazenamento). Para dada “sequência de saídas” (também gerada aleatoriamente) são aplicadas as técnicas “índice de rearranjo”, “pilha mais baixa” e a “técnica deste trabalho”. A TABELA 1 apresenta a quantidade média de movimentos improdutivos apresentada por cada técnica em cada estado de armazenamento.

A TABELA 2 apresenta o comparativo entre a técnica deste trabalho e a técnica índice de rearranjo, tanto em termos de diferença média da quantidade absoluta de movimentos improdutivos quanto em termos do percentual de redução dos momentos improdutivos ao se aplicar a técnica deste trabalho em vez da técnica índice de rearranjo. A TABELA 3 apresenta o mesmo comparativo, porém entre a técnica deste trabalho e a técnica de pilha mais baixa.

O GRÁFICO 11 apresenta o tempo médio de execução do algoritmo genético (AG) em cada estado de armazenamento. Os tempos de execução das demais técnicas não foram aferidos, porém constatou-se que são irrisórios e que a técnica desenvolvida neste trabalho apresentou tempo de execução, significativamente, maior. Porém, a redução da quantidade de movimentos improdutivos obtida com a

sua utilização (em vez das outras que foram testadas) compensa este fator negativo, visto que implica diminuição de custos financeiros e redução do tempo de operação do pátio, considerando o tempo que seria dispendido para realizar os movimentos improdutivos evitados, que, certamente, é maior que o tempo de execução do programa.

TABELA 1 - QUANTIDADE MÉDIA DE MOVIMENTOS IMPRODUTIVOS APRESENTADA POR CADA TÉCNICA EM CADA ESTADO DE ARMAZENAMENTO

Percentual de ocupação do pátio	**Técnica deste trabalho - Movimentos improdutivos**	**Índice de rearranjo - Movimentos improdutivos**	**Pilham mais baixa - Movimentos improdutivos**
20%	11,1	18,1	17,8
50%	17,6	39,1	51
80%	33,3	85,9	92,5

FONTE: O autor (2021).

TABELA 2 - COMPARATIVO ENTRE A TÉCNICA DESTE TRABALHO E A TÉCNICA ÍNDICE DE REARRANJO

Percentual de ocupação do pátio	**Diferença de movimentos improdutivos entre a técnica índice de rearranjo e a técnica deste trabalho**	**Percentual de redução de movimentos improdutivos comparado à técnica índice de rearranjo**
20%	7	38,67%
50%	21,5	54,99%
80%	52,6	61,23%

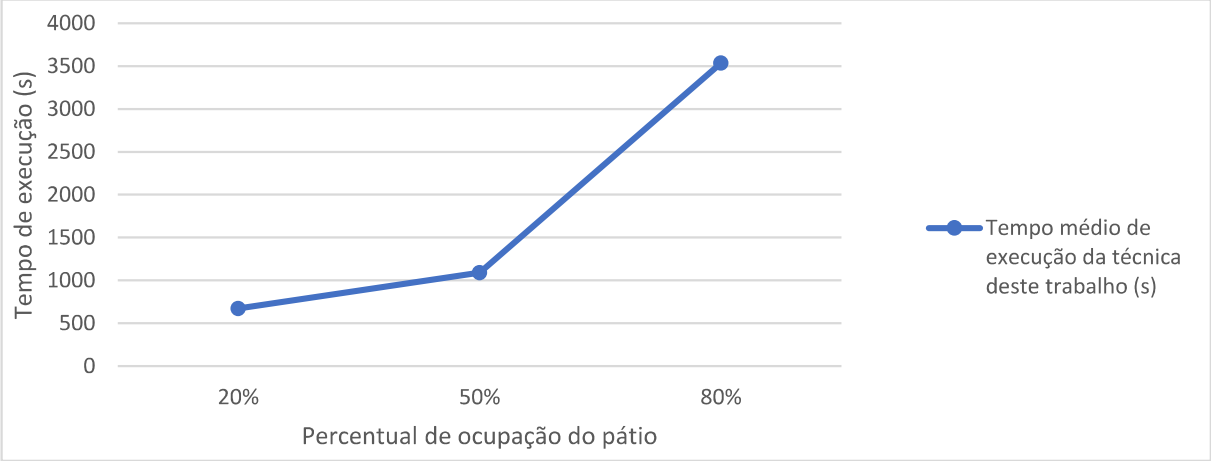
FONTE: O autor (2021).

TABELA 3 - COMPARATIVO ENTRE A TÉCNICA DESTE TRABALHO E A TÉCNICA PILHA MAIS BAIXO

Percentual de ocupação do pátio	**Diferença de movimentos improdutivos entre a técnica índice de rearranjo e a técnica deste trabalho**	**Percentual de redução de movimentos improdutivos comparado à técnica pilha mais baixa**
20%	6,7	37,64%
50%	33,4	65,49%
80%	59,2	64,00%

FONTE: O autor (2021).

GRÁFICO 11 - TEMPO MÉDIO DE EXECUÇÃO DA TÉCNICA DESTE TRABALHO EM CADA ESTADO DE ARMAZENAMENTO



FONTE: O autor (2021).

6 CONCLUSÃO

Os resultados apresentados na Seção 5.2 demonstraram que a técnica desenvolvida neste trabalho apresenta quantidade menor de movimentos improdutivos se comparada às soluções da literatura, principalmente, em pátios com graus de armazenamento pesados.

Para mensurar o benefício obtido, visto que não foi possível obter informações a respeito dos custos para movimentação de contêineres no Porto do Itaqui, pode-se tomar como referência os valores mostrados na FIGURA 16, extraídos da Tabela Geral de Preços da empresa Ecoporto Santos S.A., vigente a partir de abril de 2021 (ECOPORTO SANTOS, 2021). Os valores mostrados são cobrados por movimentação interna de contêineres (de um local para outro dentro do pátio), distinguindo-se entre a categoria FCL (sigla em inglês para contêiner totalmente carregado), que são cobrados por valor fixo, e a categoria LCL (sigla em inglês para “menos do que uma carga de contêiner”), no qual clientes, que não possuem, individualmente, a totalidade do conteúdo do contêiner, pagam valores proporcionais aos pesos de suas mercadorias.

Considerando esses custos, constata-se que as reduções dos números de movimentos improdutivos, mostradas em TABELA 2 e TABELA 3, obtidas ao se utilizar a técnica deste trabalho em vez das soluções da literatura testadas, é um benefício significativo para o Porto, no que se refere, principalmente, aos custos operacionais.

FIGURA 16 - VALORES COBRADOS POR MOVIMENTAÇÃO INTERNA DE CONTÊINERES PELA EMPRESA ECOPORTO SANTOS S.A.

Contêineres	
Contêineres FCL (por unidade)	R$ 686,41
Contêineres LCL (por ton/m³ ou fração)	R$ 67,80
Valor mínimo de cobrança Lote/LCL (por lote)	R$ 505,42

FONTE: Ecoporto Santos (2021).

Por outro lado, em situações reais, a logística de pátios de contêineres envolve entradas e saídas de contêineres ocorrendo simultaneamente e precisa lidar com mudanças repentinas nas sequências de entradas e saídas. Este dinamismo cria a necessidade de reavaliação constante, ou seja, em uma situação real, seria rotineira a execução da implementação desenvolvida para encontrar novas soluções, após mudança no cenário. Além disso, o pátio do contêiner do porto de Itaqui tem

capacidade de 1.341 TEUS (EMAP, 2019), significativamente maior que o de 10x20 utilizado nos testes. Portanto, o tempo de execução da solução desenvolvida neste trabalho seria um fator negativo em situações reais.

Todavia, o tempo de execução é um problema de implementação e não da técnica em si. Portanto, poderia ser resolvido ou atenuado através de uma nova implementação, que fosse pautada pela otimização dos recursos computacionais. A utilização de bibliotecas de algoritmos genéticos já existentes poderia contribuir com este propósito. Algumas bibliotecas deste tipo são: a GAlib, implementada em C++ (WALL, 1996), a ECJ - Evolutionary Computation Library, implementada em java (LUKE, 2010) e a brkgaAPI (TOSO; RESENDE, 2015), implementada em C++ e baseada em Algoritmos Genéticos de Chaves Aleatórias Viciadas, que contemplam a utilização de elitismo.

Em síntese, os resultados obtidos foram promissores e demonstram um ganho real na minimização de movimentos improdutivos para o Porto do Itaqui ao se utilizar a técnica desenvolvida neste trabalho em vez de outras soluções da literatura. Quanto à limitação decorrente do tempo de execução, constatou-se que é um problema isolado que pode ser resolvido ou atenuado por meio de mudanças na implementação.

REFERENCIAL BIBLIOGRÁFICO

AMARAL, R.; ALFREDINI, P. **Modelação Hidrossedimentológica no Canal de Acesso do Complexo Portuário do Maranhão**. Revista Brasileira de Recursos Hídricos v. 15, p. 5-14, 2010.

AMJAD M. at al. **Recent Research Trends in Genetic Algorithm Based Flexible Job Shop Scheduling Problems**. Hindawi - Mathematical Problems in Engineering, 2018.

AMT MARINE SOFTWARE. **SimpleStow**. AMT Marine Software, 2021. Disponível em: <https://www.amtmarine.ca/Pages/18>. Acesso em: 19 outubro, 2021.

ANTAQ, Agência Nacional de Transporte Aquaviário. **Boletim Informativo Aquaviário 3° Trimestre – 2018**. Disponível em: <https://www.gov.br/antaq/pt-br/assuntos/noticias/movimentacao-dos-portos-publicos-e-terminais-privados-atinge-295-milhoes-de-toneladas-no-terceiro-trimestre>. Acesso em: 12 abril. 2019.

ANTAQ, Agência Nacional de Transporte Aquaviário. **Desempenho do Setor Aquaviário 2018**. Disponível em: <http://portal.antaq.gov.br/wp-content/Uploads /2019/02/Anu%C3%A1rio-2018-Layout-4-3.pdf>. Acesso em: 12 abril. 2019.

ASM TREINAMENTOS. **Operador de Reach Stacker**. ASM Treinamentos, 2021. Disponível em: <https://asmtreinamentos.com.br/cursos2/operador-de-reach-stacker/>. Acesso em: 22 outubro, 2021.

AVRIEL, M.; PENN, M.; SHPIRER, N. **Container ship stowage problem: complexity and connection to the coloring of circle graphs**. Discrete Applied Mathematics v. 103, p. 271-279, 2000.

BORGMAN, B.; ASPEREN, E.; DEKKER, R. **Online rules for container stacking**. Operational Research Spectrum v. 32, p. 687-716, 2010.

CARRARO,L. **Algoritmo de seleção clonal para a minimização de rearranjos em operações de pilhas de contêineres**. 2012. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Prebiteriana Mackenzie, São Paulo - SP.

ECOPORTO SANTOS. **Tabela de Preços**. Ecoporto Santos, 2021. Disponível em:<https://ecostorageapp002.blob.core.windows.net/content/Ecoporto/Servicos/Pdf/Tabela-de-Pre%C3%A7os.pdf?V=2017021552544>. Acesso em: 19 outubro, 2021.

EMAP, Empresa Maranhense de Administração Portuária. **Porto do Itaqui pronto para retomada da linha regular de contêineres**. Disponível em: <http://www.portodoitaqui.ma.gov.br/imprensa/noticia/porto-do-itaqui-pronto-para-retomada-da-linha-regular-de-conteineres>. Acesso em: 13 abril. 2019.

ENGEPLUS. **Conheça as etapas da importação dentro do Porto**. Engeplus, 2021, Disponível em: <http://www.engeplus.com.br/noticia/economia/2018/conheca-as-etapas-da-importacao-dentro-do-porto>. Acesso em: 22 outubro, 2021.

GAD, A. **Introduction to Optimization with Genetic Algorithm**. Disponível em: <https://towardsdatascience.com/introduction-to-optimization-with-genetic-algorithm-2f5001d9964b>. Acesso em: 11 maio. 2020.

HAUPT, R.; HAUPT, S. **Practical Gernetic Algorithms**. 2. ed. Nova Jersey: Wiley-Interscience, 2004.

JUNQUEIRA, C. **Modelagem e resolução do problema de otimização conjunta do plano de estiva e movimentação de contêineres em pátios portuários**. 2015. Dissertação (Mestrado em Pesquisa Operacional) - Universidade Estadual de Campinas, Limeira - SP.

LEE, D.; WANG, H.; MIAO, L. **Quay crane scheduling with non-interference constraints in port container terminals**. Transportation Research Part E: Logistics and Transportation Review v. 44, p. 124-135, 2008.

LINDEN, R. **Algoritmos Genéticos: Teoria e Implementação**. 3ª Edição. Editora Ciência Moderna, 1 janeiro 2012.

LUKE, S. **The ECJ Owner's Manual: A User Manual for the ECJ Evolutionary Computation Library**. Department of Computer Science - George Mason University, 2010. Disponível em: <https://docs.google.com/document/d/11fAmHmHxChEI9dh-iXMJLHBIlgMhE6p_MsGn-X-qstI/edit>. Acesso em: 19 outubro. 2021.

MAGICLOGIC. **Cube-IQ: Advanced Load Planning Software on your Desktop.** MAGICLOGIC, 2021. Disponível em: <https://magiclogic.com/cubeiq/>. Acesso em: 19 outubro, 2021.

MURTY, K. G.; LIU, JIYIN; WAN, Y.-WAH; LINN, RICHARD. **A decision support system for operations in a container terminal**. Decision Support Systems, v. 39, p. 309-332, 2005.

MURTY, K.; LIU, J.; WAN, Y.; LINN, R. **A decision support system for operations in a container terminal**. Decision Support Systems v. 39, p. 309-332, 2005.

NAUTIC EXPO. **Empilhadeira reach-stacker de tração traseira F500.RS series**. Nautic Expo, 2021. Disponível em: <https://www.nauticexpo.com/pt/prod/cvs-ferrari/product-30627-387117.html>. Acesso em: 22 outubro, 2021.

NETO, M.; SANTOS, C.; PRADO, A.; LIMA, J. **Equipamentos portuários de movimentação de contêineres: portêiner e guindaste móvel sobre pneus**. FATEC Guaratinguetá, 2021. Disponível em: <http://www.fatecguaratingueta.edu.br/fateclog/artigos/Artigo_129.PDF>. Acesso em: 19 outubro. 2021.

PAIVA, R. **Zonas de Influência Portuária (Hinterlands) e um Estudo de Caso em um Terminal de Contêineres com a Utilização de Sistemas de Informação Geográfica**. 2006. Dissertação (Mestrado em Engenharia Industrial) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro - RJ.

PIVA, M. **Aplicação de Algoritmo Genético no Planejamento de Embarque em Terminais de Contêineres**. 2008. Dissertação (Mestrado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia.

REYNOSO, R. **Comparação de Modelos de Otimização para Armazenamento de Contêineres nos Pátios Portuários**. 2017. Dissertação (Mestrado em Pesquisa Operacional) - Universidade Estadual de Campinas, Faculdade de Ciências Aplicadas, Limeira - SP.

SEA NEWS. **Global Container Services deploys 2nd rail-mounted gantry near Moscow**. Sea News, 2021. Disponível em: <https://www.seanews.com.tr/global-container-services-deploys-2nd-rail-mounted-gantry-near-moscow/94885/>. Acesso em: 22 outubro, 2021.

STEENKEN, D.; VOSS, S.; STAHLBOCK, R. **Container terminal operation and operations research – a classification and literature review**. Operations Research-Spektrum v. 26, p. 3-49, 2004.

TOSO, R.; RESENDE, M. **A c++ application programming interface for biased random-key genetic algorithms**. Optimization Methods and Software v. 30, p. 81-93, 2015.

WALL, M. **GAlib: A C++ Library of Genetic Algorithm Components**. Mechanical Engineering Department - Massachusetts Institute of Technology, 1996. Disponível em: <http://mirror.krakadikt.com/2004-11-13-genetic-algorithms/lancet.mit.edu/ga/dist/galibdoc.pdf>. Acesso em: 19 outubro. 2021.

WAN, Y.; LIU, J.; TSAI, P. C. **The assignment of storage locations to containers for a container stack**. Naval Research Logistics (NRL) v. 56, p. 699–713, 2009.