INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE
IFFLUMINENSE *CAMPUS* CAMPOS CENTRO
PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

MAICO PIASSAROLI TRANCOSO

APLICAÇÃO DE UM MODELO PARA OTIMIZAÇÃO DE FLUXO RODOVIÁRIO: UM ESTUDO DE CASO EM UM TERMINAL PORTUÁRIO

Campos dos Goytacazes/RJ
2021

2021	MAICO PIASSAROLI TRANCOSO	MPSAEG/IFF

MAICO PIASSAROLI TRANCOSO

APLICAÇÃO DE UM MODELO PARA OTIMIZAÇÃO DE FLUXO RODOVIÁRIO: UM ESTUDO DE CASO EM UM TERMINAL PORTUÁRIO

Dissertação apresentada ao Programa de Pós-Graduação em Sistemas Aplicados à Engenharia e Gestão do Instituto Federal de Educação, Ciência e Tecnologia Fluminense *campus* Campos Centro como requisito parcial para a conclusão do Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão (SAEG).

Orientador: Prof. Dr. Henrique Rego Monteiro da Hora.

Co-orientador: Prof. Dr. Valdecy Pereira

Campos dos Goytacazes, RJ
2021

Biblioteca Anton Dakitsch

CIP - Catalogação na Publicação

T772a Trancoso, Maico Piassaroli

APLICAÇÃO DE UM MODELO PARA OTIMIZAÇÃO DE FLUXO RODOVIÁRIO: UM ESTUDO DE CASO EM UM TERMINAL PORTUÁRIO / Maico Piassaroli Trancoso - 2021.

85 f.

Orientador: Henrique Rego Monteiro da Hora

Coorientador: Valdecy Pereira

Dissertação (mestrado) -- Instituto Federal de Educação, Ciência e Tecnologia Fluminense, Campus Campos Centro, Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão, Campos dos Goytacazes, RJ, 2021.

Referências: f. 74 a 75.

1. Otimização. 2. Pesquisa Operacional. 3. Apoio à Decisão. 4. Porto. 5. Job Shop Scheduling. I. da Hora, Henrique Rego Monteiro, orient. II. Pereira, Valdecy, coorient. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da Biblioteca Anton Dakitsch do IFF com os dados fornecidos pelo(a) autor(a).

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

Maico Piassaroli Trancoso

APLICAÇÃO DE UM MODELO PARA OTIMIZAÇÃO DE FLUXO RODOVIÁRIO: UM ESTUDO DE CASO EM UM TERMINAL PORTUÁRIO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação do Instituto Federal de Educação, Ciência e Tecnologia Fluminense, no Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão (MPSAEG), como parte dos requisitos necessários à obtenção do título de Mestre em Sistemas Aplicados Engenharia e Gestão.

Aprovado(a) em 23 de dezembro de 2021.

Banca Examinadora:

Henrique Rego Monteiro da Hora:01857552750
Assinado de forma digital por Henrique Rego Monteiro da Hora:01857552750
Dados: 2022.12.08 10:13:49 -03'00'

Prof. Dr. Henrique Rego Monteiro da Hora, Título
Instituto Federal de Educação, Ciência e Tecnologia Fluminense - IFF
(Orientador)

VALDECY PEREIRA:08983644770
Digitally signed by VALDECY PEREIRA:08983644770
DN: c=BR, o=ICP-Brasil, ou=videoconferencia, ou=33683111000107, ou=Secretaria da Receita Federal do Brasil - RFB, ou=ARSERPRO, ou=RFB e-CPF A1, cn=VALDECY PEREIRA:08983644770
Date: 2022.12.15 09:44:29 -03'00'

Prof. Dr. Valdecy Pereira
Universidade Federal Fluminense - UFF
(Coorientador)

Frederico Galaxe Paes
Assinado de forma digital por Frederico Galaxe Paes
Dados: 2022.12.17 11:13:48 -03'00'

Prof. Dr. Frederico Galaxe Paes
Instituto Federal de Educação, Ciência e Tecnologia Fluminense - IFF

Rafael Brandao de Rezende Borges:09434113784
Assinado de forma digital por Rafael Brandao de Rezende Borges:09434113784
Dados: 2022.12.19 19:12:25 -03'00'

Prof. Dr. Rafael Brandão de Rezende Borges
Universidade do Estado do Rio de Janeiro - UERJ

Dedico este trabalho à minha mãe, Neide Piassaroli, por estar presente em todos os momentos da minha vida, e à minha companheira, Luiza Pin, pelo apoio, incentivo e compreensão nos momentos difíceis.

AGRADECIMENTOS

Agradeço à minha mãe, por me apoiar nas minhas decisões, pela paciência, amor e por ser meu exemplo de força, inspiração e determinação.

À minha família, minha companheira e meus amigos, pelos incentivos, pelo compartilhamento de alegrias e pela compreensão das ausências.

Ao meu orientador, professor Henrique Rego Monteiro da Hora, pelos ensinamentos, pela orientação e pela compreensão nos momentos difíceis da pesquisa.

À empresa estudada, pelo tempo dependido nas entrevistas e fornecimento dos dados para o trabalho.

Agredeço também a todos aqueles que indiretamente também contribuíram para que esse trabalho fosse concluído.

RESUMO

O desenvolvimento da logística empresarial torna-se necessário a viabilização do alto nível do comércio mundial, sendo ela a essência do comércio e contribuindo na melhoria do padrão econômico da vida em geral. Um dos setores de grande papel e contribuição em toda a cadeia de suprimentos é o setor portuário e os operadores logísticos relacionados a este ramo. Dessa forma, atualmente o tempo de entrega dos serviços e a busca por eficiência da utilização de ativos é um desafio para o setor. Neste estudo o objetivo é avaliar um modelo de otimização para aumento da produtividade da expedição rodoviária de um terminal portuário de granéis líquidos através da função de sequenciamento e alocação de veículos (*Job Shop Scheduling*) nas estações de carga. Para isso, é elaborado um modelo de otimização baseado em algoritmo genético e experimentado a partir dos registros reais dos carregamentos do terminal portuário estudado, após serem extraídos e organizados, fornecendo as informações específicas para o modelo. Como resultado, o modelo retornou resultados acima das expectativas, onde se mostrou muito eficiente para baixas quantidades de veículos em fila. Ainda assim, a redução da variação do *makespan* operacional pdfara qualquer quantidade de veículos em fila também foi um resultado positivo do modelo.

Palavras-chave: Otimização, Pesquisa Operacional, Apoio à Decisão, Porto, *Job Shop Scheduling*, Algoritmo Genético

ABSTRACT

The development of business logistics becomes necessary for the viability of the current high level of world trade, being is the essence of trade and a great contribution to improving the economic standard of life in general. One of the sectors with a great role and contribution in the entire supply chain is the port sector and the logistics operators related to this branch. Currently, the delivery time of services and the search for efficiency in the use of assets is a challenge for the sector. In this study, the objective is to evaluate a proposed optimization model to increase the productivity of road dispatch in a liquid bulk maritime terminal by solving a Job Shop Scheduling problem type in the truck loadings. For this, an optimization model based on a genetic algorithm is designed and tested using the real records of the terminal loads, after they were extracted and cleaned, as an input, providing specific information for the model. As a result, the model returned results that exceeded expectations, which proved to be very efficient for low numbers of queued vehicles. Still, the reduction of the operational makespan variation for any number of vehicles was also a positive result of the model.

Keywords: Optimization; Operational Research; Decision Support Systems; Port, Genetic Algorithm

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CLP – Computador Lógico Programável

CNT – Confederação Nacional de Transporte

COMEX – Comércio Exterior

FOB – significa frete por conta do embarcador, vem da língua inglesa “*free on board”*.

MDIC – Ministério do Desenvolvimento, Indústria, Comércio Exterior e Serviços

ANTAQ – Agência Nacional de Transportes Aquaviários

TUP – Terminal de Uso Privado

SUMÁRIO

1 INTRODUÇÃO

1.1 JUSTIFICATIVA

A busca por modelos logísticos viáveis a determinados setores tem sido o foco de estudos e pesquisas em nível mundial. Entretanto, devido a uma série de condições envolvendo clima, cultura, restrições operacionais e legislação, cada país tende a adaptar modelos que atendam sua peculiaridade, para que estes se estabeleçam economicamente vantajosos, devido a importância que o comércio exerce na economia de um país. Nesse prisma, um dos setores de grande representação e contribuição em toda a cadeia de suprimentos é o transporte marítimo.

Este trabalho aborda um modelo prático de melhoria para um problema de sequenciamento de atividades portuárias em um terminal de granéis líquidos a fim de se obter o entendimento sobre a melhoria da produtividade deste setor.

Não obstante, tem-se a contribuição desta pesquisa para o auxílio no entendimento de modo geral de ferramentas aplicadas à decisão e na gestão de terminais portuários no sentido de se obter melhores índices de performance e aproveitamento de ativos, além de realizar o aperfeiçoamento de métodos inovadores e tecnológicos para o setor.

1.2 RELEVÂNCIA

Desde a chegada dos europeus, em 1500, o processo de colonização do Brasil só foi justificado a partir do desenvolvimento de atividades econômicas lucrativas com o cultivo permanente do solo. Inicialmente, esta atividade foi o cultivo de cana-de-açúcar, produto que era comercializado nos mercados europeus, sendo o transporte marítimo o principal viabilizador deste comércio (IBGE, 2011).

Portanto, ao longo de toda a história do Brasil, por estar localizado em numa posição privilegiada e com uma extensão litorânea de aproximadamente 11.000 km (IBGE, 2018), o transporte marítimo exerce um papel fundamental no desenvolvimento econômico do país, tornando-se imperativo os investimentos para o

desenvolvimento eficiente de toda logística aquaviária, possibilitando processos mais eficientes em toda cadeia portuária.

A vantagem do transporte aquaviário é a capacidade de transportar grandes quantidades de cargas por longas distâncias, gerando um menor custo operacional por unidade de carga (CNT, 2019).

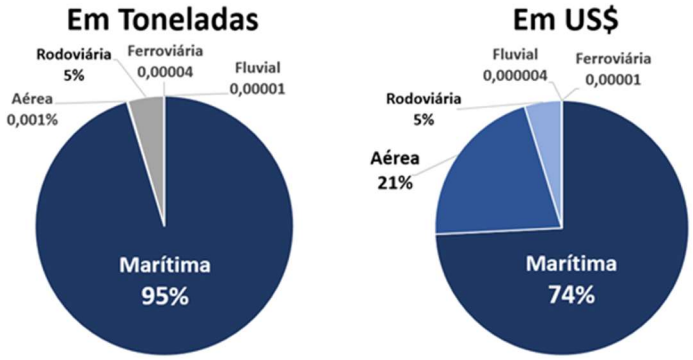
Em relação às características geográficas, o Brasil possui 19,5 mil quilômetros de hidrovias economicamente navegáveis e cerca de 58% da população está concentrada em uma faixa de até 200km do litoral (CNT, 2019).

Dentro do contexto brasileiro, são cinco os modais de transporte de cargas: rodoviário, ferroviário, aquaviário, dutoviário e aéreo. Cada um possui características especificas em sua operacionalização e custos, de maneira a serem mais adequados a cada tipo de carga. Tais modais diferem em relação a velocidade, a quantidade de carga transportada, riscos de acidentes e avarias, impactos ambientais, etc.. O investimento no setor de forma continuada é particularmente importante para um país de dimensões continentais onde a movimentação de mercadorias tem que percorrer grandes distâncias (CNT, 2019).

No modal aquaviário, segundo a ANTAQ – Agência Nacional de Transportes Aquaviários, o Brasil possui 37 portos organizados e 144 terminais de uso privado (TUP). Sendo que em 2020 foram movimentados 1,15 bilhão de toneladas de mercadorias, acarretando um aumento de mais de 4,2% em relação a 2019, mesmo considerando-se que este foi um período de pandemia por Covid-19, no qual enfraqueceram-se as atividades econômicas mundiais.

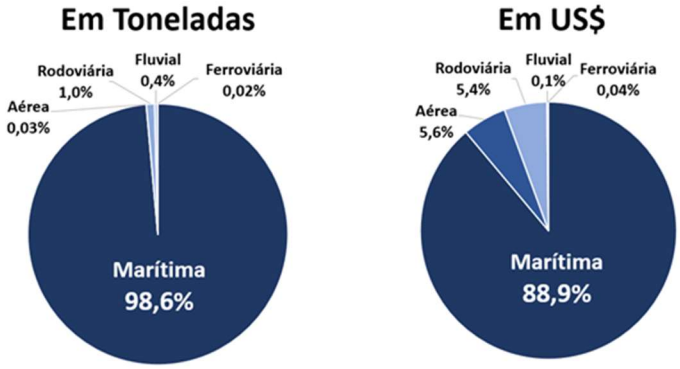
Na Figura 1 e Figura 2 são exibidas a participação em toneladas e em dólares dos modais logísticos nas importações e exportações brasileiras, evidenciando a importância do meio aquaviário para o comex brasileiro.

Figura 1 – *Share* de importação de acordo com a via utilizada



Fonte: ANTAQ (2021)

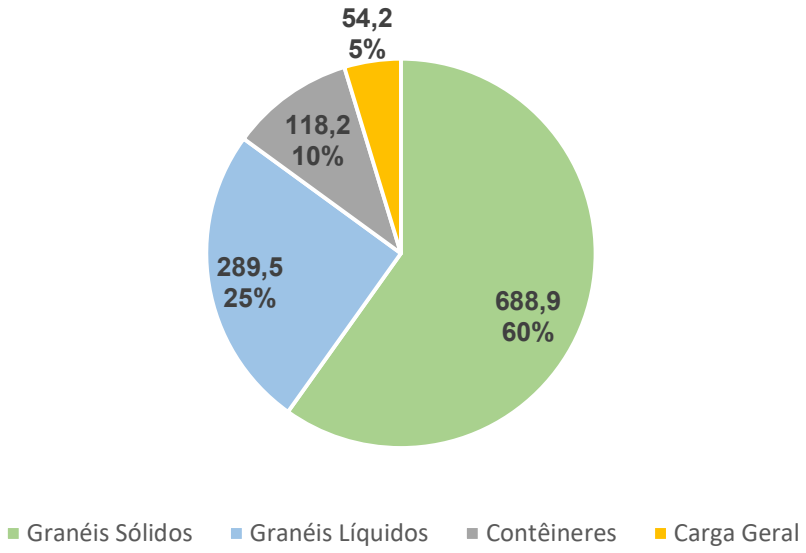
Figura 2 – *Share* de exportação de acordo com a via utilizada



Fonte: ANTAQ (2021)

Na Figura 3 é mostrado o percentual de participação de toda a movimentação brasileira para os quatro principais tipos de mercadorias.

Figura 3 – Perfil de cargas nos portos brasileiros (em milhões de toneladas)



Fonte: ANTAQ (2021)

Sendo assim, é importante observar que, a performance das operações portuárias está intrinsecamente ligada ao crescimento daquela instalação, por se tratar de uma atividade logística que em sua grande maioria movimenta produtos de commodities cuja escala de movimentação é um fator importante.

Complementando as informações expostas acima, ainda de acordo com o Anuário Estatístico Aquaviário da ANTAQ (2021), os principais indicadores portuários são definidos por:

1) Receita Tarifária Média;

2) Consignação de Navios;

3) Produtividade;

4) Tempos Operacionais;

5) Séries Históricas.

1.3 MOTIVAÇÃO

É notável a necessidade de aperfeiçoamento do setor e da implementação de formas inovadoras na contribuição desse objetivo, uma vez que o Brasil atua

ativamente na economia mundial, sendo o setor portuário uma peça fundamental para as importações, exportações e cabotagem do país.

Em 2020, foi apresentado um projeto de lei (PL nº 4199/2020) do governo federal denominado como BR do Mar, que tem como objetivo o estímulo do crescimento do transporte por cabotagem no Brasil, o qual é um modo de transporte que tem apresentado crescimento nos últimos tempos (Ministério da infraestrutura, 2020). Com isso, o programa pretende aumentar a oferta da cabotagem no Brasil, incentivar a concorrência e reduzir custos.

Ainda de acordo com o Ministério da infraestrutura (2020), considerando o potencial de crescimento do setor e os gargalos atualmente observados, principalmente por este programa estar diretamente relacionado ao crescimento do setor rodoviário atuante nos portos, a importância deste trabalho está alinhada com os desafios do setor no que se refere à otimização das operações.

1.4 OBJETIVOS

O objetivo geral deste trabalho é avaliar um modelo de otimização para aumento da produtividade da expedição rodoviária de um terminal portuário de granéis líquidos através da função de sequenciamento e alocação de veículos nas estações de carga.

Os objetivos específicos são:

a) Realizar o detalhamento e entendimento do problema;

b) Parametrizar os indicadores de produtividade do terminal;

c) Elaborar um modelo computacional capaz de realizar a otimização do fluxo rodoviário do terminal;

d) Avaliar a capacidade de melhoria dos indicadores de desempenho do terminal com a utilização do modelo de otimização das operações rodoviárias.

2 REVISÃO BIBLIOGRÁFICA

2.1 BASE CONCEITUAL

2.1.1 Terminal Portuário

Primeiramente, é preciso estabelecer aqui a definição de porto, o qual é uma área abrigada à beira de oceano, trecho de mar ou rio destinadas ao recebimento e atracação de embarcações. Além disso, os portos precisam ser compatíveis com os requisitos estruturais de terra, ou seja, deve dispor de instalações propícias ao embarque e desembarque de mercadorias e pessoas (COSTA, 2021).

Cada área portuária é dividida, conforme suas especificidades, em terminais portuários, onde cada um tem por objetivo uma classificação específica em relação ao seu tipo de carga, podendo ser carga sólida, carga líquida, multipropósito, etc..

Em relação à infraestrutura dos terminais, estes possuem estações de transporte e desempenham as seguintes funções: manuseio de carga, armazenamento de carga, interface entre modos de transportes, despacho e consolidação de carga (HANDABAKA, 1994).

Nas figuras 4 e 5 é possível verificar como uma forma exemplificativa dois tipos de terminais, sendo um terminal de container e outro de granéis líquidos, onde percebe-se o padrão da infraestrura em relação à transferência de mercadoria do navio, estocagem em um pátio ou em tanques de armazenamento e posteriormente a transferência para um outro tipo de modal, comumente sendo o rodoviário.

Figura 4 - Terminal portuário de contêineres

Fonte: www.portodesantos.com.br. acesso em: set/2021.

Figura 5 - Terminal portuário de Granéis líquidos

Fonte: www.oiltanking.com. acesso em: set/2021.

Via de regra, em sua grande maioria, a operação em um terminal portuário segue a mesmo fluxo, com o desembarque/embarque da mercadoria de navios, armazenamento por um curto período de tempo e então expedição/recebimento por modal rodoviário.

Na figura 6 é possível notar esses três elementos em um terminal de granéis líquidos.

Figura 6 – Transferência de mercadoria de granéis líquidos

Fonte: www.storageterminalsmag.com. acesso em: set/2021.

Vale comentar que toda essa transferência de mercadorias faz parte de uma cadeia logística complexa e extremamente importante, onde, em um panorama de comércio exterior, sistemas logísticos eficazes estabelecem empresas em um padrão de alto nível mundial (BALLOU, 2007a).

2.1.2 Job Shop Scheduling

Antes de explicar os problemas do tipo *Job Shop Scheduling,* é necessário entender o contexto no qual estão inseridos. Problemas desse tipo fazem parte do estudo da pesquisa operacional (PO), que envolve a coordenação das atividades operacionais das organizações. Por isso tem sido amplamente aplicada em áreas como a manufatura, transportes, planejamento financeiro, entre outros (HILLIER; LIEBERMAN, 2006).

Para encontrar a solução desses e outros problemas a pesquisa operacional utiliza de forma geral utiliza a programação linear como ferramenta, comumente envolvendo um problema genérico de alocação, da melhor forma possível, os recursos limitados para atividades que competem entre si (HILLIER; LIEBERMAN, 2006).

Uma das características da PO é que esta busca frequentemente encontrar uma melhor solução ao invés de a melhor solução, considerando que pode existem

várias soluções consideradas como melhores. Além disso, o estudo de problemas de PO envolve, em grande parte, formulações de um modelo genérico para o problema e aplicações práticas, e não apenas resolução de exercícios matemáticos (HILLIER; LIEBERMAN, 2006)

Por isso, problemas de sequenciamento fazem parte do grupo de problemas de PO, onde tarefas são formados por sistemas decisórios, no qual estes desempenham um importante papel nas indústrias de manufatura e de serviços. Estes métodos dependem de técnicas matemáticas e modelos heurísticos, no qual objetivam a alocação de recursos em atividades que devem ser executadas, onde a alocação destes recursos deve ser feita de uma forma que as companhias maximizem seus objetivos (PINEDO, 2009).

Job Shop Scheduling ou de uma forma abrangente, *scheduling,* é ato de alocar recursos finitos em um determinado período objetivando a conclusão de um conjunto de tarefas. Onde cada tarefa é dividida em operações ou atividades que são processadas por uma determinada sequência de recursos. Ou seja, em resumo pode-se dizer em termos científicos que, o objetivo é encontrar um sequênciamento para as atividades que satisfazem os objetivos da melhor maneira possível (KUHPFAHL, 2016).

Algumas terminologias são importantes para a elaboração e condução desses problemas, como exemplo:

- Tempo de processamento: É o tempo demandado para a conclusão de cada operação ou atividade.
- Tempo de prontidão: É o tempo em que uma tarefa está disponível para processamento.
- Data prevista: É a data prevista para conclusão de uma tarefa.
- Tempo de início: É o tempo em que se inicia a execução de uma tarefa ou o tempo de início da primeira atividade em uma máquina.
- Tempo de conclusão: É o tempo de término de uma tarefa ou da conclusão da última atividade em uma máquina.

Em termos práticos, problemas de sequencimaneto consideram *n* tarefas e estas devem visitar *m* máquinas seguindo uma sequência pré-determinada em diferentes

rotas umas das outras, sendo assim chamados de ***Job Shop models***. Há também certas variações, como no caso onde todas as tarefas têm a mesma rota ou sequência de máquinas, chamado de ***Flow Shop Models***. Além disso, podem existir modelos em que existam várias máquinas idênticas, assim as tarefas podem ser executadas em várias máquinas para uma mesma atividade, sendo assim chamadas de ***Parallel Machine Models*** (PINEDO, 2009).

Como exemplo, em um modelo geral de *Job Shop Scheduling* existem várias tarefas para serem processadas em várias máquinas diferentes, sendo relevante ou não a ordem de processamento de cada atividade, gerando assim tempos de processamentos para cada sequência de tarefas. O ponto crítico, portanto, ocorre na competição entre atividades pelo uso de uma mesma máquina, como visto na tabela 1.

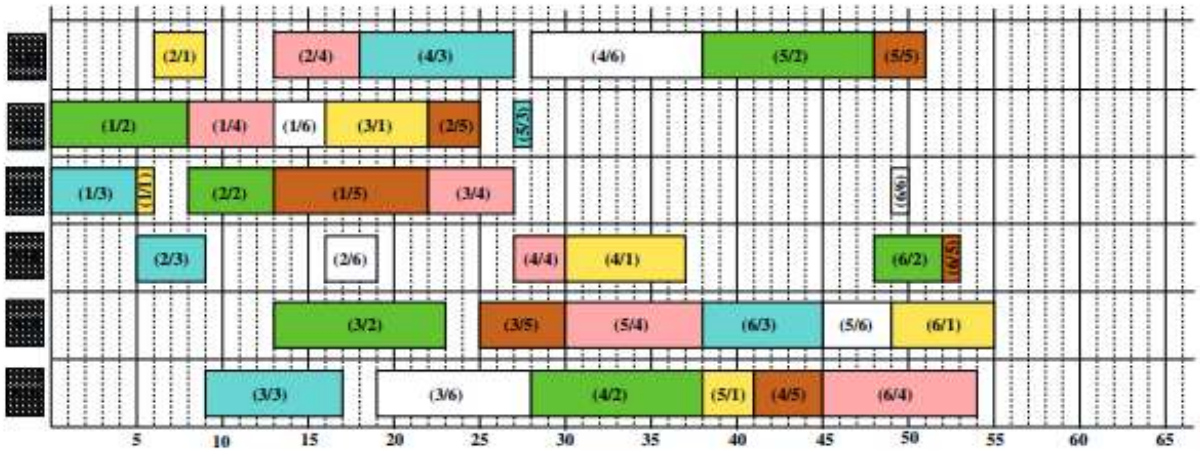
Tabela 1 – Exemplo de problema de sequenciamento de atividades

Atividades	**Sequência de máquinas**	**Tempos de processamento**
1	1, 2, 3	P_{11} = 10, P_{21} = 5, P_{31} = 7
2	3, 2, 1, 4	P_{32} = 7, P_{22} = 5, P_{12} = 10, P_{42} = 8
3	2, 3, 4	P_{23} = 4, P_{33} = 6, P_{43} = 11
...	...	...

Fonte: Elaborado pelo próprio autor (2021)

Ao fim, são vários os tipos de objetivos ou função objetivo para serem atendidas, como exemplo a maximização do total de tarefas atendidas em um período, a minimização do tempo total de atraso ou ainda a minimização do tempo da conclusão de um conjunto de tarefas (*makespan*). Na figura 7 é exibido como exemplo um gráfico de Gantt para objetivo de *makespan*.

Figura 7 - Exemplo de gráfico de Gantt para uma solução ótima de sequênciamento.



Fonte: Kuhpfahl (2016)

2.1.3 Algoritmo Genético

Os algoritmos genéticos são técnicas heurísticas de otimização global e estes fazem parte da família de códigos que utilizam os conceitos de algoritmos evolucionários, que são aqueles que usam modelos computacionais dos processos evolutivos naturais para buscar soluções de problemas. Apesar da grande variedade de modelos computacionais, todos têm em comum o conceito de simulação da evolução das espécies, através da reprodução e seleção de indivíduos de acordo com seu desempenho. (LINDEN, 2012).

Objetivando a solução de problemas através de heurísticas, o algoritmo genético busca a melhor solução viável em um tempo razoável, não sendo necessariamente a melhor solução. Sendo utilizadas portanto técnicas probabilísticas, e não técnicas determinísticas (LINDEN, 2012).

Assim, como um método heurístico, o algoritmo é suficientemente eficaz para lidar com problemas muito grandes e encontrar soluções próximas a ótima em um tempo razoável (HILLIER; LIEBERMAN, 2006).

A principal característica dos algorítimos genéticos é a geração e seleção de novas populações, como dito anteriormente, baseado em processos evolucionários naturais. Baseado em regras de seleção combinada com um operador de crossover, dois cromossomos são recombinados com o objetivo de produzir uma nova solução.

Além disso, operadores de mutação criam pequenas mudanças para diversificar as soluções criadas. Ao fim, novas populações substituem as antigas, menos aderentes aos objetivos, até que um critério de parada definido seja encontrado (KUHPFAHL, 2016).

2.2 ESTABELECIMENTO DO ESTADO DA ARTE

Esta seção está dividida em: (2.2.1) análise sistemática; (2.2.2) método de busca; (2.2.3) resultados e discussões

2.2.1 Análise sistemática

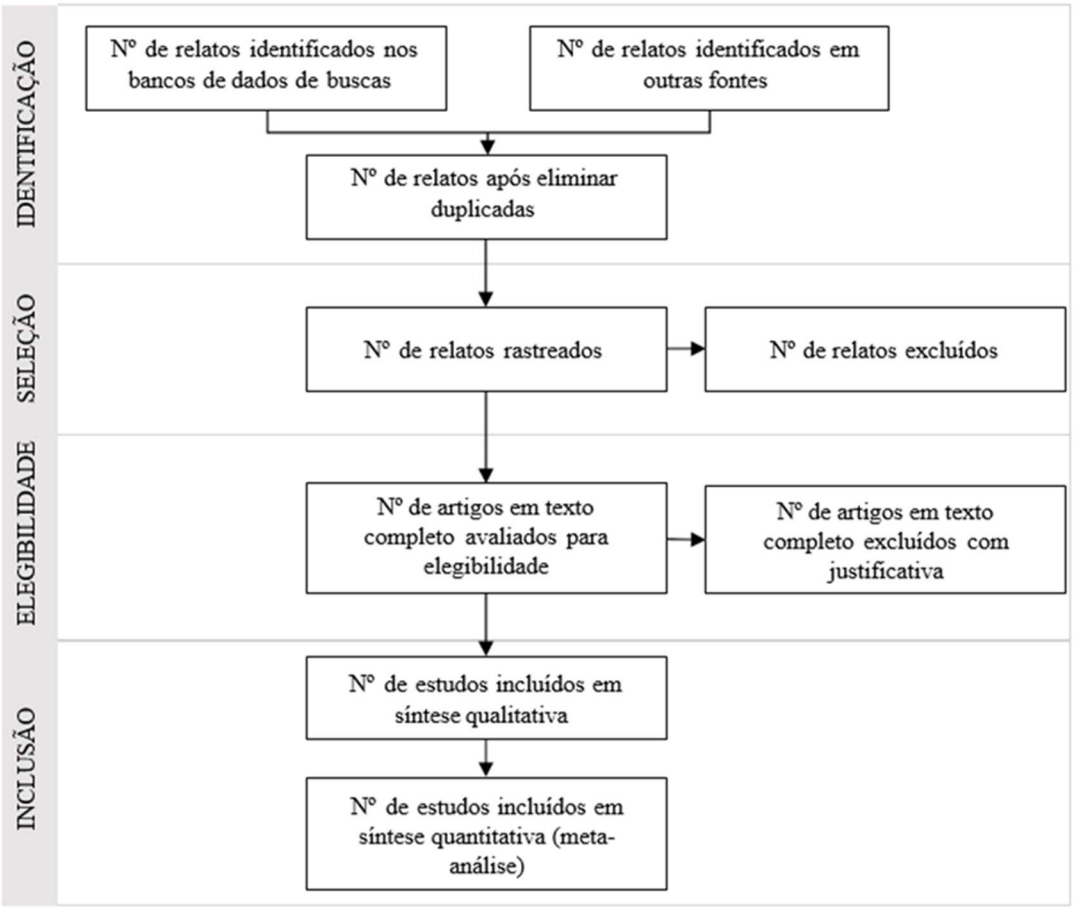
A revisão sistêmica é um método que vem se tornando cada vez mais importante para manter atualizada a relação de uma área de estudo específica. Porém a qualidade de relatórios de revisões sistêmicas não é otimizada e mesmo quando a possibilidade do viés da publicação é avaliada, não há garantia que os revisores fizeram a avaliação ou interpretação adequada (MOHER, 2015).

Neste sentido, neste trabalho é feito uma revisão sistemática na literatura para o estabelecimento e mapeamento do estado da arte dos sistemas de otimizações atuais desenvolvidos para os problemas de alocações e sequenciamento de atividades em áreas portuárias.

O método utilizado para este levantamento foi a adaptação da recomendação PRISMA - Preferred Reporting Items for Systematic Reviews and Meta-Analyses. O qual consiste em diretrizes que consideram um conjunto de itens para, de uma forma objetiva e imparcial, melhorar o relato de revisões sistemáticas e meta-análises.

Para a aplicação do checklist do fluxograma PRISMA, fez-se um ajuste da sistemática devido às características do presente estudo. O fluxograma descrito pela Figura 88 sintetiza as quatro fases descritas na recomendação PRISMA.

Figura 8 – Fluxograma PRISMA



Fonte: MOHER et al. (2015)

A partir da lógica proposta pelo PRISMA, foram escolhidos quatro critérios. Identificação, seleção, elegibilidade e inclusão - conforme descrito abaixo:

a) Identificação:

Para a execução do critério de identificação dos trabalhos que pudessem auxiliar na elaboração desse estudo, em janeiro de 2021 foi realizada uma revisão sistemática, através de pesquisa bibliográfica nas bases de dados tradicionais SCOPUS®, Web of Science® e Science Direct®. Tais buscas cumpriram os seguintes critérios:

Lógica da String de pesquisa:

• (TITLE-ABS-KEY ("jop shop scheduling" OR "job-shop scheduling" OR "parallel machine scheduling" OR "open-shop scheduling") AND TITLE-ABS-KEY ("port operations" OR "port" OR "wharf" OR "truck" OR "seaport" OR "harbor" OR

"logistic*" or "blending facility" or "storage tank") AND NOT TITLE-ABS-KEY ("symposium" OR "conference" OR "network*"));

- Período de publicação: houve limitação de 20 anos (2001 a 2021).
- Idioma: inglês e português.

b) Seleção:

A etapa de seleção constituiu-se na leitura dos títulos e resumos das publicações obtidas nas bases de dados. Em seguida, houve a triagem para a busca daqueles que abordassem a questão de sistemas de otimização de processos produtivos aplicados em terminais portuários.

c) Elegibilidade:

A fase de elegibilidade constituiu-se na leitura e análise crítica dos artigos e nesta etapa, foram considerados apenas aqueles que tivessem como foco o estudo de indicadores para sistemas de otimização de processos produtivos aplicados em terminais portuários.

Foram excluídos relatos que:

- Dissertassem sobre uso de sistemas de otimização de processos produtivos, mas em outras áreas de atuação, que não de um gerenciamento da produtividade de um terminal portuário;
- Abordassem apenas metodologias para a construção de indicadores, mas não citassem, de forma direta, estes indicadores;
- Tivessem foco em outro setor de atividades específico que não seja o setor portuário.

d) Inclusão:

Para fins de inclusão na revisão sistemática, foram selecionados os artigos nos quais foram identificados indicadores específicos de sistemas de otimização de processos produtivos aplicados em terminais portuários.

Os artigos incluídos, segundo a aplicação do fluxograma PRISMA, compuseram o conjunto de indicadores proposto por este artigo.

2.2.2 Método de busca

A estratégia de busca utilizada considerou dois conceitos relacionados ao método da solução proposta utilizada, chamada aqui de “Conceito quanto ao método” e relacionado ao local proposto no estudo, chamado de “Conceito quanto à aplicação”. Para o método de busca pelas palavras-chave de um mesmo conceito, foi utilizado o operador booleano (OU), já para palavras-chave entre conceitos, foi utilizado o operador booleano (E).

Tabela 2 - Estratégia de busca utilizada

CONCEITOS QUANTO AO MÉTODO	CONCEITOS QUANTO A APLICAÇÃO
Job shop scheduling	*port operations*
Job-shop scheduling	*port*
Parallel machine scheduling	*wharf*
Open-shop scheduling	*truck*
	seaport
	harbor
	logistic
	blending facility
	storage tank

Fonte: Elaborado pelo próprio autor (2021)

2.2.3 Critério de elegibilidade

Como critério de elegibilidades dos trabalhos, foram considerados aqueles que:

- Não fossem trabalhos provenientes de simpósio ou conferências;

- Apresentassem como resultado algum estudo em relação à aplicação dos métodos de otimização utilizados.

2.2.4 Resultados e discussões

Como resultado do levantamento realizado a partir da busca nas bases de conhecimento, foram encontrados 131 artigos que atendiam aos conceitos da busca. Destes, após as etapas constantes no método PRISMA, foram incluídos na síntese quantitativa para análise, 10 artigos, conforme estão resumidos na tabela 3.

Tabela 3 - Artigos selecionados

Nº	Ano	Título do Artigo
1	2012	*Berth allocation with time-dependent physical limitations on vessels*
2	2014	*Optimal Job Scheduling of a Rail Crane in a Rail Terminal*
3	2017	*A flexible crane scheduling methodology for container terminals*
4	2016	*A Cooperative Approach to Dispatching and Scheduling Twin-Yard Cranes in Container Terminals*
5	2018	*Scheduling cooperative gantry cranes with seaside and landside jobs*
6	2021	*Coordinated optimization of equipment operations in a container terminal*
7	2019	*A decomposition-based approach to the scheduling of identical automated yard cranes at container terminals*
8	2019	*Practical approaches to chemical tanker scheduling in ports: a case study on the Port of Houston*
9	2021	*Ship-unloading scheduling optimization for a steel plant*
10	2020	*A Job Sequencing Problem of an Overhead Shuttle Crane in a Rail-Based Automated Container Terminal*

Fonte: Elaborado pelo próprio autor (2021)

Xu, Li, Leung (2012) têm como objetivo a programação de alocação de navios a berços de atracação com restrição a calado e altura de maré. O problema é tratado como sendo um *parallel-machine scheduling problem* dividido em dois períodos com dois tempos de processamento nestes períodos sendo diferentes. Para isso, foi desenvolvida uma heurística considerando o problema estático e um problema dinâmico. No problema estático, todas as chegadas dos navios respeitam o tempo zero, ou seja, todas as chegadas disponíveis ao mesmo tempo, e a dinâmica o tempo de chegada dos navios podem ser diferentes de zero.

Nguyen e Yun (2014) estudam o problema de programação de descarga e carga de contêiner entre vagões e caminhões em um terminal ferroviário. Para a redução do *makespan* do problema foi utilizado um algoritmo de *parallel simulated annealing*.

Dik e Kozan (2017) utilizam uma heurística de busca tabu para um problema de alocação de guindastes de contêiner em comparação a soluções da literatura onde são propostos trabalhos fixos. Neste artigo, são propostos trabalhos flexíveis e dinâmicos. Os resultados foram positivos, com gerações de programações satisfatórias sendo gerados em um período consideravelmente curto.

Huang e Li (2016) apresentam um método baseado na teoria dos jogos para elaborar um modelo de comportamento da expedição e programação dos trabalhos dos guindastes do terminal para minimizar o tempo total do trabalho. Ao final, é realizada uma comparação dos algoritmos criados com outros já conhecidos.

Jaehn e Kress (2018) abordam o problema da programação de dois guindastes pórticos, objetivando a minimização do tempo de permanência dos navios nos berços de atracação. Para a solução do problema, foram utilizadas duas heurísticas onde levou-se em consideração algumas regras específicas do problema em particular para sua construção. As questões propostas foram positivamente respondidas, considerando aspectos realísticos, obtendo uma redução do *makespan* em 20%.

Jonker et al. (2021) desenvolvem um modelo coordenado de *scheduling* aplicado entre os equipamentos do terminal de uma forma conectada, para aumento de produtividade dos equipamentos do tipo portêiner em um terminal de contêineres. Para o modelo, os autores utilizam um *Hybrid Flow Shop* feito através de um algoritmo do tipo *Simulated Anneling*.

Eilken (2019) aborda a necessidade de minimização do tempo de operação dos navios em um terminal de contêiner objetivando o aumento da eficiência dos guindastes pelo seu planejamento operacional. O planejamento das atividades contempla a atribuição das atividades a cada guindaste, a sequência de tarefas e a programação do movimento dos guindastes sujeito as janelas de tempo e restrições precedentes.

Cankaya, Wari, Eren Tok (2019) têm como objetivo o desenvolvimento de um modelo matemático prático para a programação de navios tanques que visitam o porto de Houston (EUA). Inicialmente, toda a programação é feita através da regra "*first come, first served*"; entretanto, esta regra gera ineficiências uma vez que há uma série

de restrições operacionais e comerciais envolvidas, criando um problema de programação complexa. Para isso, dois métodos do tipo *Open-Shop Scheduling* foram propostos: programação nteira e programação por restrições.

Gao et al. (2021) estuda um problema de programação de descarga de navio para a minimização dos custos de sobre-estadia das operações marítimas de descarga de matérias primas do tipo granéis sólidos para uma usina de aço. Para isso, foi proposta uma técnica de otimização para problemas do tipo *Job Shop Scheduling* flexível, onde as esteiras transportadoras foram tratadas como máquinas e os navios, as atividades. Os resultados foram satisfatórios e espera-se uma redução de 20 milhões de *yuan* chineses desses custos.

Fibrianto, Kang, Hong (2020) apresentam um problema de sequenciamento de atividades (*Job Sequencing* P*roblem*) desenvolvido em duas etapas com a utilização de um algoritmo genético aplicado a um terminal de contêiner com guindastes automatizados. O objetivo foi a diminuição do tempo de atraso das atividades, considerando um tempo razoável para o cálculo da heurística.

Realizando um comparativo geral entre os trabalhos analisados, pode-se classificá-los quanto a três aspectos, que são: local de aplicação, objetivo e método de solução. Na tabela abaixo são exibidos os resultados desta classificação.

Tabela 4 - Resumo da análise sistemática

Nº	Autor	Local de Aplicação	Problema	Objetivo	Método de solução
1	Xu, Li, Leung	Terminal de contêiner	Alocação de berços de atracação	Minimização do custo total dos navios	Heurística
2	Nguyen e Yun	Terminal ferroviário	Sequenciamento de guindastes ferroviários	Minimização do *Makespan*	Parallel simulated annealing algorithm
3	Dik e Kozan	Terminal multimodal de contêiner	Planejamento de Portêiner	Minimização do tempo de *Turnaround*	Flexible neighbourhood search strategy baseado no algorítimo de Busca Tabu
4	Huang e Li	Terminal de contêiner	Planejamento de guindastes de pátio	Minimização do tempo de *Turnaround*	Algorítimo de programação baseado em non-zero-sum game
5	Jaehn e Kress	Terminal de contêiner	Planejamento de guindastes de pátio	Minimização do tempo de permanência de navios	Heurística
6	Jonker et al.	Terminal de contêiner	Planejamento coordenado entre Portêiner e guindastes de pátio	Minimização do tempo de Turnaround	Meta-Heurística de Simulated Annealing (SA)
7	Eilken	Terminal de contêiner	Planejamento de guindastes de pátio	Minimização do tempo de Turnaround	Heurística com algorítimo Branch and Bound
8	Cankaya, Wari, Eren Tok	Terminais químicos	Planejamento de navios tanques	Minimização do *Makespan*	Métodos matemáticos

9	Gao et al.	Terminal de granéis sólidos	Planejamento de atracação de navios	Minimização dos custos de Demurrage	Programação Inteira
10	Fibrianto et al.	Terminal de contêiner	Planejamento de guindastes de pátio	Minimização do tempo total de atraso	Heurística

Fonte: Elaborado pelo próprio autor (2021)

Analisando os resultados obtidos com a revisão sistemática, é possível identificar alguns pontos de predominância dos estudos desenvolvidos. De modo geral, dos 10 trabalhos analisados, 8 são desenvolvidos em terminais onde a carga movimentada são contêineres, 1 trabalho foi desenvolvido em um terminal de granéis sólidos, no recebimento de matérias primas de uma planta siderúrgica e 1 trabalho no porto de Houston de navios de cargas líquidas.

Como a maioria dos artigos abordam terminais de contêineres, o problema em estudo é o sequenciamento ou planejamento de tarefas ou atividades alocados aos recursos que são destinados para movimentação deste tipo de carga. Nguyen e Yun (2014), Dik e Kozan (2017), Huang e Li (2016), Jaehn e Kress (2018), Jonker et al. (2021), Eilken (2019) e Fibrianto, Kang, Hong (2020) estudam sobre o sequenciamento de atividades dos guindastes tanto do pátio de contêineres como dos berços, onde os contêineres são carregados ou descarregados dos navios.

Xu, Li, Leung (2012), Cankaya, Wari, Eren Tok (2019) e Gao et al. (2021) abordam o problema de sequenciamento de navios nos berços de atracação.

Em relação ao objetivo estudado, basicamente todos os trabalhos perseguem o mesmo objetivo ou objetivos similares do ponto de vista de seu conceito. Nguyen e Yun (2014) e Cankaya, Wari, Eren Tok (2019) objetivam a minimização do makespan enquanto Dik e Kozan (2017), Huang e Li (2016), Jaehn e Kress (2018), Jonker et al. (2021), Eilken (2019) objetivam a minimização do tempo de permanência dos navios ou *Turnaround*. Fibrianto, Kang, Hong (2020) objetivam a redução do atraso dos navios. Xu, Li, Leung (2012) e Gao et al. objetivam a minimização dos custos com os navios, sendo estes os custos totais ou o *demurrage*, o qual é a cobrança pela sobre-estadia do navio. Dessa forma, percebe-se a importância dos indicadores de performance dos navios, mesmo quando o problema estudado não está diretamente

relacionado com sua operação. Em especial a velocidade da operação dos navios é um fator importante no setor de transporte marítimo; por isso, há uma busca para minimização do tempo de *turnaround* dos navios (BISH, 2003).

Quanto aos métodos de solução utilizados, aproximadamente metade das metodologias utilizaram foram estocásticas ou determinísticos para tratamento do problema e a outra metade utilizou métodos Heurísticos e Meta-Heurísticos. Nguyen e Yun (2014) formularam o problema em um modelo matemático e propuseram a solução através de um algorítimo Branch and Bound e um algoritimo de *Parallel simulated Annealing*. Cankaya, Wari, Eren Tok (2019) e Gao et al. (2021) utilizam a métodos de programação inteira para solução de problemas de atracação de navios. Huang e Li (2016) utilizam um algorítimo baseado em teoria dos jogos para a solução do problema de programação de guindaste de pátio.

Por outro lado, Xu, Li, Leung (2012), Jaehn e Kress (2018), Eilken (2019) e Fibrianto, Wari, Eren Tok (2019) aplicaram soluções Heurísticas para os problemas propostos enquanto Dik e Kozan (2017) e Jonker et al. (2021) utilizaram métodos Meta-Heurísticos.

Dessa forma, no contexto geral percebe-se uma grande inclinação para problema aplicados em terminais de contêineres, entretanto prevalecendo o objetivo de minimização dos tempos operacionais dos navios cargueiros. Os recursos estudados foram desde problemas de alocações dos navios até máquinas relacionadas às operações de pátio. Já em relação aos métodos de soluções percebe-se uma diversidade quanto ao tipo, considerando métodos Determinísticos, Estocásticos, Heurísticos e Meta-Heurísticos.

3 METODOLOGIA DA PESQUISA

3.1 CARACTERIZAÇÃO DO PROBLEMA

A busca por métodos tecnológicos inovadores que aumentem a eficiência do sistema produtivo nos terminais portuários são prioridades para os gestores dessas organizações, que em muitos momentos acabam esbarrando em certas restrições.

Como exemplo, pode-se citar a limitação de espaço físico para novas expansões de capacidade ou a necessidade de racionamento dos ativos produtivos dos terminais, gerando dessa forma uma necessidade atual de aumento da eficiência. Dessa forma, a aplicação de modelos de otimização de fluxos e sequenciamento de atividades se apresentam como boas soluções.

A organização das atividades dos terminais é feita na maioria das vezes de forma puramente empírica, sem uma estratégia previamente traçada de maneira sistemática e científica. O presente trabalho identifica nisto uma oportunidade de aumento da produtividade dos terminais, sem um aumento significativo dos custos. Neste sentido, modelos de otimização dos sistemas decisórios das atividades de expedição rodoviária podem ser uma alternativa para que haja aumento da capacidade produtiva.

Descrevendo os terminais portuários de forma geral, estes se caracterizam em sua grande maioria pelo recebimento de produtos ou mercadorias vindas de navios que atracam nos chamados berços de atracação. A carga é descarregada no pátio de armazenamento para, neste caso, tanques de armazenamento. Em seguida a carga é transferida para caminhões-tanque em lotes menores que seguem para os consumidores finais em regiões próximas ao porto.

Um dos agravantes destes processos é que o terminal tende a sofrer um desbalanceamento da demanda do mercado, o qual se mostra não linear. Ou seja, por fatores do próprio mercado, em certos períodos a demanda é atendida de forma adequada com um nível de serviço satisfatório, porém, em outros períodos o sistema de carregamento torna-se o gargalo do processo de transferência de mercadorias,

ocasionando em possíveis atrasos das operações marítimas. Isto resulta numa baixa qualidade no atendimento de toda demanda, devido aos atrasos dos carregamentos.

Para uma melhor compreensão do cenário estudado, se faz necessária a descrição simplificada do processo macro do carregamento rodoviário, a qual é composta basicamente de 5 etapas principais.

Figura 9 – Macroprocesso do carregamento rodoviário

Fonte: Elaborado pelo próprio autor (2021)

Etapa 01 - Programação da retirada da carga pelo cliente (responsável pela carga)

Etapa 02 - O caminhão se dirige ao pátio de carregamento registrando seu acesso de entrada. Neste momento acontece o registro do horário por meio do sistema de automação do terminal.

Etapa 03 - O carregamento é feito de acordo com os produtos na programação realizada pelo cliente. Essa programação pode variar em relação aos tipos de produtos a serem carregados e suas quantidades.

Etapa 04 - Após o carregamento ser realizado, há o acerto fiscal que consiste na emissão da nota fiscal do produto.

Etapa 05 - Por último, há a saída do caminhão do pátio, onde todo o processo é finalizado.

Este processo acontece 6 dias por semana durante 24h por dia, havendo apenas uma interrupção de 1 hora no dia, às 23h para elaboração de relatórios de fechamento do dia.

Todas as etapas durante o carregamento são registradas via sistema de automação e podem ser consultados ou utilizadas como alimentação para outros

sistemas. Os marcos importantes para este estudo são os horários de acesso ao terminal, início dos carregamentos e o término dos carregamentos.

Além disso, é importante detalhar a infraestrutura onde acontecem os carregamentos, para que haja a correta compreensão do modelo desenvolvido.

Todos os caminhões que serão carregados deverão estar posicionados em baias próprias para este fim, sendo que nestas baias existem CLPs (Computador Lógico Programável) que realizam os carregamentos automaticamente. Porém, é importante destacar que cada CLP tem a capacidade de carregar apenas um tipo de produto. Desta forma, caso um veículo necessite carregar 3 produtos diferentes, ele deverá operar 3 CLPs diferentes.

Nas Figura 1010 e 11 são exemplificadas as plataformas de carregamento de caminhões.

Figura 10 – Plataforma de carregamento com 4 baias

Fonte: Elaborado pelo próprio autor (2021).

Figura 11 – Plataforma de carregamento com 8 baias

Fonte: Adaptado de Tribuna Hoje. Disponível em: <tribunahoje.com/noticias/brasil/2021/03/19/ale-combustiveis-oferece-oito-vagas-de-emprego-e-tres-de-estagio>. Acesso em: nov 2021.

No momento em que o veículo entra no pátio, um subprocesso importante é executado, que é a decisão em relação a qual ponto de carregamento (ou baia) o caminhão realizará sua operação. Atualmente, esse processo é absolutamente empírico e depende do operador do terminal que supervisiona as plataformas de cargas. Em outras palavras, não há nenhum tipo de procedimento sistematizado na alocação dos veículos às baias, tornando o processo bastante suscetível a variações.

No momento em que um caminhão é alocado em alguma baia, seu processo de carga é iniciado e um produto é carregado por vez. Dependendo do requerimento do cliente, um caminhão pode variar em relação a:

I. Tipos de produtos a serem carregados;

II. Volume de cada produto.

Em relação aos tipos de produtos, no terminal atualmente existem 5 diferentes tipos de produtos. Cada produto é comandado por um PLC individual, sendo que, o carregamento é realizado através de braços de carga, que são tubulações que acoplam nos compartimentos dos caminhões e permitem a transferência de carga, conforme Figura 122.

Figura 12 – Braços de carregamento de caminhões

Fonte: Catálogo Incontrol. Disponível em: https://incontrolprojects.co.uk/product/top-loading. Acesso em: nov 2021.

Fazendo uma ampliação da visão em uma plataforma de carga, tem-se uma estrutura metálica onde são instalados todos os equipamentos responsáveis pela transferência e controle das cargas dos caminhões. Tais estruturas metálicas são chamadas de ilhas, e cada ilha consegue atender a 2 baias de carregamento, que estão adjacentes a ela, em outras palavras, uma ilha de carga está ao meio de duas baias de carregamento, possibilitando o atendimento das duas com a utilização dos braços de carregamento.

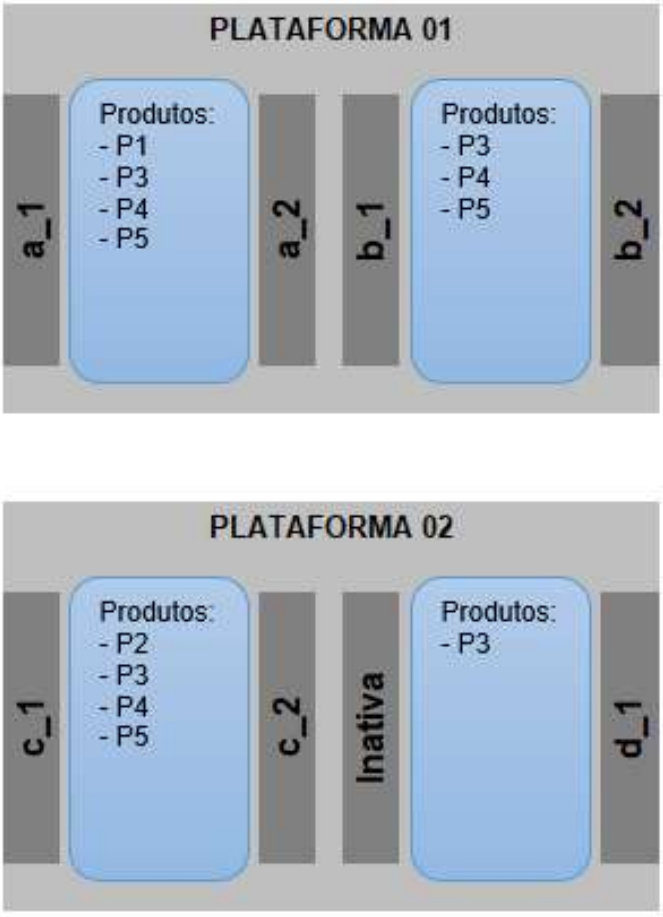
Cada braço de carregamento é capaz de atender duas baias vizinhas através de seu movimento de rotação 360º. Por isso, uma vez que inicía-se a utilização do braço de carregamento em um desses veículos, consequentemente o produto daquele braço se torna indisponível para o outro veículo. Sendo assim, este problema pode ser caracterizado como sendo do tipo *Parallel Machine Scheduling.*

Considerando as informações supracitadas nota-se que, dependendo dos tipos de produtos programados para as cargas, na ausência de um correto sequenciamento e alocação desses veículos, a concorrência de produtos em baias vizinhas se torna prejudicial. Como consequência, nota-se uma perda de produtividade pela ociosidade dos veículos que aguardam na espera do braço.

Ao todo, no terminal em estudo, existem 2 plataformas que somam 7 baias de carregamento, capazes de expedir 5 tipos de produtos diferentes, chamados neste trabalho de ***p1***, ***p2***, ***p3***, ***p4***, ***p5***.

Na Figura 133 é exemplificada a disposição dos produtos nas baias de cada plataforma. Assim, cada produto é identificado por um código que, adicionado ao código de sua localização nas plataformas, servirá como base para construção do modelo de otimização.

Figura 13 – Disponibilidade dos produtos nas baias de carga



Fonte: Elaborado pelo próprio autor (2021).

É importante informar que, um caminhão pode ser carregado com vários produtos diferentes uma vez que cada caminhão é construído com um *layout* diferente, ou seja, pode haver caminhões com apenas um compartimento, até caminhões com 09 compartimentos, por exemplo. Cada um desses compartimentos pode ser carregado com um produto específico.

3.2 CLASSIFICAÇÃO DE PESQUISA

Este trabalho desenvolve a solução de um problema prático e aplicado ao desenvolvimento de um modelo de otimização operacional de um terminal portuário, ou seja, uma representação de um conjunto de elementos e métodos que estão interrelacionados e seguem conceitos específicos criados para a utilização em um problema específico, a fim de alcançar a otimização de fluxos rodoviários e reduzir seus tempos operacionais.

Com base no disposto por Gil (2002), este trabalho classifica-se como sendo de natureza descritiva, uma vez que ele descreve e avalia o aumento de indicadores de performance através da elaboração e avaliação de um modelo computacional de otimização de fluxo rodoviário.

Em relação aos procedimentos utilizados, o trabalho é um estudo de caso, onde são levantadas questões específicas do local estudado e a aplicação acontece especificamente neste ambiente considerando suas características. Entretanto, o conceito e aplicação do modelo podem ser utilizados em outros locais considerando possíveis adaptações.

Além disso, de acordo com Vergara (2016), as pesquisas podem ser definidas quanto aos fins e quanto aos meios. Quanto aos fins, pode-se definir como exploratória, descritiva, explicativa, etc.. Quanto aos meios pode ser pesquisa de campo, pesquisa de laboratório, documental, etc..

Considerando aos meios de investigação deste trabalho, pode-se classifica-lo como uma pesquisa aplicada, ao passo que esta se justifica pela necessidade de resolução de problemas práticos. Quanto aos fins, este trabalho é classificado como descritivo, pelo teor de descrição de um problema prático.

O problema prático foi a busca pela melhoria da otimização dos transbordos de cargas rodoviárias dentro de um terminal portuário do porto de Vitória, situado no estado do Espírito Santo, com a aplicação de um algoritmo para redução dos tempos nos carregamentos de granéis líquidos.

Os granéis líquidos envolvidos são basicamente combustíveis derivados de petróleo ou Biocombustíveis, como Gasolina, Disel S00, Diesel S10, Álcool Hidratado, etc.. E em média, por dia são carregados 200 caminhões.

3.3 VARIÁVEIS DA PESQUISA

Para a avaliação do estudo deste trabalho e o acompanhamento quanto ao sucesso do modelo proposto, se faz necessário escolher algumas variáveis de avaliação do processo, as quais apresentação valores que leverão a um melhor entendimento do sucesso do modelo proposto.

Dessa forma, é considerado neste estudo o uso de variáveis independentes e dependentes, seguindo o conceito de variáveis de Lakatos (1992). As variáveis independentes dizem respeito ao nível ou quantidade de fluxo rodoviário presente no sistema para carregamento, ou seja, é a quantidade de veículos no sistema no aguardo do carregamento. Esta variável é entendida como independente pois não sofre interferência durante o processo, sendo apenas uma demanda do cliente. Já as variáveis dependentes dizem respeito ao tempo de processamento do modelo proposto em relação ao objetivo de sequenciamento e alocação dos veículos aos postos de carga e, como segunda variável dependente, o ganho ou aumento da produtividade do terminal em relação à realidade sem o auxílio de uma ferramenta computacional, onde haverá o ganho ou a perda de tempo de operação total de um conjunto de veículos.

3.4 COLETA DE DADOS

Os dados utilizados neste estudo são obtidos por meio de uma consulta ao sistema de automação dos carregamentos de caminhões da empresa estudada, que teve o nome omitido no trabalho por uma decisão da própria empresa, sendo estes interpretados e analisados através de uma planilha eletrônica.

É importante observar que a confiabilidade dos dados é garantida devido ao sistema de automação da empresa, onde os registros dos tempos de cada operação são feitos por acessos automáticos, que são então registrados automaticamente e armazenados nos servidores. Por isso, a disponibilidade histórica dos registros, caso haja necessidade, é uma vantagem deste processo de carregamento e posteriores estudos e análises.

Todos os registros dos carregamentos são divididos individualmente por cada veículo que opera no terminal e contém os seguintes tempos:

I. Entrada no sistema para carregamento;
II. Início do carregamento;
III. Fim do carregamento.

Todos os dados foram coletados ao longo do ano de 2021. Entretanto, para a experimentação e avaliação do modelo, foram utilizados dados do carregamento de

todos os dias do mês de outubro de 2021, em razão de ser um mês no qual houve pouca interferência no sistema de carregamento, por exemplo uma manutenção em um equipamento ou a necessidade de interrupção de algum serviço.

Além disso, neste mês ocorreram diferentes cenários de movimentação no terminal; por exemplo, cada veículo que opera no terminal pode levar uma quantidade de carga diferente uns dos outros, como dito anteriormente, podendo haver caminhões com cargas muitos fracionadas (pouca quantidade e muita diversidade) como cargas de transferência (muita quantidade e pouca diversidade). Assim, pode-se ter um ambiente quase que totalmente livre de interferências externas e um ambiente operacional com diversos perfils de carga.

3.5 PROCEDIMENTOS TÉCNICOS

Após a fase de coleta de dados, inicia-se o tratamento e análise dos dados reais das operações, onde a validação desses dados ocorre com o gestor da área da empresa para garantir que os dados coletados refletem o ambiente real de carregamento. Após isso é feito a organização e compilação das informações. Então os procedimentos técnicos de construção do modelo são realizados.

Para a validação da funcionalidade do algoritmo, são utilizados dados reais dos tempos das operações, obtidos pelo registro dos sistemas de automação da empresa.

A Figura 144 mostra as etapas de obtenção dos dados que serão utilizados como base para o algoritmo.

Figura 14 – Fluxograma da obtenção de dados para o algoritmo.

Fonte: Elaborado pelo próprio autor (2021).

Para implementação do modelo, um algoritmo é desenvolvido utilizando a linguagem de programação Python e construído na IDE Spyder™.

Logo após a construção do modelo, as informações dos carregamentos rodoviários são inseridas no algoritmo de otimização, e após a execução deste algoritmo, há a comparação dos resultados obtidos em relação a performance do carregamento real, para avaliação do desempenho do modelo.

4 RESULTADOS

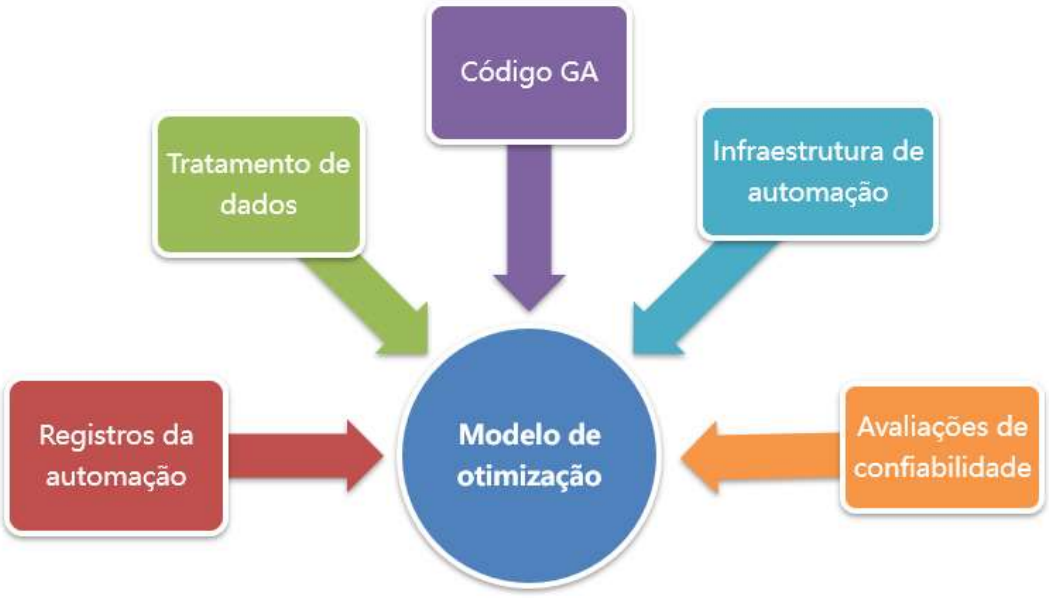
4.1 ELABORAÇÃO DO MODELO

A denomicação de modelo de otimização estudado neste trabalho se caracteriza pela representação de um conjunto de elementos e procedimentos que, podem ser utilizados como um produto final para a empresa na resolução de um problema específico, por isso, apenas o desenvolviemento de um algoritmo de otimização seria algo muito genérico e por isso, não se caracterizaria como uma solução de otimização para o problema especificamente.

Além disso. O modelo de otimização deste trabalho é proposto considerando-se a oportunidade de melhoria identificada com os gestores da empresa e suportado pela avaliação da falta de racionalização no processo de tomada de decisão dos sequenciamentos dos veículos e suas alocações às baias de carga.

Atualmente, todo o sequenciamento segue o modelo FCFS *(First Come First Served)*, conforme modelo definido por Dudin, Klimenok, vishnevsky (2020). Sendo assim, oportunidades de diminuição do tempo total da fila do sistema são perdidas devido ao modelo atual.

Figura 15 – Elementos do modelo proposto



Fonte: Elaborado pelo próprio autor (2021).

A Figura 155 resume os elementos necessários para implantação do modelo na prática. Porém, neste trabalho tratou-se apenas de alguns deles.

A “Infraestrutura de automação” e “Avaliações de confiabilidade” do modelo, que requerem elementos mais detalhados e devem ser mais bem estudados para a aplicação correta no ambiente do terminal, não serão tratados nessa oportunidade.

O objetivo nesta etapa é a apresentação inicial em caráter experimental através dos registros disponíveis e considerando a realidade atual do terminal, de um modelo capaz de subsidiar a avaliação para o incremento da produtividade do terminal, municiando também os gestores da companhia na tomada de decisão para a possibilidade de implementação de modelos mais autônomos, onde a tomada de decisão se torna mais racional e sistemática.

4.2 CONSTRUÇÃO DO ALGORITMO

O algoritmo proposto utiliza o conceito evolutivo de um algoritmo genético para, neste caso, encontrar uma solução próxima à ótima em um tempo rezoavelmente curto. Tal escolha foi feita devido à necessidade do modelo ser utilizado várias vezes ao longo do dia como sequenciador de veículos em fila.

Para avaliação das soluções factíveis, uma função objetivo deve ser considerada e por isso, a sua minimização se torna o objetivo do problema. A representação da função objetivo é então construída a partir das variáveis do problema utilizando uma representação matemática resumida:

$$V = \{V_1, V_2, V_3, \dots, V_n\}$$

Onde, Vn é o enésimo veículo da fila de espera. V_1

$$M = \{M_1, M_2, M_3, M_4, M_5\}$$

Onde, M é a lista de máquinas disponíveis no sistema.

Assim, cada veículo é composto por um vetor cujos elementos correspondem ao tempo de operação necessário em cada máquina, conforme equação abaixo:

$$Vj = [Mq_1, Mq_2, Mq_3, Mq_4, Mq_5] \text{ para todo j = 1, 2, 3, ..., n.}$$

Onde, Vj é o j-ésimo veículo do sistema. Mqx, é um coeficiente de tempo proporcional a quantidade de carga a ser carregada pela máquina específica. Uma característica do sistema é que pode haver troca da ordem de carregamento em cada máquina, ou seja, a ordem para os carregamentos não precisa obrigatoriamente seguir uma sequência específica.

Sendo assim, o objetivo do algoritmo é a minimização do *Makespan,* o qual é o tempo requerido desde o início da operação do primeiro veículo até a conclusão da operação do último veículo.

No caso específico da experimentação e validação do modelo, os elementos Mqx foram substituídos por tempos de processamento de dias reais, dessa forma, haverá maior precisão dos resultados gerados, do contrário, a comparação entre o resultado do modelo e o tempo real dos carregamentos sofreria influência desse coeficiente de tempo.

Conforme mencionado anteriormente, o código foi elaborado na linguagem Python através da IDE Spyder™ e o pseudocódigo segue conforme a

Figura 166.

Figura 16 – Pseudocódigo do algoritmo genético

Início Algoritmo Genético
1. *Gera população inicial randomicamente de acordo com 'tamanho_população' P(0)*
2. *n_gerações = 0*
3. ***Enquanto** n_gerações < config_gerações, **faça***
4. *Avalia a população inicial pelo Fitness*
5. *Aplica Crossover de acordo com probabilidade_fitness*
6. *Seleciona solução Elite*
7. *Gera descendentes (tamanho_população - 1)*
8. *Aplica mutação nos descendentes de acordo com probabilidade_de_mutação*
9. *n_gerações++*
10. ***Fim enquanto***

Fim algoritmo

Fonte: Elaborsxado pelo próprio autor (2021).

O código completo está descrito no anexo A.

Inicialmente, uma população é formada aleatoriamente, onde cada indivíduo é formado por no máximo 5 cromossomos, que representa a posição do seu carregamento para cada produto que este for carregar.

No algorítimo genético, cada indivíduo é traduzido como uma solução viável. E os cromossomos representam as ordens de carregamentos de cada veículo.

Por exemplo, considerando um sistema com 4 veículos em fila, um indivíduo será uma possível solução viável com uma disposição de cromossomos que será a ordem de carregamento de cada veículo. Por isso, um indivíduo ou cromossomo poderá ser representado da seguinte forma:

Indivíduo A

$[V_0]$, $[a1\_p1; a1\_p2]$;

$[V_1]$, $[a2\_p1; a2\_p2]$;

$[V_2]$; $[b1\_p3]$;

$[V_3]$; $[b2\_p4; b2\_p3]$;

Ou

Indivíduo B

$[V_3]$; $[b2\_p3; b2\_p4]$;

$[V_2]$; $[b2\_p3]$;

$[V_0]$, $[a1\_p1; a1\_p2]$;

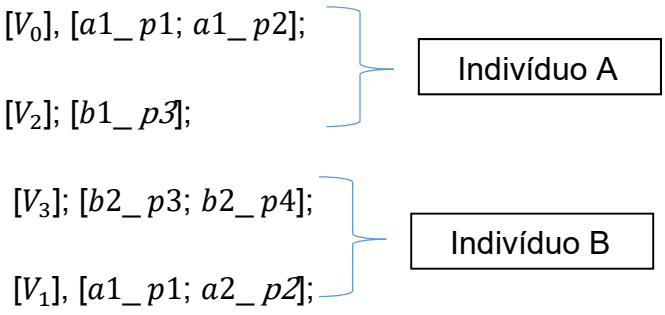
$[V_1]$, $[a1\_p1; a2\_p2]$;

Após a avaliação da solução viável através da função f*itness*, a aplicação do *Crossover* é feita considerando sua probabilidade de reprodução de acordo com sua

posição em relação a função *fitness,* por isso, quanto melhor for o indivíduo, maior será sua probabilidade de reprodução.

Na função de *Crossover* é feita a troca de cromossomos aleatoriamente entre dois indivíduos, gerando uma nova solução, como exemplificado abaixo:

Novo Indivíduo



Para inserção das informações referentes aos veículos no modelo experimental, foi utilizada a base de dados do mês de outubro/2021, considerando que, o tempo de processamento em cada máquina foi o tempo real para garantir maior fidelidade do modelo em relação ao seu desempenho.

Na **Erro! Fonte de referência não encontrada.**5 é exibido o relatório em planilha eletrônica após o tratamento dos dados que servirão de input para o algoritmo.

Tabela 5 - Histório de registro dos carregamentos

ID	Máquina 01	Máquina 02	Máquina 03	Máquina 04	Máquina 05	Chegada	Término	Tempo de corte	Makespan
483326	0	0	4	15	34	04/10/2021 09:18	04/10/2021 12:14	04/10/2021 12:14	
483330	0	0	0	10	13	04/10/2021 10:20	04/10/2021 12:05	04/10/2021 12:00	
483409	0	0	0	33	0	04/10/2021 09:53	04/10/2021 12:27	04/10/2021 12:27	
483424	0	0	4	14	0	04/10/2021 09:45	04/10/2021 12:11	04/10/2021 12:12	
483312	0	0	0	5	15	04/10/2021 10:23	04/10/2021 12:32	04/10/2021 12:33	
483416	0	0	4	4	17	04/10/2021 09:54	04/10/2021 12:28	04/10/2021 12:29	
483401	0	0	0	19	0	04/10/2021 10:30	04/10/2021 12:52		19,1
483343	0	0	0	35	0	04/10/2021 10:15	04/10/2021 13:32		59,133333
483315	0	0	9	4	14	04/10/2021 10:16	04/10/2021 13:24		51,466667
483336	7	0	16	0	0	04/10/2021 10:31	04/10/2021 13:31		58,466667
483131	0	0	13	10	11	04/10/2021 10:32	04/10/2021 14:00		87,083333
483390	0	0	14	5	0	04/10/2021 11:33	04/10/2021 14:00		87,383333
483306	0	0	14	12	10	04/10/2021 10:58	04/10/2021 14:30		117,88333
483332	0	0	21	8	18	04/10/2021 11:22	04/10/2021 14:51		138,15
483380	3	0	9	5	0	04/10/2021 11:40	04/10/2021 14:31		118,53333
483385	0	0	16	4	0	04/10/2021 11:16	04/10/2021 14:31		118,16667
483413	0	0	4	4	4	04/10/2021 11:47	04/10/2021 14:58		145,98333

Fonte: Elaborada pelo autor (2021).

É importante considerar na execução do modelo que, todos os veículos que são alocados devem estar na situação de fila, por isso, durante a etapa de tratamento dos dados, a identificação dessa distinção deve ocorrer. Em razão disso, durante o carregamento dos dados para o algoritmo, todos os tempos de chegada devem ser considerados como zero (todos veículos em fila) disponível para carregamento.

Por outro lado, no momento que o modelo for executado, deve-se lembrar que o processo de carregamento em curso de outras cargas poderá estar ocorrendo, ou seja, esses veículos não deverão ser utilizados no sequenciamento pelo algoritmo.

Para a fase de experimentação, algumas adaptações são feitas devido à complexidade da coleta desses dados, ao passo que no modelo aplicado, tais informações podem ser coletadas instantaneamente, por isso o tempo de carregamento de cada produto foi arredondado para cima em um múltiplo de 5.

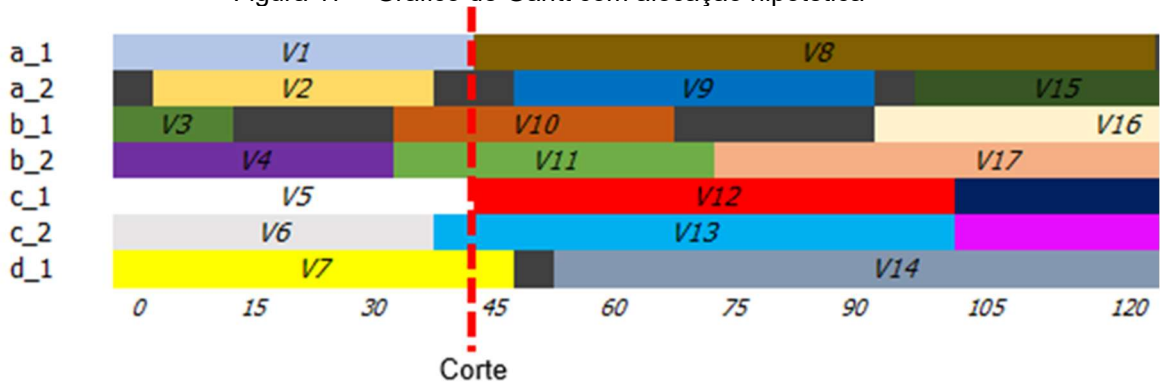
Exemplificando, caso o tempo real de carregamento tenha sido de 7 minutos, o tempo utilizado no algoritmo para aquele carregamento é de 10 minutos. Caso o tempo real de carregamento tenha sido de 14 minutos, o tempo utilizado no algoritmo foi de 15 minutos, e assim por diante.

O objetivo dessa adaptação é aplicar ao algoritmo possíveis tempos utilizados para movimentação dos trabalhadores, como por exemplo estacionamento e deslocamento.

Outra adaptação feita nos dados é que todas as máquinas durante a execução do algoritmo estão disponíveis a partir do tempo 0, então para garantir um aumento da precisão do modelo, há uma avaliação para que a experimentação do dia seja feita em um momento em que o número de máquinas ou o somatório do tempo remanescente para o término da operação de cada veículo seja o menor possível. Esse momento é tratado neste estudo como tempo de corte, que é o momento em que se busca o momento operacional que haja o maior número de máquinas paradas ou prestes a operar.

Para exemplificar a situação o tempo de corte, é exibido na Figura 17 um cenário hipotético de carregamento, no qual há a escolha do tempo para considerar os veículos que devem ser relacionados como em fila para o modelo.

Figura 17 – Gráfico de Gantt com alocação hipotética



Fonte: Elaborada pelo autor (2021).

Neste caso os veículos considerados em fila são: *V8; V9; V10; V11; V12; V13; V15; V16; V17 e V14.*

As configurações iniciais para de execução do algoritmo são:

- Número de gerações: 200
- Tamanho da população: 50
- Taxa de mutação: 5%

4.3 EXPERIMENTOS COMPUTACIONAIS

Como dito anteriormente, para avaliação do modelo, a experimentação foi feita durante todos os dias operacionais com a utilização dos registros do mês de outubro de 2021. Adicionalmente, o horário do cenário a ser avaliado é em torno de 12h da manhã, ou seja, o modelo é executado considerando o cenário do terminal às 12h da manhã aproximadamente, ajustado ao horário de corte (Figura 17).

O resultado da análise é mostrado na Tabela 6, no qual são compilados as informações do makespan real (min), o makespan do modelo (min), o ganho/perda do tempo (min) e o percentual do ganho/perda.

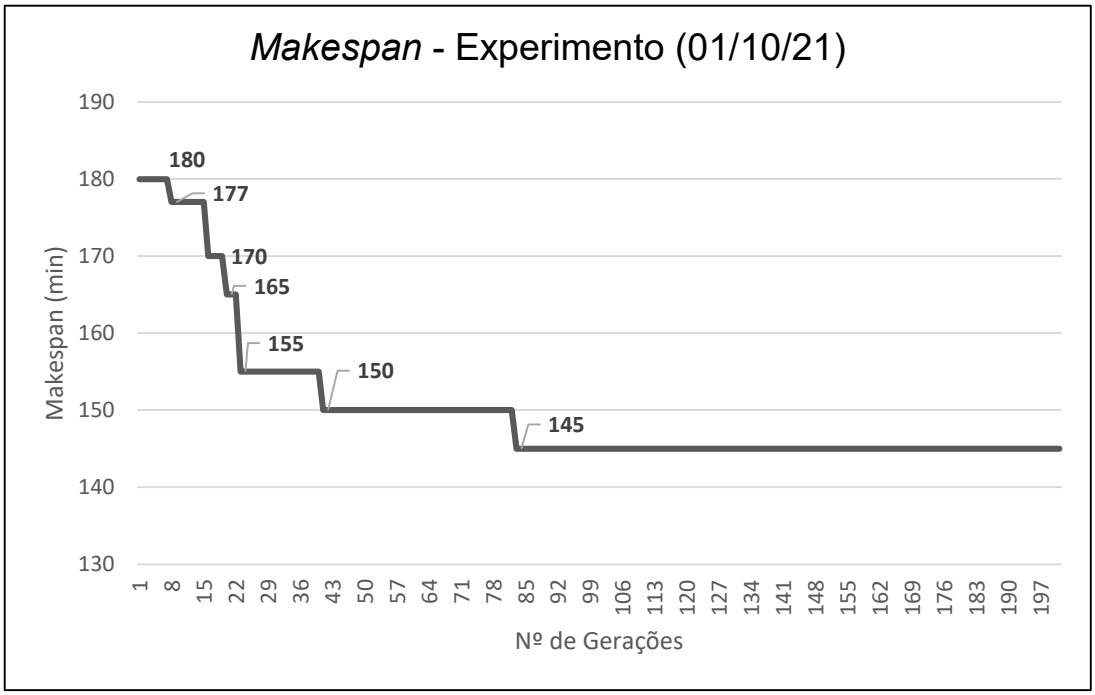
Tabela 6 - Resultados das experimentações

Data	Veículos em fila (und)	Makespan real (min)	Makespan Modelo (min)	Ganho (min)	% de ganho de tempo
01/10/2021	19	157	145	12	7,6%
02/10/2021	7	118	80	38	32,2%
04/10/2021	11	146	125	21	14,4%
05/10/2021	12	119	95	24	20,2%
06/10/2021	28	305	270	35	11,5%
07/10/2021	30	281	245	36	12,8%
08/10/2021	51	549	515	34	6,2%
09/10/2021	26	210	240	-30	-14,3%
11/10/2021	21	198	175	23	11,6%
13/10/2021	14	214	155	59	27,6%
14/10/2021	19	158	105	53	33,5%
15/10/2021	19	116	115	1	0,9%
16/10/2021	1	46	*Não houve simulação		
18/10/2021	18	216	165	51	23,6%
19/10/2021	22	353	200	153	43,3%
20/10/2021	16	194	140	54	27,8%
21/10/2021	16	174	135	39	22,4%
22/10/2021	24	309	220	89	28,8%
23/10/2021	7	97	70	27	27,8%
25/10/2021	23	129	125	4	3,1%
26/10/2021	21	173	150	23	13,3%
27/10/2021	9	85	75	10	11,8%
28/10/2021	20	200	160	40	20,0%
29/10/2021	29	195	200	-5	-2,6%
30/10/2021	33	353	350	3	0,8%

Fonte: Elaborada pelo próprio autor (2021).

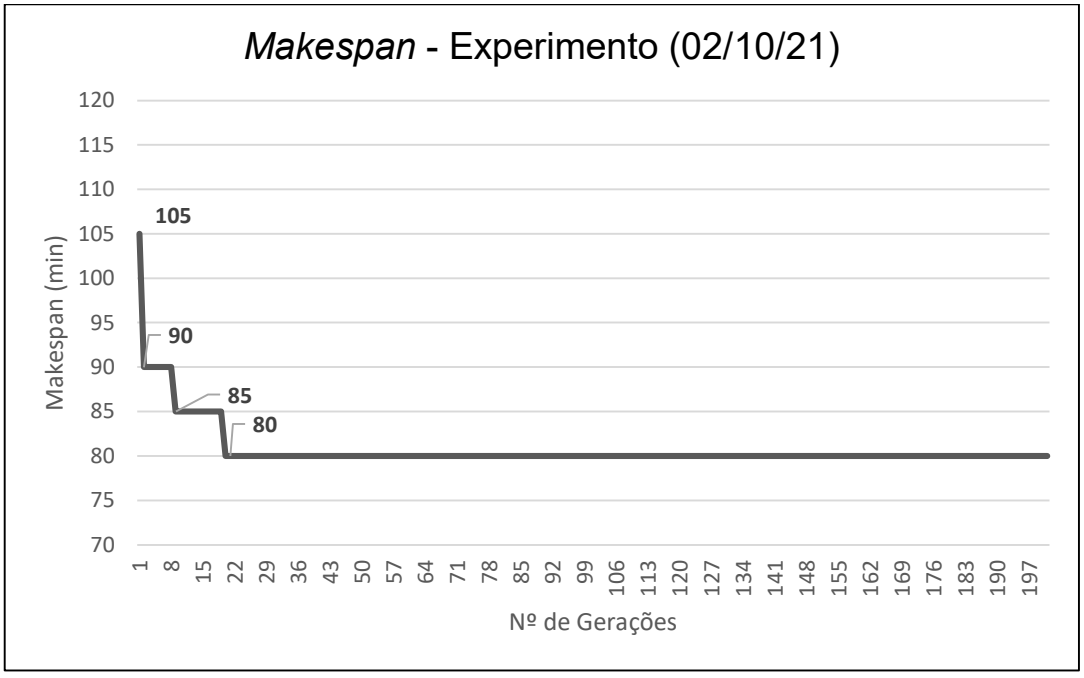
Abaixo, encontram-se as figuras da evolução do makespan do modelo ao longo das gerações.

Figura 18 – Detalhamento do *Makespan* 01/10/21



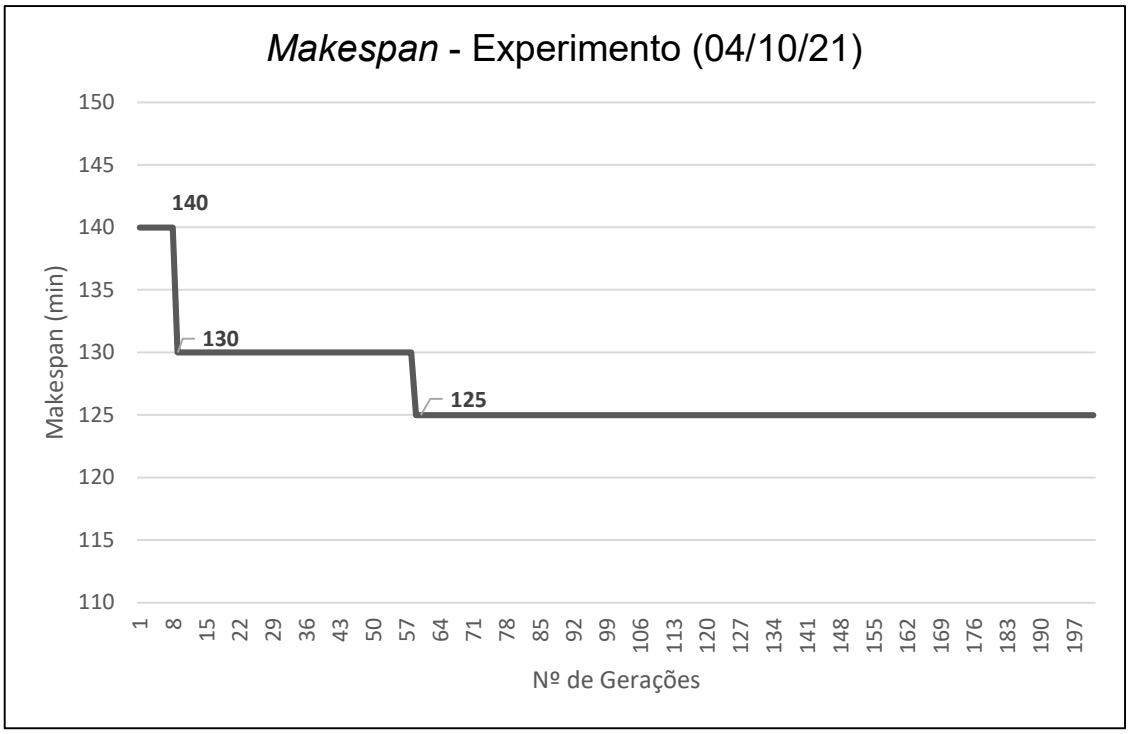
Fonte: Elaborada pelo próprio autor (2021).

Figura 19 – Detalhamento do *Makespan* 02/10/21



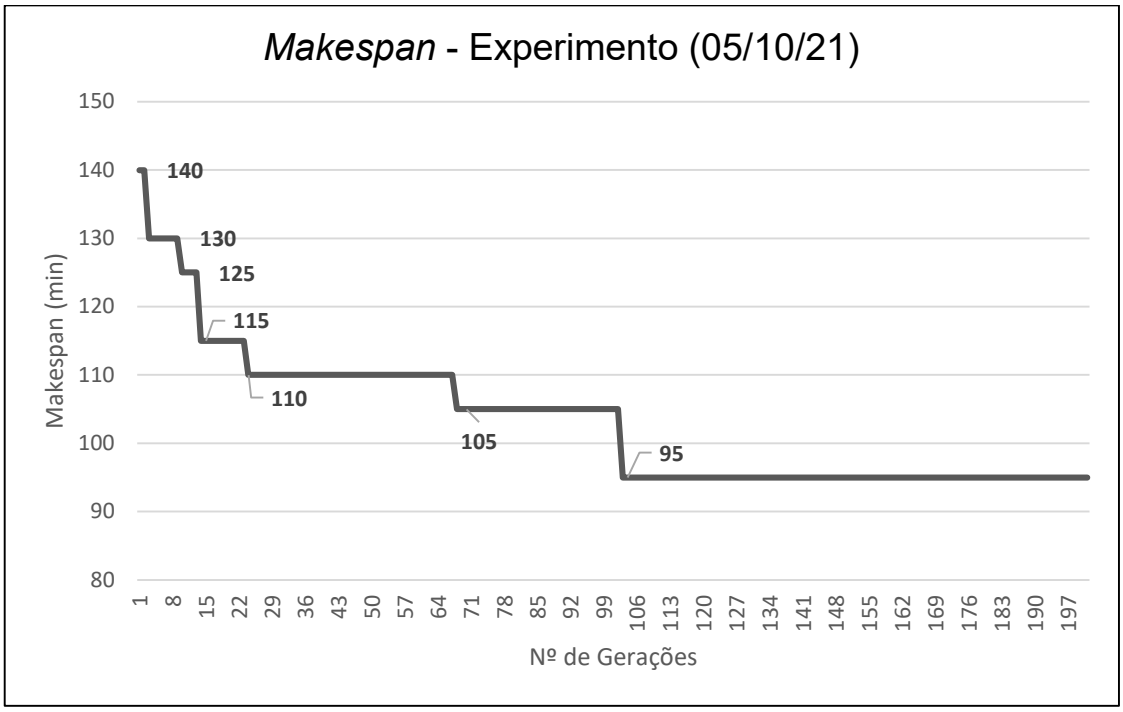
Fonte: Elaborada pelo próprio autor (2021).

Figura 20 – Detalhamento do *Makespan* – experimento 04/10/2021



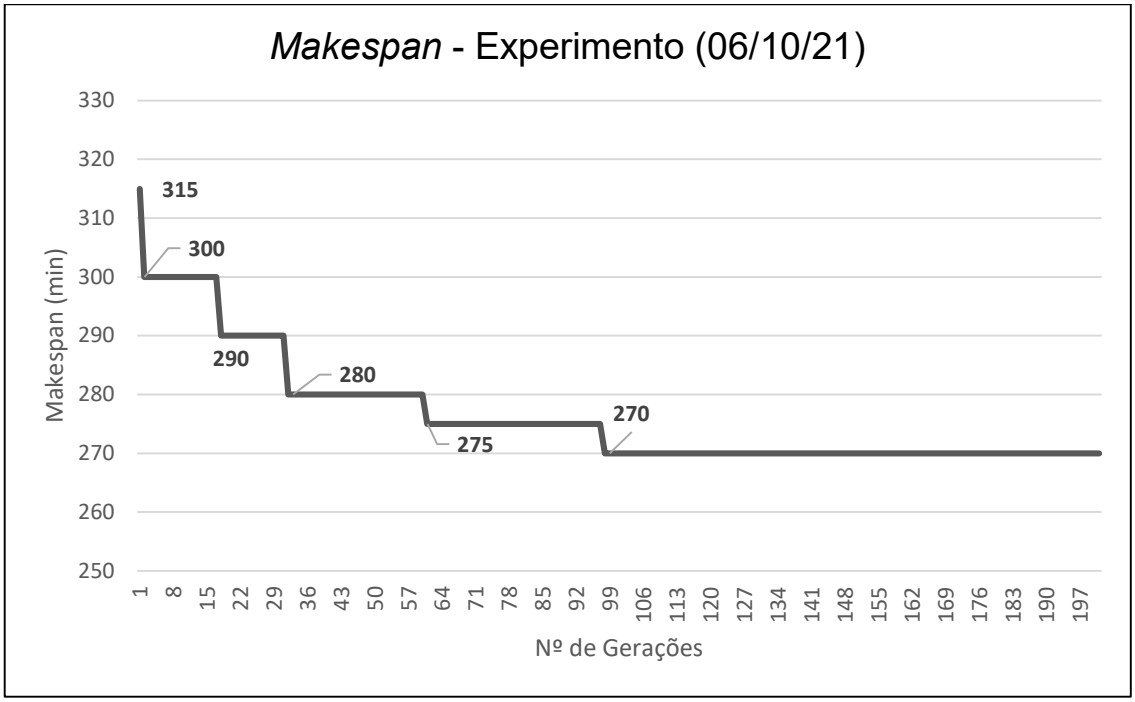
Fonte: Elaborada pelo próprio autor (2021).

Figura 21 – Detalhamento do *Makespan* – experimento 05/10/2021



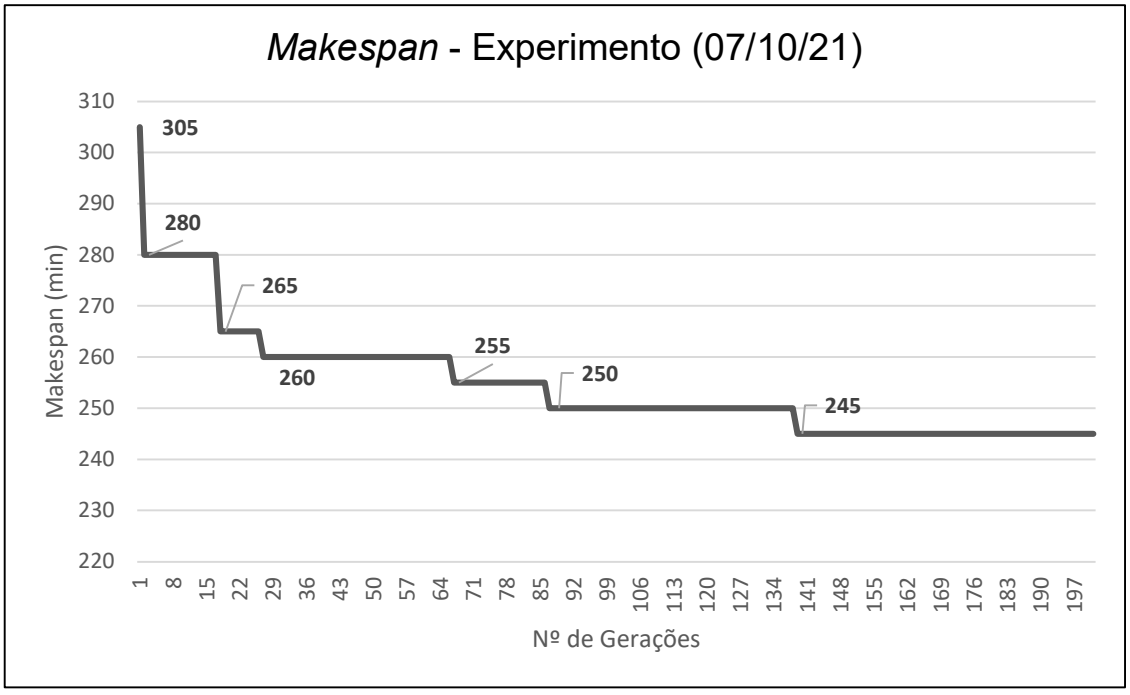
Fonte: Elaborada pelo próprio autor (2021).

Figura 22 – Detalhamento *Makespan* – experimento 06/10/2021



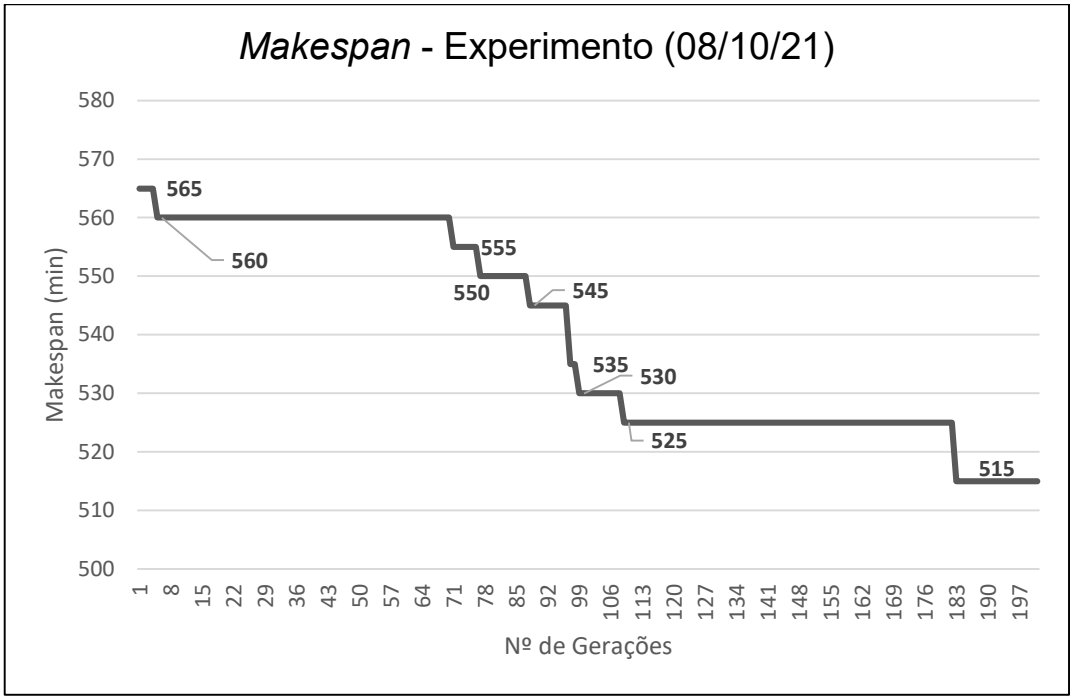
Fonte: Elaborada pelo próprio autor (2021).

Figura 23 – Detalhamento *Makespan* – experimento 07/10/2021



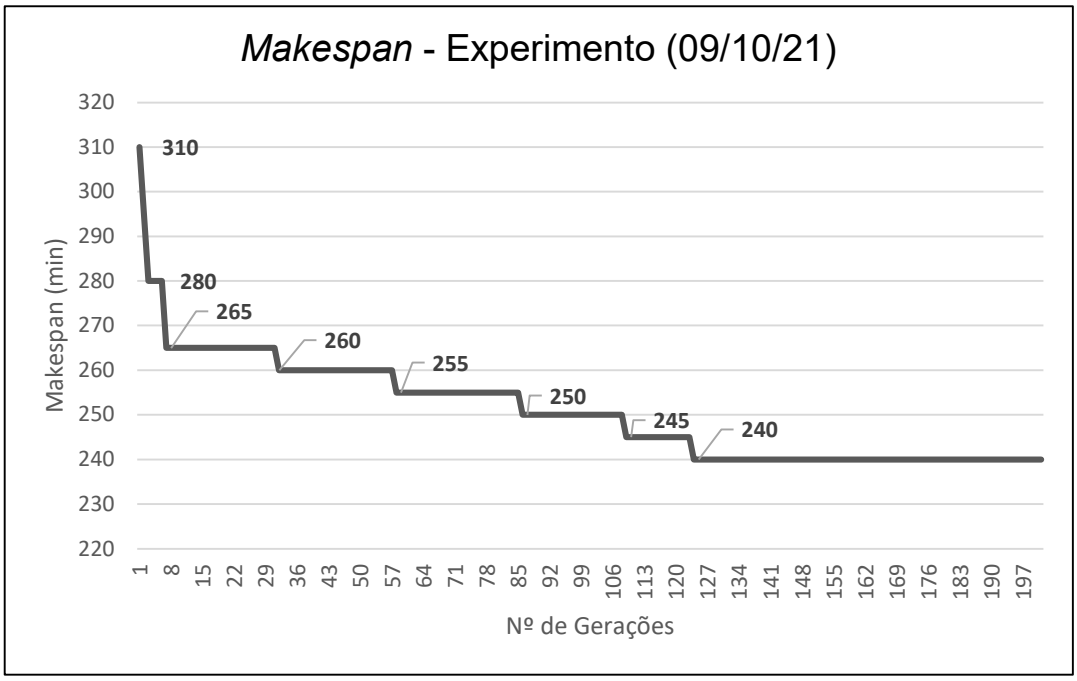
Fonte: Elaborada pelo próprio autor (2021).

Figura 24 – Detalhamento *Makespan* – experimento 08/10/2021



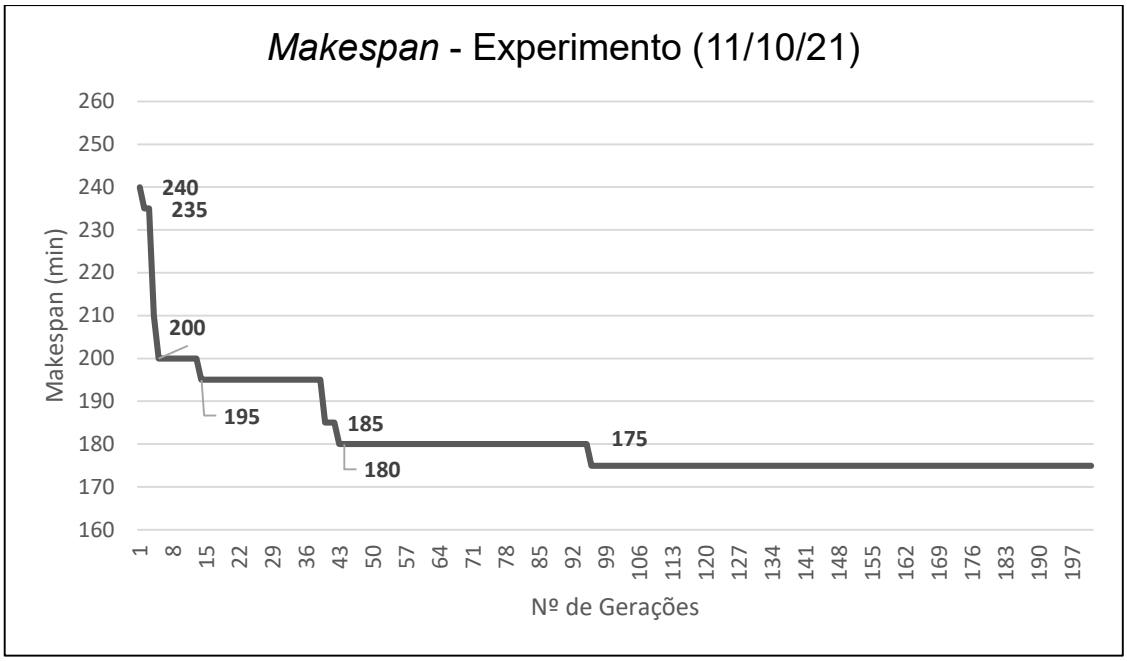
Fonte: Elaborada pelo próprio autor (2021).

Figura 25 – Detalhamento *Makespan* – experimento 09/10/2021



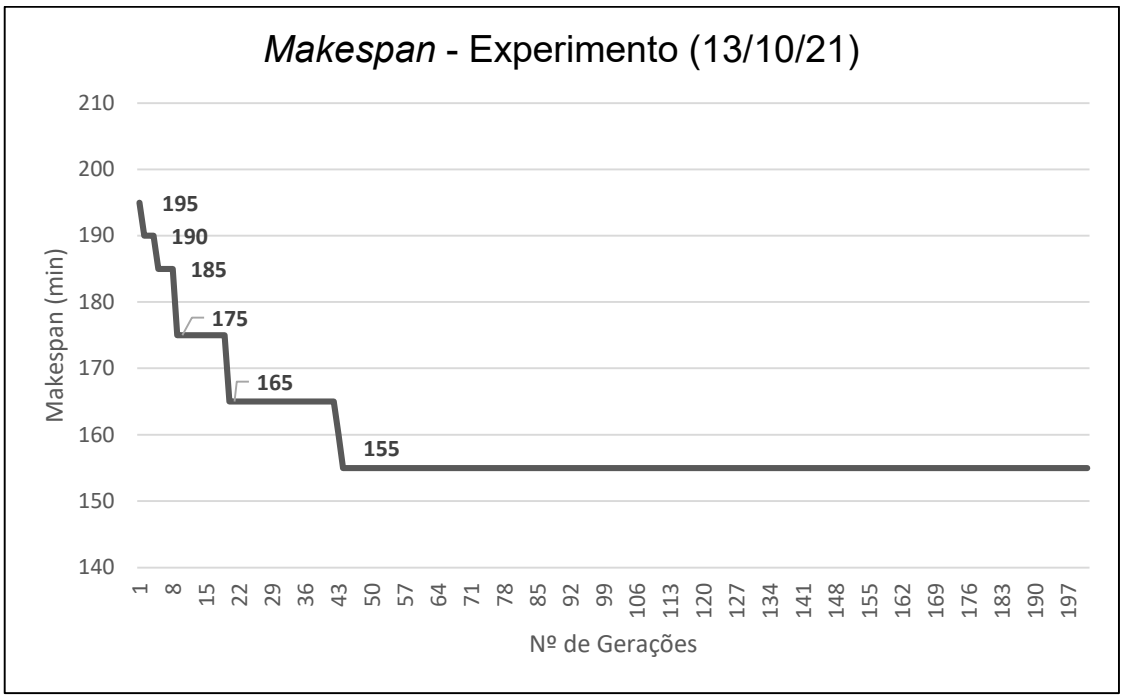
Fonte: Elaborada pelo próprio autor (2021).

Figura 26 – Detalhamento *Makespan* – experimento 11/10/2021



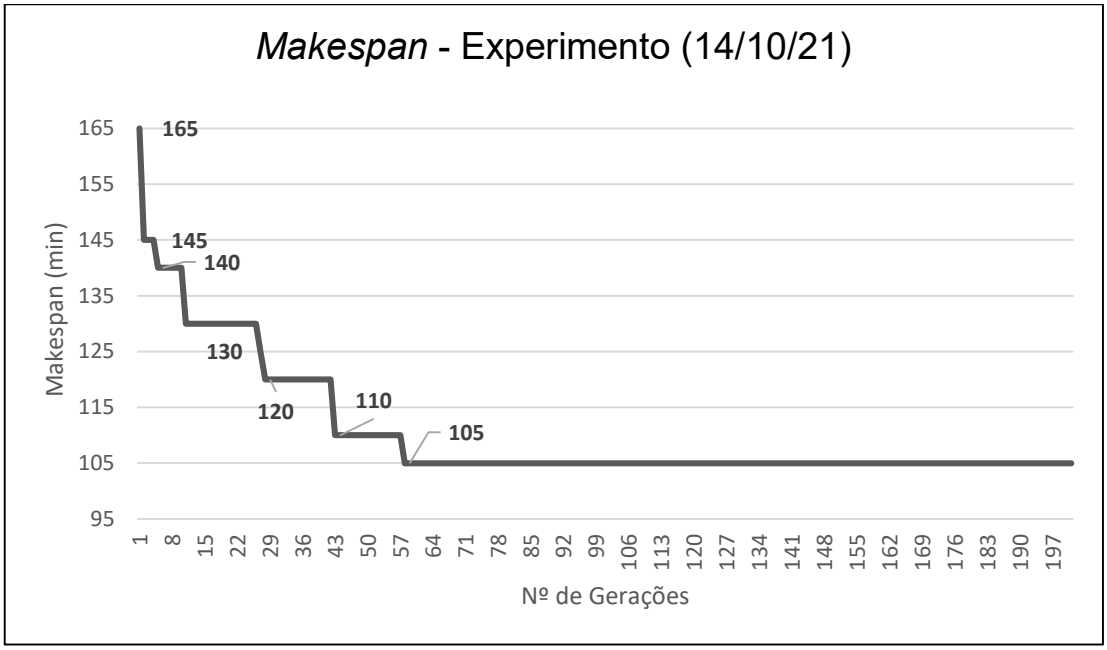
Fonte: Elaborada pelo próprio autor (2021).

Figura 27 – Detalhamento Makespan – experimento 13/10/2021



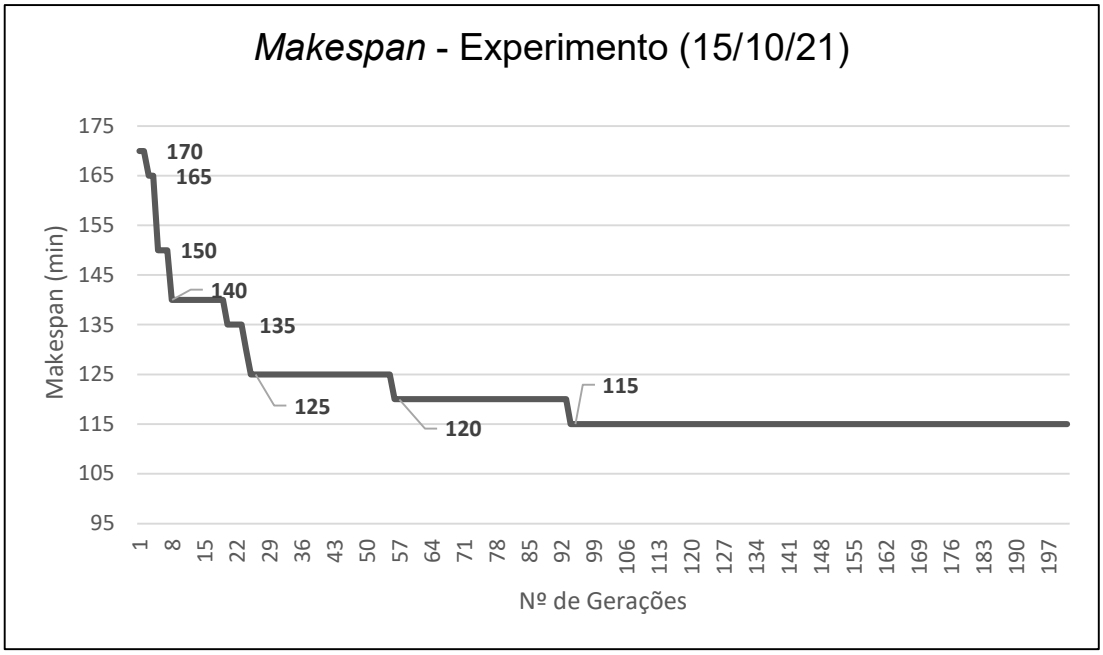
Fonte: Elaborada pelo próprio autor (2021).

Figura 28 – Detalhamento *Makespan* – experimento 14/10/2021



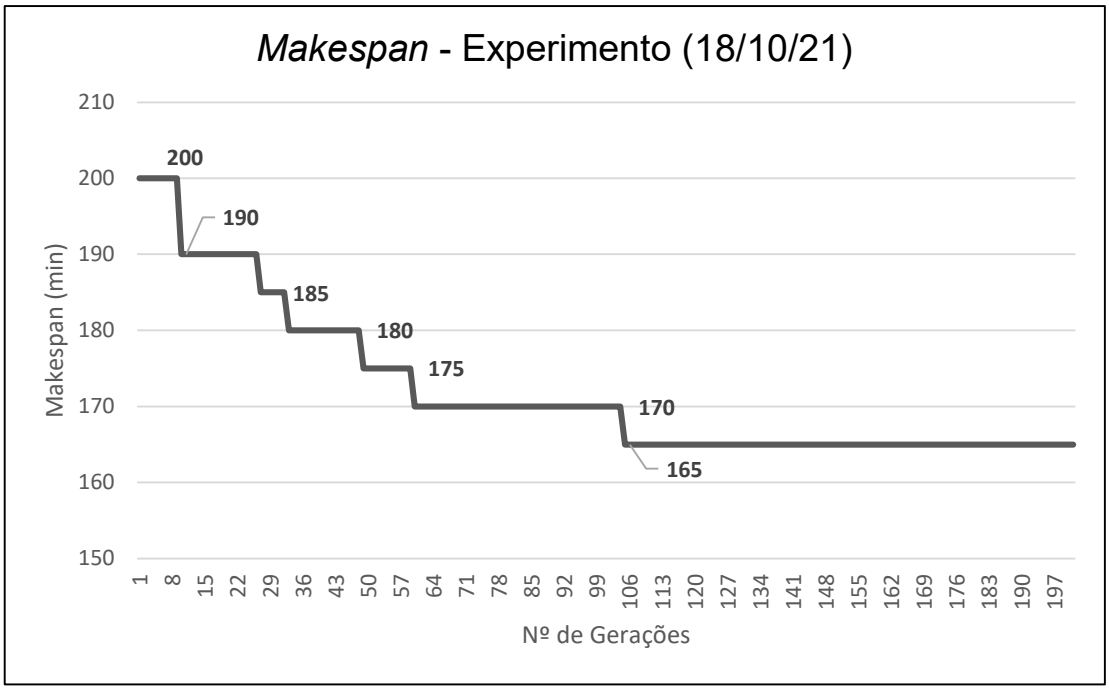
Fonte: Elaborada pelo próprio autor (2021).

Figura 29 – Detalhamento *Makespan* – experimento 15/10/2021



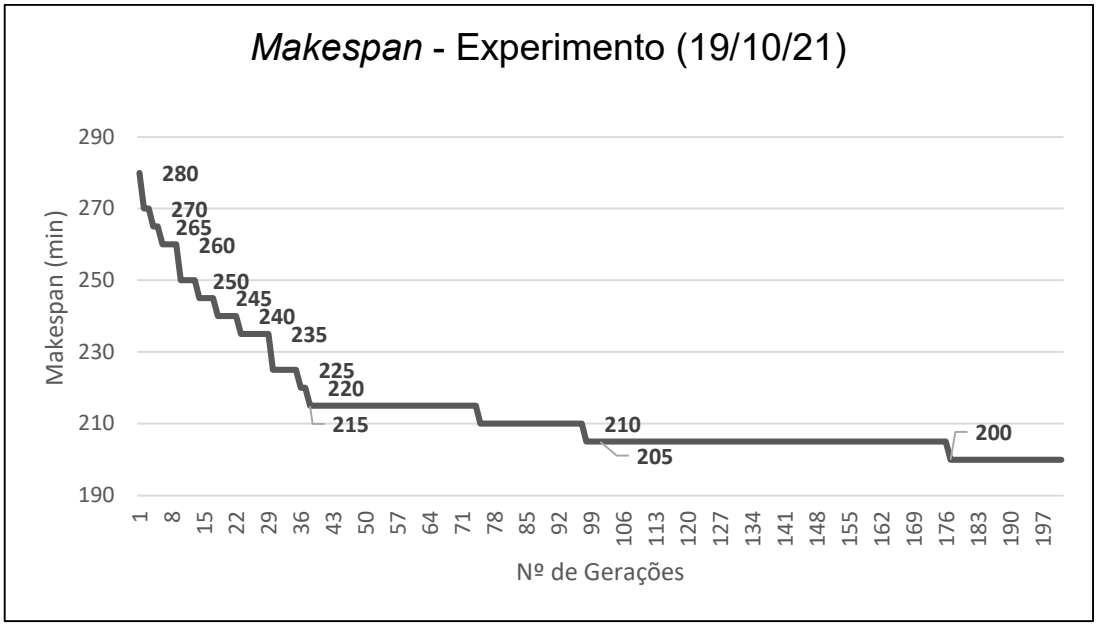
Fonte: Elaborada pelo próprio autor (2021).

Figura 30 – Detalhamento *Makespan* – experimento 18/10/2021



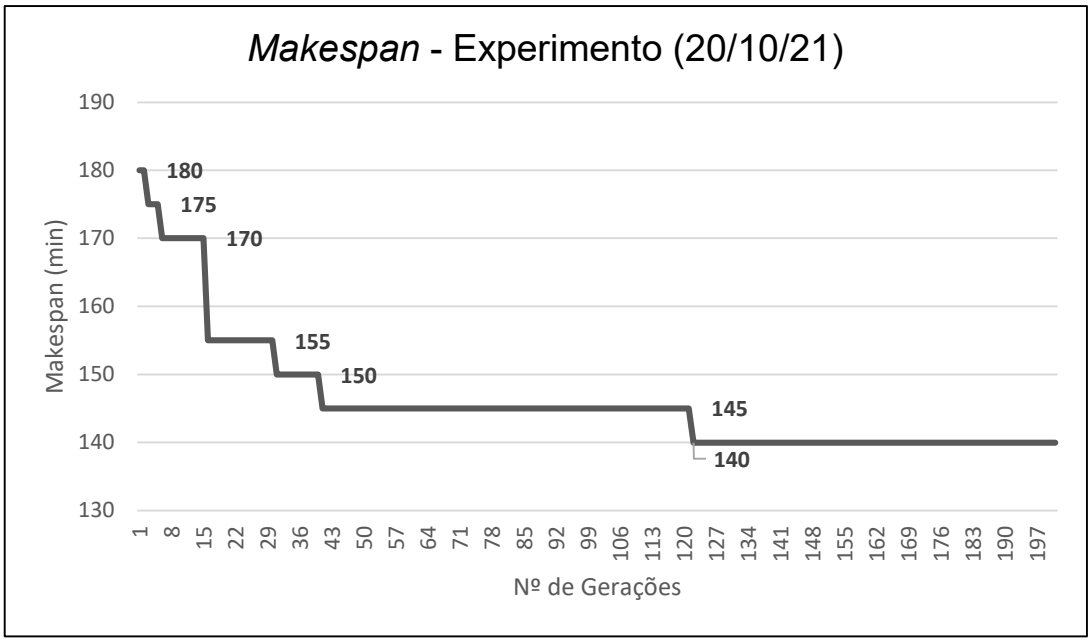
Fonte: Elaborada pelo próprio autor (2021).

Figura 31 – Detalhamento Makespan – experimento 19/10/2021



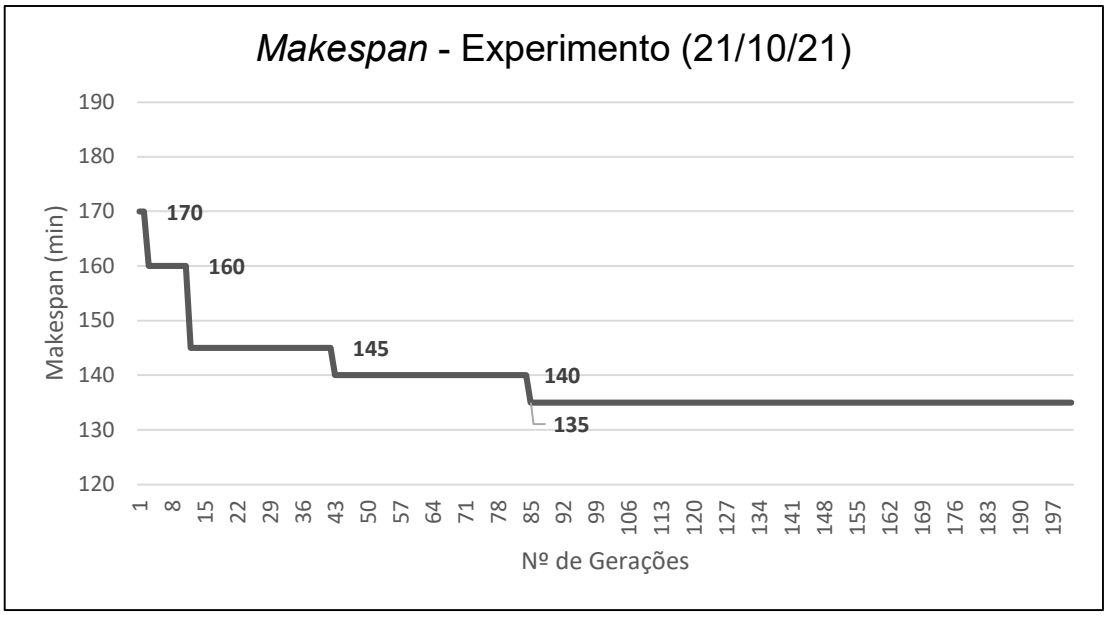
Fonte: Elaborada pelo próprio autor (2021).

Figura 32- Detalhamento *Makespan* – experimento 20/10/2021



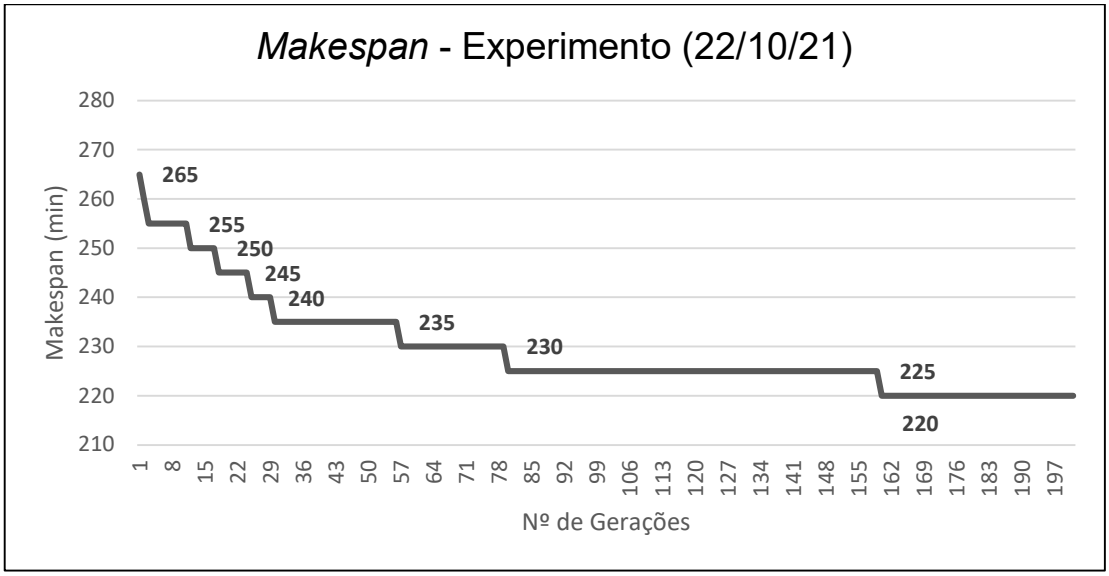
Fonte: Elaborada pelo próprio autor (2021).

Figura 33 – Detalhamento *Makespan* – experimento 21/10/2021



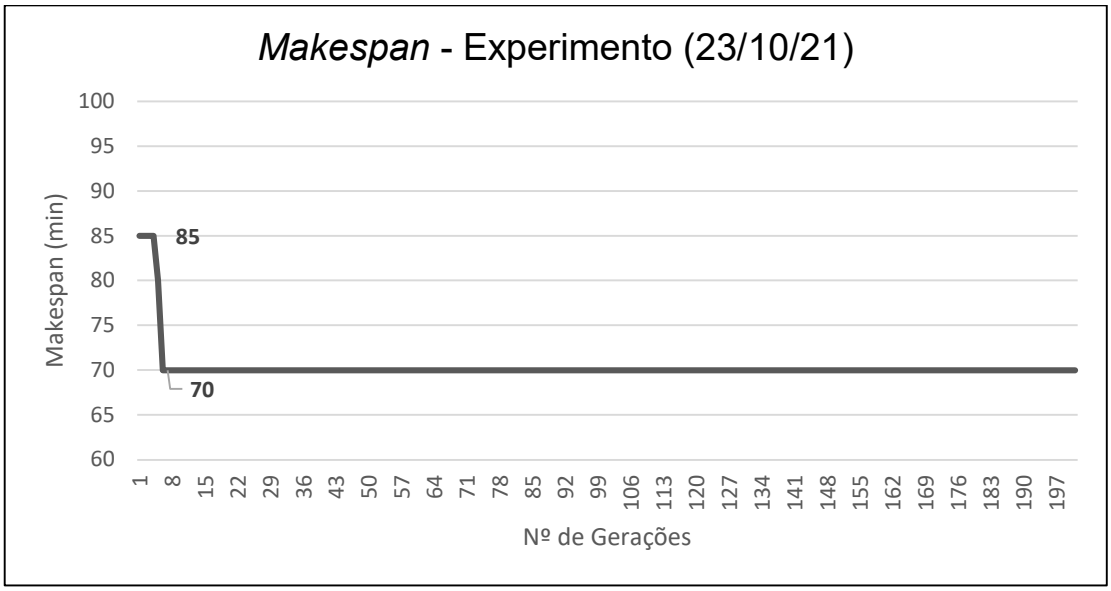
Fonte: Elaborada pelo próprio autor (2021).

Figura 34 – Detalhamento *Makespan* – experimento 22/10/2021



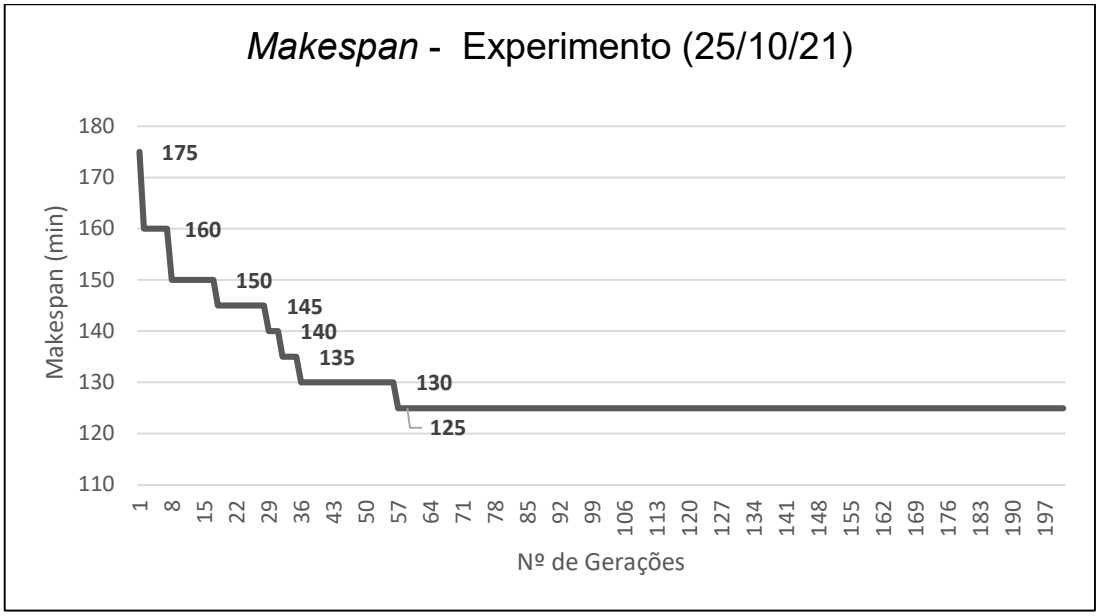
Fonte: Elaborada pelo próprio autor (2021).

Figura 35 – Detalhamento *Makespan* – experimento 23/10/2021



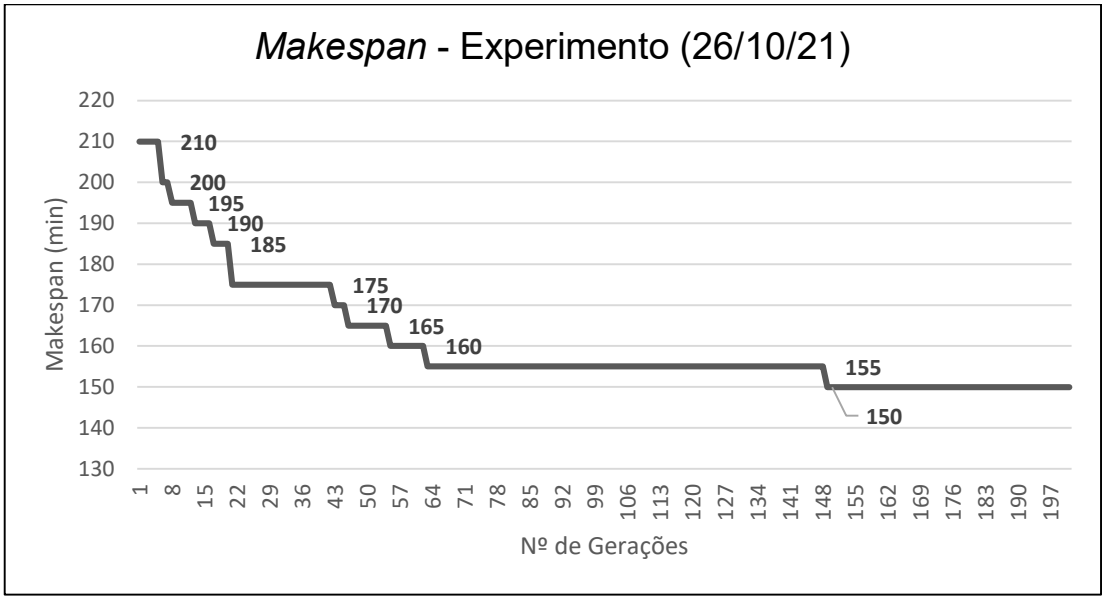
Fonte: Elaborada pelo próprio autor (2021).

Figura 36 – Detalhamento Makespan – experimento 25/10/2021



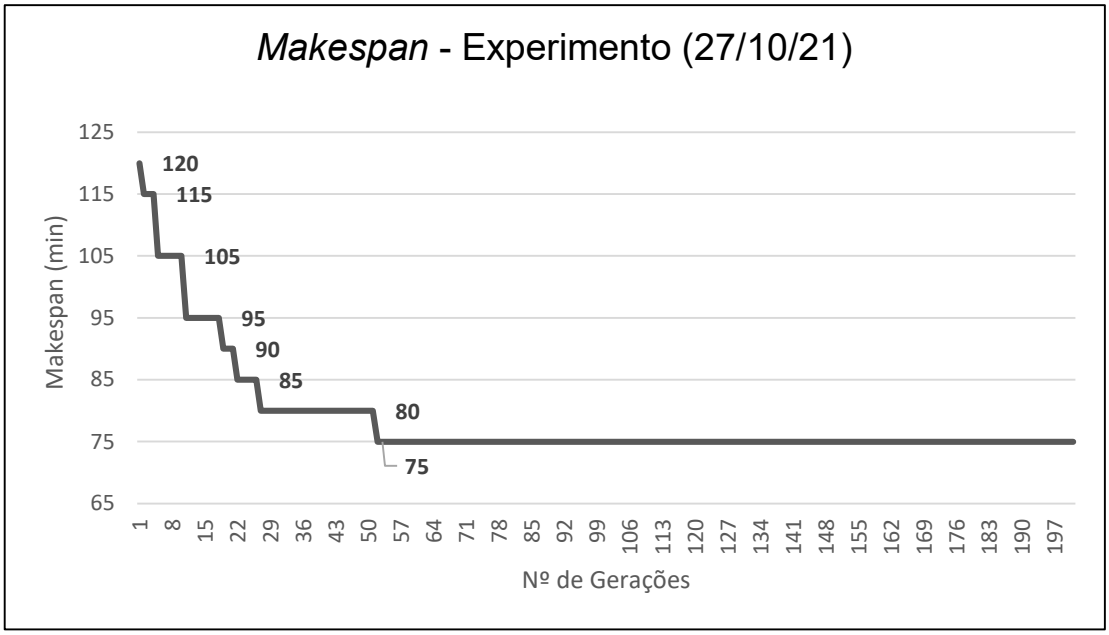
Fonte: Elaborada pelo próprio autor (2021).

Figura 37 – Detalhamento *Makespan* – experimento 26/10/2021



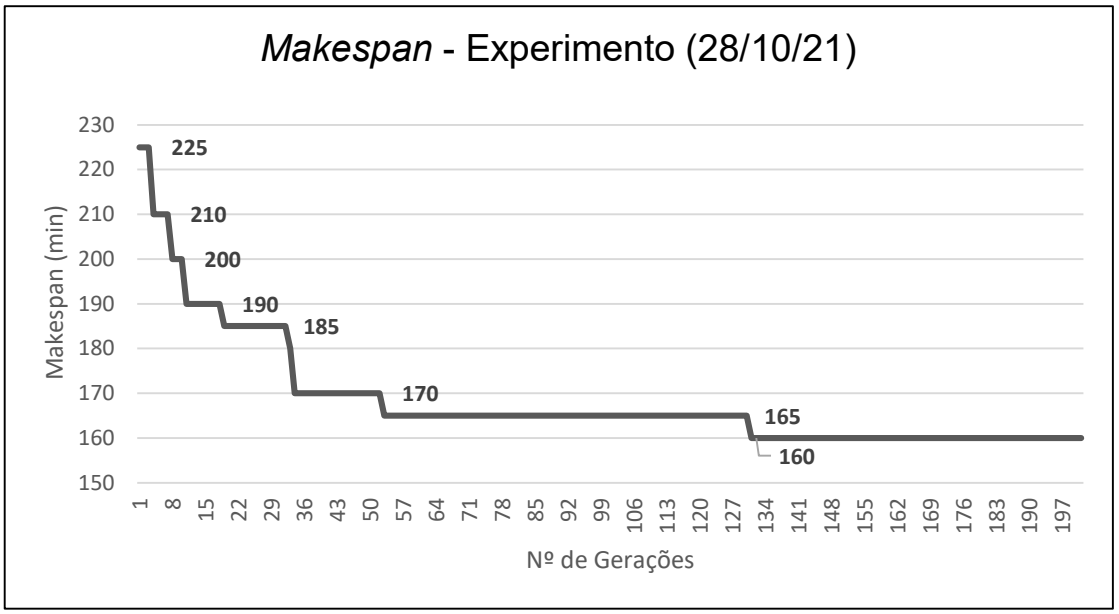
Fonte: Elaborada pelo próprio autor (2021).

Figura 38 – Detalhamento *Makespan* – experimento 27/10/2021



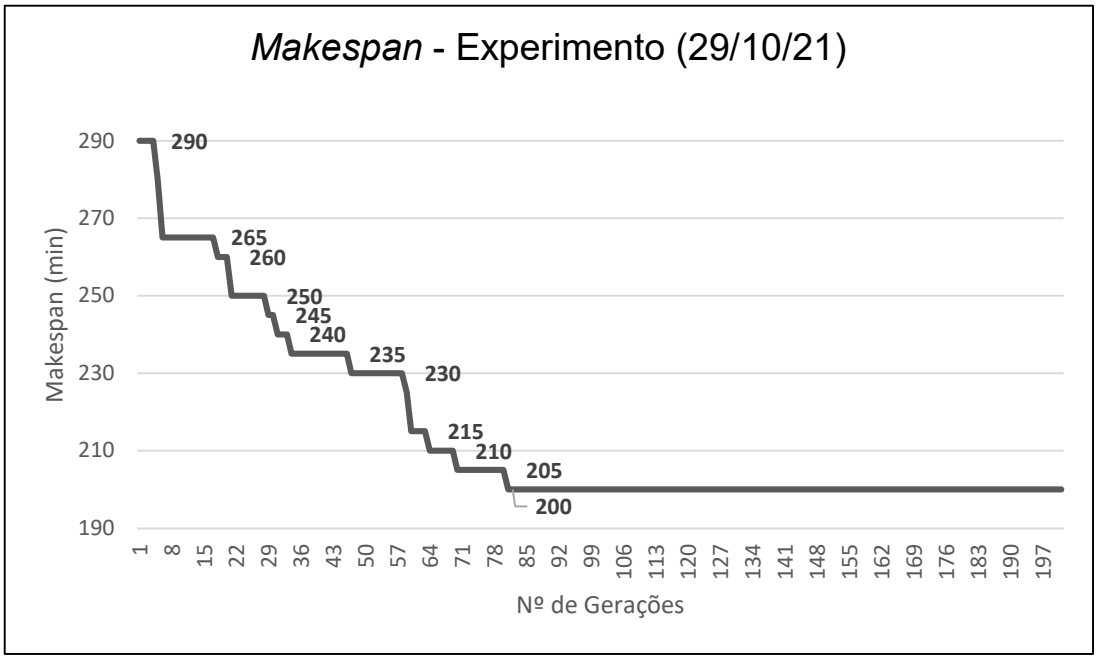
Fonte: Elaborada pelo próprio autor (2021).

Figura 39 – Detalhamento *Makespan* – experimento 28/10/2021



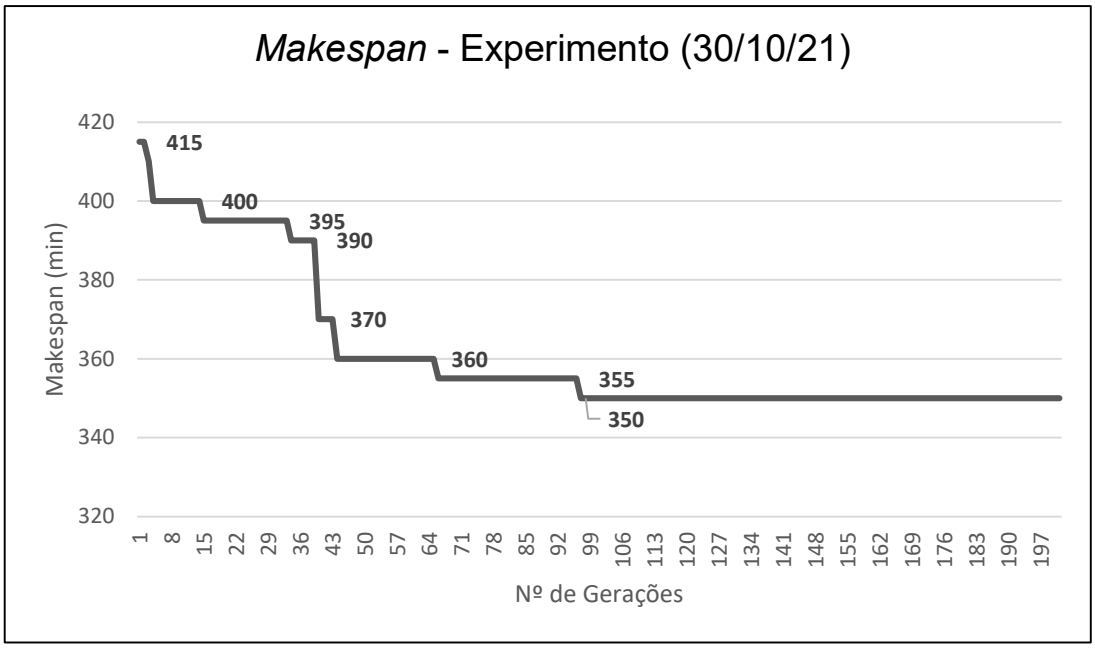
Fonte: Elaborada pelo próprio autor (2021).

Figura 40 – Detalhamento *Makespan* – experimento 29/10/2021



Fonte: Elaborada pelo próprio autor (2021).

Figura 41 – Detalhamento *Makespan* – experimento 30/10/2021



Fonte: Elaborada pelo próprio autor (2021).

5 DISCUSSÕES

No cenário atual (sem a aplicação do modelo) observa-se a variação no tempo de *makespan* em dias com a mesma quantidade de veículos na situação “em fila”, indicando possíveis casos de problemas pontuais ocorridos no processo de carregamento do terminal.

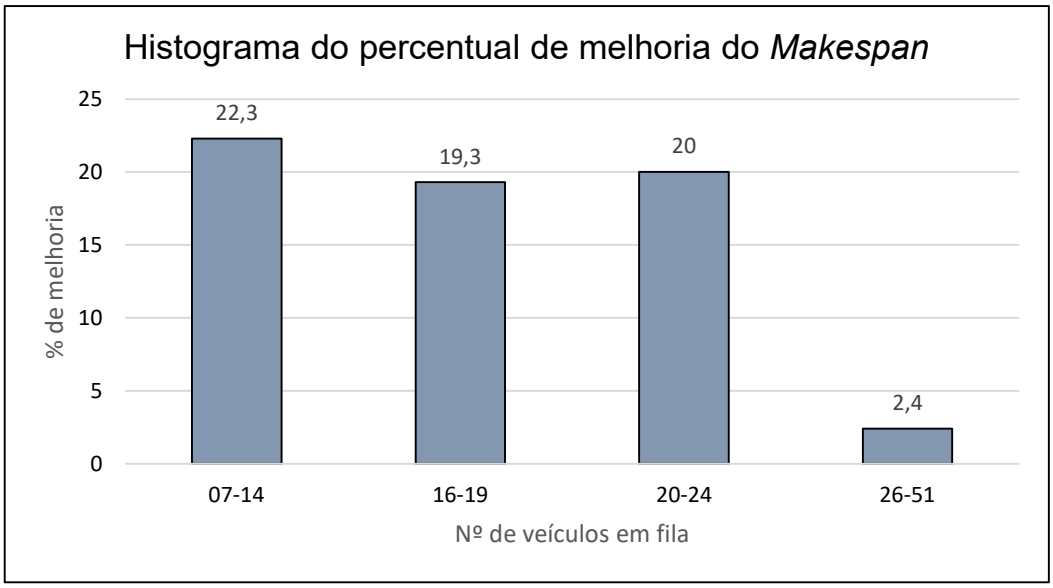
Essa variação corrobora com a afirmação do impacto gerado na capacidade de expedição da empresa pela ausência da utilização de métodos automatizados. Portanto, sugere-se uma investigação mais detalhada em relação aos dias com variação relevante.

Em relação ao modelo experimental, o resultado da otimização proposta se mostrou eficaz, no qual percebe-se pouca variação do *makespan* do modelo em função da quantidade de veículos em fila, demonstrando consistência na solução factível.

De modo geral, o sistema demonstrou também eficácia do ponto de vista de redução do *makespan* em relação ao cenário real, apresentando média de melhora do tempo em 16%. Ao todo foram feitas experimentações em 25 dias diferentes no mês de outubro de 2021 e em apenas 02 dias houve a constatação da redução da performance na experimentação. Em 01 dos 25 dias não foi possível a realização da experimentação devido a quantidade insuficiente de veículos no dia.

Um fator que chamou a atenção dos gestores da empresa foi que, percentualmente, as maiores melhorias do *makespan* foram encontrados nos dias com menos veículos em fila. Mesmo se for retirados os dias em que não houve melhoria, a média ainda ficaria abaixo dos 10% Ao contrário, era esperado que as maiores reduções fossem encontradas em períodos de maior demanda da expedição rodoviária do terminal. O histograma do percentual de melhoria do *makespan* é demonstrado na Figura 42.

Figura 42 - Histograma do percentual de melhoria do *Makespan*



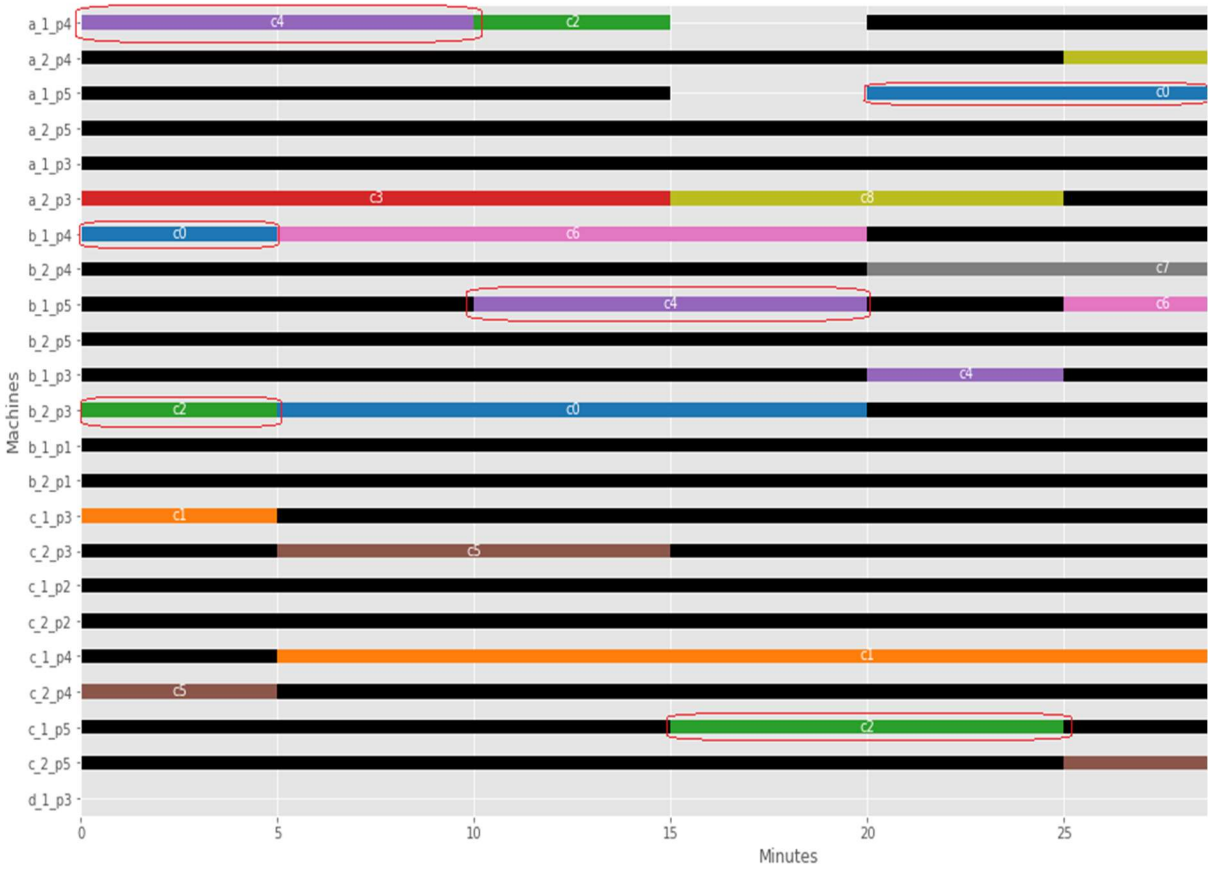
Fonte: Elaborada pelo próprio autor (2021).

Neste sentido, o potencial de melhoria do modelo para os indicadores de desempenho do terminal supera as expectativas, com potenciais de redução de tempo total da operação, aumento de capacidade operacional e melhoria dos níveis de atendimento tanto rodoviário quanto marítimo. Além disso, há um otimismo em relação ao equilíbrio das variações de *makespan* em função do número de veículos em fila.

Vale ressaltar também que, para garantir maior fidelidade das soluções geradas em relação à realidade do terminal, sugere-se a avaliação adicional futura de um número maior de aspectos do modelo e a construção de regras mais detalhadas para o modelo de otimização. Como exemplos, é a inclusão da análise em relação ao tempo de atraso total de atendimento dos veículos, inclusão do tempo de inatividade de cada máquina em relação aos veículos em carregamento e a estruturação de tempos para troca de máquinas entre baias, conforme elementos marcados na figura 43.

Nesta figura, é notado que os veículos circulados em vermelho mudam de baias com frequência, fato que acaba sendo prejudicial para a operação devido aos excessos de manobras dos veículos.

Figura 43 – Solução factível



Fonte: Elaborada pelo próprio autor (2021).

6 CONCLUSÃO

No presente estudo, foi proposto um modelo de otimização para investigação da melhoria da performance das operações de um terminal marítimo de granéis líquidos situado na região do porto de Vitória/ES, através do emprego de um algoritmo genético para solução de um problema do tipo *Job Shop Scheduling*. Para este fim, foi desenvolvido um modelo experimental que poderá ser implementado no setor operacional da empresa, considerando a adequação de alguns recursos como a comunicação com a atual automação do terminal.

Foi demonstrado a melhoria dos indicadores operacionais a partir da experimentação do modelo no período de 1 mês operacional, com resultados expressivos mesmo nas situações com baixa quantidade de veículos em fila. Adicionalmente percebeu-se uma diminuição na variação do *makespan* operacional na utilização do modelo.

O método desenvolvido neste trabalho está focado na melhoria operacional da empresa em questão, entretanto, serve como modelo na aplicação de outros terminais que utilizam o mesmo modelo operacional apresentado.

Por fim, considerando os resultados encontrados, percebe-se que muitas são as oportunidades de melhorias no setor portuário e estas são uma necessidade face ao contínuo crescimento do setor, predominante na extensa costa brasileira. E que há muita variação dos métodos aplicados e aos problemas tratados conforme demonstrado na análise sistemática. Dessa forma, avaliações mais detalhadas dos ganhos operacionais e um maior incremento na robustez dos modelos são oportunidades futuras para melhoria do estudo.

REFERÊNCIAS

ANTAQ. **Anuário Estatístico Aquaviário - 2020**: Apresentação, 2021.

IBGE. **Atlas geográfico das zonas costeiras e oceânicas do Brasil**. Rio de Janeiro, 2011.

BALLOU, R. H. **Gerenciamento da cadeia de suprimentos/logística empresarial**. 5 ed., Porto Alegre: Bookman, 2007.

BISH, E. K. **A multiple-crane-constrained scheduling problem in a container terminal**. European Journal of Operational Research, v. 144, n. 1, p. 83–107, 2003.

CANKAYA, B.; WARI, E.; EREN TOKGOZ, B. **Practical approaches to chemical tanker scheduling in ports:** a case study on the Port of Houston. Maritime Economics & Logistics, v. 21, n. 4, p. 559–575, 2019.

CNT. **O transporte move o Brasil:** resumo das propostas da CNT ao país. 2019.

COSTA, T. **Introdução ao Shipping**. 2021.

DIK, G.; KOZAN, E. **A flexible crane scheduling methodology for container terminals**. **Flexible Services and Manufacturing Journal**, v. 29, n. 1, p. 64–96, 2017.

DUDIN, A. N.; KLIMENOK, V. I.; VISHNEVSKY, V. M. **The Theory of Queuing Systems with Correlated Flows**. Cham: Springer International Publishing, 2020.

EILKEN, A. **A decomposition-based approach to the scheduling of identical automated yard cranes at container terminals**. Journal of Scheduling, v. 22, n. 5, p. 517–541, out. 2019.

FIBRIANTO, H. Y.; KANG, B.; HONG, S. **A Job Sequencing Problem of an Overhead Shuttle Crane in a Rail-Based Automated Container Terminal.** IEEE Access, v. 8, p. 156362–156377, 2020.

GAO, Z. et al. **Ship-unloading scheduling optimization for a steel plant.** Information Sciences, v. 544, p. 214–226, jan. 2021.

GIL, A. C. **Como elaborar projetos de pesquisa**. São Paulo: Atlas, 2009.

HANDABAKA, A. R. **Gestão Logística da Distribuição Física Internacional**. Maltese, 1994.

HILLIER, F. S.; LIEBERMAN, G. J. **Introdução à pesquisa operacional**. São Paulo: McGraw-Hill, 2006.

HUANG, J.; WANG, F.; SHI, N. **Resource Allocation Problems in Port Operations:** A Literature Review, 2014

HUANG, S. Y.; LI, Y. **A Cooperative Approach to Dispatching and Scheduling Twin-Yard Cranes in Container Terminals**. Computational Logistics. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016.

IBGE. **Anuário estatístico do Brasil**. Rio de Janeiro: 2018

JAEHN, F.; KRESS, D. Scheduling cooperative gantry cranes with seaside and landside jobs. **Discrete Applied Mathematics**, v. 242, p. 53–68, jun. 2018.

JONKER, T. et al. Coordinated optimization of equipment operations in a container terminal. **Flexible Services and Manufacturing Journal**, v. 33, n. 2, p. 281–311, jun. 2021.

KUHPFAHL, J. **Job Shop Scheduling with Consideration of Due Dates**. Wiesbaden: Springer Fachmedien Wiesbaden, 2016.

LAKATOS, E. M.; MARCONI, M. DE A. **Metodologia do trabalho científico:** procedimentos básicos, pesquisa bibliográfica, projeto e relatório, publicações e trabalhos científicos. São Paulo: Atlas, 1995.

LINDEN, R. **Algoritmos Genéticos**. 03. ed., Ciência Moderna, 2012.

MINISTÉRIO DA INFRAESTRUTURA. **BR do Mar**, 2020. Disponível em: <https://www.gov.br/infraestrutura/pt-br/brdomar>

NGUYEN, D.; YUN, W. Y. Optimal Job Scheduling of a Rail Crane in a Rail Terminal. p. 129–140, 2014.

PINEDO, M. L. **Planning and Scheduling in Manufacturing and Services**. New York, NY: Springer New York, 2009.

VERGARA, S. C. **Projetos e relatórios de pesquisa em administração**. Rio de Janeiro: Grupo Gen - Atlas, 2016.

XU, D.; LI, C. L.; LEUNG, J. Y. T. **Berth allocation with time-dependent physical limitations on vessels**. European Journal of Operational Research, v. 216, n. 1, p. 47–56, 2012.

YANG, W. et al. **Real-Time Production and Logistics Self-Adaption Scheduling Based on Information Entropy Theory**. Sensors, v. 20, n. 16, p. 4507, 2020.

ANEXO A – CÓDIGO

```
# Created by: Prof. Valdecy Pereira, D.Sc.
# UFF - Universidade Federal Fluminense (Brazil)
# email:  valdecy.pereira@gmail.com

# Required Libraries
import copy
import itertools
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import random
import time as tm

# Helper Function: Variable Names
def variablename(var):
    return [tpl[0] for tpl in filter(lambda x: var is x[1], globals().items())]

# Helper Function: Plot Gantt Chart
def gantt_chart(gantt_matrix, max_makespan, machines_list, vehicles_list):
    plt.style.use('ggplot')
    fig, ax     = plt.subplots(figsize = (20, 10))
    yticks      = list(range(5, len(machines_list)*10, 10))
    ylabels     = [variablename(x)[0] for x in machines_list]
    height      = 4
    cycol       = itertools.cycle(['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
'#e377c2', '#7f7f7f', '#bcbd22', '#17becf', '#bf77f6', '#ff9408', '#d1ffbd', '#c85a53', '#3a18b1', '#ff796c',
'#04d8b2', '#ffb07c', '#aaa662', '#0485d1', '#fffe7a', '#b0dd16', '#85679', '#12e193', '#82cafc', '#ac9362',
'#f8481c', '#c292a1', '#c0fa8b', '#ca7b80', '#f4d054', '#fbdd7e', '#ffff7e', '#cd7584', '#f9bc08', '#c7c10c'])
    keys_c      = [variablename(x)[0] for x in vehicles_list]
    colors_dict = dict(zip(keys_c, cycol))
    ax.set_yticks(yticks)
    ax.set_ylabel('Machines')
    ax.set_ylim(0, len(machines_list)*10)
    ax.set_yticklabels(ylabels)
    ax.set_xlim(0, max_makespan + 1)
    ax.set_xlabel('Minutes')
    ax.grid(True)
```

```
            for i in range(0, len(gantt_matrix)):
                tasks = [x for x in list(dict.fromkeys(gantt_matrix[i])) if type(x) == str]
                for j in range(0, len(tasks)):
                    if ('x' not in tasks[j]):
                        idxes = [a for a, b in enumerate(gantt_matrix[i]) if b == tasks[j]]
                        idx_s = idxes[ 0]
                        idx_e = idxes[-1]
                        dur   = idx_e - idx_s + 1
                        ax.broken_barh( [(idx_s, dur) ], (yticks[i] - height/2, height), facecolors =
colors_dict[tasks[j]])
                        ax.text(x = idx_e - (dur - 2)/2, y = yticks[i], s = tasks[j], ha = 'center', va = 'center', color
= 'white')
                    else:
                        idxes = [a for a, b in enumerate(gantt_matrix[i]) if b == tasks[j]]
                        idx_s = idxes[ 0]
                        idx_e = idxes[-1]
                        dur   = idx_e - idx_s + 1
                        ax.broken_barh( [(idx_s, dur) ], (yticks[i] - height/2, height), facecolors = 'black')
            plt.gca().invert_yaxis()
            return

        # Helper Function: Solution Report
        def show_report(gantt_matrix, machines_list):
            column_names = ['Machine', 'Vehicle', 'Start (min)', 'End (min)', 'Duration (min)']
            machines     = [variablename(x)[0] for x in machines_list]
            rprt_mch     = []
            for i in range(0, len(gantt_matrix)):
                tasks = [x for x in list(dict.fromkeys(gantt_matrix[i])) if type(x) == str]
                for j in range(0, len(tasks)):
                    if ('x' not in tasks[j]):
                        idxes = [a for a, b in enumerate(gantt_matrix[i]) if b == tasks[j]]
                        idx_s = idxes[ 0]
                        idx_e = idxes[-1] + 1
                        dur   = idx_e - idx_s
                        rprt_mch.append([machines[i], tasks[j], idx_s, idx_e, dur])
            report_df = pd.DataFrame(rprt_mch, columns = column_names)
            return report_df

        ##########################################################################
#
```

```
        # Function: Target Function
        def target_function(individual, machines_list, machines_int, twins_int, vehicles_dict, locations_dict, arrival_dict, pnlt_1):
            penalty_1    = 0
            makespan     = []
            locations    = []
            gantt_matrix = [ [0]*24*60 for i in range(0, len(machines_list)) ]
            for k in range(0, len(individual)):
                location = []
                for i in range(0, len(individual[k][2])):
                    time = individual[k][2][i]
                    mch  = individual[k][1][i]
                    tws  = twins_int[mch]
                    pkl  = locations_dict[machines_int[mch][:3]]
                    pkl  = [item for item in pkl if item not in [mch]]
                    fit  = True
                    idx  = 0
                    if (arrival_dict[individual[k][0][0]] > 0):
                        idx = arrival_dict[individual[k][0][0]]
                    while (fit):
                        for _ in range(idx, idx + time):
                            if (gantt_matrix[mch][idx] != 0):
                                fit = True
                                idx = idx + 1
                            else:
                                fit = False
                                for z in range(idx, idx + time):
                                    if ([b[z] for a, b in enumerate(gantt_matrix)].count(vehicles_dict[individual[k][0][0]]) > 0):
                                        fit = True
                                        idx = idx + 1
                                        break
                                    else:
                                        fit = False
                    gantt_matrix[mch][idx:idx+time] = [vehicles_dict[individual[k][0][0]]]*time
                    if (tws != mch):
                        gantt_matrix[tws][idx:idx+time] = ['x']*time
                    for p in range(0, len(pkl)):
                        for q in range(idx, idx + time):
```

```
                if(gantt_matrix[pkl[p]][q] == 0):
                    gantt_matrix[pkl[p]][q] = 'x'
            idx = idx + time
            location.append(machines_int[mch][:3])
            makespan.append(idx)
        locations.append(location)
        if (location.count(location[0]) != len(location)):
            penalty_1 = penalty_1 + 1
    cost = max(makespan) + penalty_1*pnlt_1
    return gantt_matrix, max(makespan), penalty_1, cost

# Function: Initial Population
def initial_population(vehicles_list, products, machines_dict, size):
    vehicles_list_shuffle = copy.deepcopy(vehicles_list)
    population            = []
    for _ in range(0, size):
        random.shuffle(vehicles_list_shuffle)
        individual = [ [ [], [], [] ] for i in range(0, len(vehicles_list))] # [Vehicle], [Machines], [Demand]
        for i in range(0, len(vehicles_list_shuffle)):
            individual[i][0].append(vehicles_list_shuffle[i][0][0])
            for j in range(0, len(vehicles_list_shuffle[i][1])):
                if (vehicles_list_shuffle[i][1][j] > 0):
                    mch = random.choice(products[j])
                    individual[i][2].append(vehicles_list_shuffle[i][1][j])
                    individual[i][1].append(machines_dict[mch])
        population.append(individual)
    return population

# Function: Fitness
def fitness_function(cost):
    population_size = len(cost)
    fitness         = np.zeros((population_size, 2))
    for i in range(0, fitness.shape[0]):
        fitness[i,0] = 1/(1 + cost[i] + abs(np.min(cost)))
    fit_sum      = fitness[:,0].sum()
    fitness[0,1] = fitness[0,0]
    for i in range(1, fitness.shape[0]):
        fitness[i,1] = (fitness[i,0] + fitness[i-1,1])
    for i in range(0, fitness.shape[0]):
        fitness[i,1] = fitness[i,1]/fit_sum
```

```
    return fitness

# Function: Selection
def roulette_wheel(fitness):
    ix     = 0
    random = int.from_bytes(os.urandom(8), byteorder = 'big') / ((1 << 64) - 1)
    for i in range(0, fitness.shape[0]):
        if (random <= fitness[i, 1]):
          ix = i
          break
    return ix

# Function: Crossover
def crossover(parent_1, parent_2):
    offspring  = []
    A          = 1
    B          = len(parent_1) - 1
    if (B == 0):
        B = 2
    cut        = random.choice(list(range(A, B)))
    k_parent_1 = random.sample(list(range(0, len(parent_1)-1)), cut)
    k_parent_2 = [item for item in list(range(0, len(parent_1))) if item not in k_parent_1]
    for j in range(0, len(k_parent_1)):
        for i in range(0, len(parent_1)):
            if (k_parent_1[j] == parent_1[i][0][0]):
                offspring.append(parent_1[i])
    for k in range(0, len(k_parent_2)):
        for i in range(0, len(parent_2)):
            if (k_parent_2[k] == parent_2[i][0][0]):
                offspring.append(parent_2[i])
    return offspring

# Function: Breeding and Improvement
def breeding(cost, population, elite, fitness):
    offspring = copy.deepcopy(population)
    if (elite > 0):
        cost, population = (list(t) for t in zip(*sorted(zip(cost, population))))
        for i in range(0, elite):
            offspring[i] = copy.deepcopy(population[i])
    for i in range (elite, len(offspring)):
```

```
        parent_1, parent_2 = roulette_wheel(fitness), roulette_wheel(fitness)
        while parent_1 == parent_2:
            parent_2 = random.sample(range(0, len(population) - 1), 1)[0]
        parent_1 = copy.deepcopy(population[parent_1])
        parent_2 = copy.deepcopy(population[parent_2])
        rand     = int.from_bytes(os.urandom(8), byteorder = 'big') / ((1 << 64) - 1)
        if (rand > 0.5):
            offspring[i] = crossover(parent_1, parent_2)
        elif (rand <= 0.5):
            offspring[i] = crossover(parent_2, parent_1)
    return offspring

# Function: Mutation - Reverse Machines Order in a Vehicle
def mutation_rvs_machines(individual):
    machines_idx = []
    for i in range(0, len(individual)):
        if (len(individual[i][1]) > 1):
            machines_idx.append(i)
    idx = random.choice(machines_idx)
    k   = random.sample(list(range(0, len(individual[idx][1]))), 2)
    k1  = min(k)
    k2  = max(k) + 1
    r   = individual[idx][1][k1:k2]
    r.reverse()
    individual[idx][1][k1:k2] = r
    return individual

# Function: Mutation - Swap Machines in a Vehicle
def mutation_swp_machines(individual):
    machines_idx = []
    for i in range(0, len(individual)):
        if (len(individual[i][1]) > 1):
            machines_idx.append(i)
    idx = random.choice(machines_idx)
    k   = random.sample(list(range(0, len(individual[idx][1]))), 2)
    k1  = k[0]
    k2  = k[1]
    A   = individual[idx][1][k1]
    B   = individual[idx][1][k2]
    individual[idx][1][k1] = B
```

```
    individual[idx][1][k2] = A
    return individual

# Function: Mutation - Reverse Vehicles Order in an Individual
def mutation_rvs_vehicles(individual):
    k   = random.sample(list(range(0, len(individual))), 2)
    k1  = min(k)
    k2  = max(k) + 1
    r   = individual[k1:k2]
    r.reverse()
    individual[k1:k2] = r
    return individual

# Function: Mutation - Swap Vehicles in an Individual
def mutation_swp_vehicles(individual):
    k   = random.sample(list(range(0, len(individual))), 2)
    k1  = k[0]
    k2  = k[1]
    A   = individual[k1]
    B   = individual[k2]
    individual[k1] = B
    individual[k2] = A
    return individual

# Function: Mutation
def mutation(offspring, mutation_rate, elite):
    for i in range(elite, len(offspring)):
        probability = int.from_bytes(os.urandom(8), byteorder = 'big') / ((1 << 64) - 1)
        if (probability <= mutation_rate):
            rand1 = int.from_bytes(os.urandom(8), byteorder = 'big') / ((1 << 64) - 1)
            if (rand1 <= 0.5): # Machines
                rand2 = int.from_bytes(os.urandom(8), byteorder = 'big') / ((1 << 64) - 1)
                if (rand2 <= 0.5):
                    offspring[i] = mutation_rvs_machines(offspring[i])
                elif(rand2 > 0.5):
                    offspring[i] = mutation_swp_machines(offspring[i])
            elif(rand1 > 0.5): # Vehicles
                rand3 = int.from_bytes(os.urandom(8), byteorder = 'big') / ((1 << 64) - 1)
                if (rand3 <= 0.5):
                    offspring[i] = mutation_rvs_vehicles(offspring[i])
```

```
                elif(rand3 > 0.5):
                    offspring[i] = mutation_swp_vehicles(offspring[i])
            return offspring

        ###############################################################################
#

        # GA-Scheduling Function
        def genetic_algorithm_scheduling(vehicles_list, machines_list, n_products, mutation_rate,
generations, elite, size, pnlt_1, graph = False):
            start          = tm.time()
            keys_v         = list(range(0, len(vehicles_list)))
            values_v       = [variablename(x)[0] for x in vehicles_list]
            vehicles_dict  = dict(zip(keys_v, values_v))
            keys_a         = list(range(0, len(vehicles_list)))
            values_a       = [vehicle[2][0] for vehicle in vehicles_list]
            arrival_dict   = dict(zip(keys_a, values_a))
            keys_m         = [variablename(x)[0] for x in machines_list]
            values_m       = list(range(0, len(machines_list)))
            machines_dict  = dict(zip(keys_m, values_m))
            machines_int   = dict(zip(values_m, keys_m))
            keys_tw        = list(range(0, len(machines_list)))
            values_tw      = [machines_dict[machine[0][0]] for machine in machines_list]
            twins_int      = dict(zip(keys_tw, values_tw))
            keys_L         = list(set([variablename(x)[0][:3] for x in machines_list]))
            values_L       = []
            locations      = [variablename(x)[0][:3] for x in machines_list]
            for i in range(0, len(keys_L)):
                values_L.append([j for j, x in enumerate(locations) if x == keys_L[i]])
            locations_dict = dict(zip(keys_L, values_L))
            count          = 0
            products       = []
            for i in range(0, n_products):
                strg = 'p' + str(i + 1)
                products.append([machine[0][0] for machine in machines_list if machine[1][0] == strg ])
            population     = initial_population(vehicles_list, products, machines_dict, size)
            cost         = [target_function(individual, machines_list, machines_int, twins_int, vehicles_dict,
locations_dict, arrival_dict, pnlt_1)[-1] for individual in population]
            fitness        = fitness_function(cost)
            cost, population = (list(t) for t in zip(*sorted(zip(cost, population))))
```

```
            elite_ind       = cost[0]
            solution        = population[0]
            print('Generation = ', count, ' f(x) = ', round(elite_ind, 2))
            while (count <= generations - 1):
               offspring       = breeding(cost, population, elite, fitness)
               offspring       = mutation(offspring, mutation_rate, elite)
               cost                    = [target_function(individual, machines_list, machines_int, twins_int,
vehicles_dict, locations_dict, arrival_dict, pnlt_1)[-1] for individual in offspring]
               cost, population = (list(t) for t in zip(*sorted(zip(cost, offspring))))
               fitness         = fitness_function(cost)
               elite_child     = cost[0]
               if(elite_ind > elite_child):
                  elite_ind = elite_child
                  solution  = copy.deepcopy(population[0])
               count = count + 1
               print('Generation = ', count, ' f(x) = ', round(elite_ind, 2))
            gantt_matrix, max_makespan, _, _ = target_function(solution, machines_list, machines_int,
twins_int, vehicles_dict, locations_dict, arrival_dict, pnlt_1)
            if (graph == True):
               gantt_chart(gantt_matrix, max_makespan, machines_list, vehicles_list)
            end = tm.time()
            print('Algorithm Time: ', round((end - start), 2), ' seconds')
            return solution, max_makespan, gantt_matrix

         # Parameters - Vehicles =  [ ['Vehicle'], ['product_1', 'product_2', 'product_3', 'product_4',
'product_5'], [Arrival] ]
         c0 = [ [0], [ 0, 0, 15, 10, 0], [0] ]
         c1 = [ [1], [ 0, 0, 0, 25, 25], [0] ]
         c2 = [ [2], [ 0, 0, 10, 0, 10], [0] ]
         c3 = [ [3], [ 0, 0, 10, 5, 0], [0] ]
         c4 = [ [4], [ 0, 0, 20, 20, 5], [0] ]
         c5 = [ [5], [ 0, 0, 35, 0, 0], [0] ]

         # Parameters - Machines =  [ ['twin'], ['product type'] ] * p1 = Alcool; p2 = MGO; p3 = Gasolina;
p4 = S10; p5 = S500 *
         a_1_p4 = [ ['a_2_p4'], ['p4']  ]
         a_1_p5 = [ ['a_2_p5'], ['p5']  ]
         a_1_p3 = [ ['a_2_p3'], ['p3']  ]
         a_2_p4 = [ ['a_1_p4'], ['p4']  ]
         a_2_p5 = [ ['a_1_p5'], ['p5']  ]
```

```
a_2_p3 = [ ['a_1_p3'], ['p3']  ]

b_1_p4 = [ ['b_2_p4'], ['p4']  ]
b_1_p5 = [ ['b_2_p5'], ['p5']  ]
b_1_p3 = [ ['b_2_p3'], ['p3']  ]
b_1_p1 = [ ['b_2_p1'], ['p1']  ]
b_2_p4 = [ ['b_1_p4'], ['p4']  ]
b_2_p5 = [ ['b_1_p5'], ['p5']  ]
b_2_p3 = [ ['b_1_p3'], ['p3']  ]
b_2_p1 = [ ['b_1_p1'], ['p1']  ]

c_1_p3 = [ ['c_2_p3'], ['p3']  ]
c_1_p2 = [ ['c_2_p2'], ['p2']  ]
c_1_p4 = [ ['c_2_p4'], ['p4']  ]
c_1_p5 = [ ['c_2_p5'], ['p5']  ]
c_2_p3 = [ ['c_1_p3'], ['p3']  ]
c_2_p2 = [ ['c_1_p2'], ['p2']  ]
c_2_p4 = [ ['c_1_p4'], ['p4']  ]
c_2_p5 = [ ['c_1_p5'], ['p5']  ]

d_1_p3 = [ ['d_1_p3'], ['p3']  ] # No Twin

# Parameters - Lists
vehicles_list = [c0, c1, c2, c3, c4, c5, c6]
machines_list = [a_1_p4, a_2_p4, a_1_p5, a_2_p5, a_1_p3, a_2_p3,
                 b_1_p4, b_2_p4, b_1_p5, b_2_p5, b_1_p3, b_2_p3, b_1_p1, b_2_p1,
                 c_1_p3, c_2_p3, c_1_p2, c_2_p2, c_1_p4, c_2_p4, c_1_p5, c_2_p5,
                 d_1_p3]

# Solution - Run GA-Scheduling Function
solution, max_makespan, gantt_matrix = genetic_algorithm_scheduling(vehicles_list, machines_list, n_products = 5, mutation_rate = 0.05, generations = 3, elite = 1, size = 50, pnlt_1 = 10, graph = False)

# Solution - Plot
gantt_chart(gantt_matrix, max_makespan, machines_list, vehicles_list)

# Solution - Report
report = show_report(gantt_matrix, machines_list)
report.sort_values(by = ['Vehicle', 'Start (min)'])
```