UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

FERNANDA BIA PETEAM

Abordagens do Algoritmo Genético de Chaves Aleatórias Viciadas para o Problema de alocação em berços

Campinas

2024

Fernanda Bia Peteam

Abordagens do Algoritmo Genético de Chaves Aleatórias Viciadas para o Problema de alocação em berços

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutora em Matemática Aplicada.

Orientadora: Kelly Cristina Poldi

Este trabalho corresponde à versão final da Tese defendida pela aluna Fernanda Bia Peteam e orientada pela Profa. Dra. Kelly Cristina Poldi.

Campinas

2024

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

P441a

Peteam, Fernanda Bia, 1991-
Abordagens do algoritmo genético de chaves aleatórias viciadas para o problema de alocação em berços / Fernanda Bia Peteam. – Campinas, SP : [s.n.], 2024.

Orientador: Kelly Cristina Poldi.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Problema de alocação de berços. 2. Algoritmos genéticos. 3. Algoritmos de agrupamento. I. Poldi, Kelly Cristina, 1979-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações Complementares

Título em outro idioma: Approaches of the biased random key genetic algorithm for the berth allocation problem
Palavras-chave em inglês:
Berth allocation problem
Genetic algorithms
Clustering algorithms
Área de concentração: Matemática Aplicada
Titulação: Doutora em Matemática Aplicada
Banca examinadora:
Kelly Cristina Poldi [Orientador]
Glaucia Maria Bressan
Daniela Renata Cantane
Aurelio Ribeiro Leite de Oliveira
Antonio Carlos Moretti
Data de defesa: 08-04-2024
Programa de Pós-Graduação: Matemática Aplicada

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0009-0009-0864-5296
- Currículo Lattes do autor: http://lattes.cnpq.br/6971554354097151

Tese de Doutorado defendida em 08 de abril de 2024 e aprovada

pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). KELLY CRISTINA POLDI

Prof(a). Dr(a). GLAUCIA MARIA BRESSAN

Prof(a). Dr(a). DANIELA RENATA CANTANE

Prof(a). Dr(a). AURELIO RIBEIRO LEITE DE OLIVEIRA

Prof(a). Dr(a). ANTONIO CARLOS MORETTI

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

Agradecimentos

Agradeço à minha orientadora Kelly Cristina Poldi por acompanhar e auxiliar no meu desenvolvimento e no do presente trabalho.

À professor Márcia Aparecida Gomes Ruggiero por ter me acompanhado desde a época da graduação, sempre ensinando, guiando, orientando e oferendo suporte.

Ao professor Antônio Carlos Moretti por sempre estar aberto para discutir ideias, indicar materiais de leitura para meu desenvolvimento na área.

Agradeço aos professores Glaucia Bressan, Daniela Cantane, Aurelio Ribeiro Leite e Antonio Carlos Moretti pela presença na banca, contribuindo com discussões, questionamentos e sugestões que enriqueceram o presente trabalho.

Ao Israel Campiotti pelo companheirismo, ensinamentos, discussões de ideias e suporte no desenvolvimento da qualidade de programação.

Agradeço à Unicamp e ao IMECC pela infraestrutura oferecida.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

O presente trabalho foi realizado também com o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq), número do processo: 142415/2017-0.

Resumo

O presente trabalho considera o Problema de Alocação de Navios em Berços (PAB), que visa determinar a ordem, o local e o horário de atendimento dos navios quando chegam ao porto. O modelo utilizado como base neste trabalho é o proposto na literatura, que modela o PAB como um Problema de Roteamento de Veículos com Janela de Tempo (PRVJT), com o objetivo de minimizar o tempo total de permanência dos navios no porto. Nesta tese propomos uma adaptação do modelo (PRVJT-O) que passa a considerar o tempo de ociosidade dos berços, uma perspectiva de interesse do porto e não somente dos clientes (donos dos navios). Para resolver os PABs sem e com ociosidade propomos uma abordagem por Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA) com uma decodificação que define a ordem e o local de atracação (berços) de maneira independente, e uma implementação vetorizada buscando um melhor desempenho. Testamos diferentes estratégias para melhorar os resultados obtidos pelo BRKGA, como realizar múltiplas execuções do algoritmo e adotar a melhor solução, utilizar a ferramenta OPTUNA para busca de melhores hiper parâmetros e aplicar *clusterização* na escolha dos pais na fase do cruzamento para uma maior variabilidade nos herdeiros, mas mantendo a característica elitista da fase. Além disso, testamos a utilização das soluções obtidas pelo BRKGA como solução inicial para o *solver* CPLEX, ou seja, para a resolução exata do modelo, uma vez que as soluções encontradas pelo BRKGA proposto foram factíveis para o problema. Comparamos diferentes estratégias de resolução utilizando o BRKGA e o BRKGA como solução inicial para o *solver* CPLEX. Dos testes computacionais realizados é possível concluir que utilizar o resultado de uma única execução do BRKGA ou o CPLEX apenas com limite de tempo não são as melhores opções. Com relação ao tempo computacional e qualidade de solução, concluímos que, para os modelos sem e com ociosidade de berços, as estratégias de combinação com a clusterização e as múltiplas execuções em paralelo com parâmetros fixos foram as estratégia com melhores equilíbrio.

Palavras-chave: Problema de alocação de navios em berços; Algoritmo genético de chaves aleatórias viciadas; clusterização.

Abstract

This work considers the Berth Allocation Problem (BAP), to determine the order, location and service time of ships when they arrive at the port, with the base model used based on the Vehicle Routing Problem with Time (PRVJT), proposed in the literature, with the objective of minimizing the total time spent by ships in port. We proposed an adaptation of the model (PRVJT-O) to consider the idle times of berths, a perspective of interest to the port and not just the customers (ship owners). To solve BAPs without and with idleness, we proposed a Biased Random Key Genetic Algorithm (BRKGA) model with a decoding that defines the order and berthing location (berths) independently, and a vectorized implementation, aiming better performance. We tested different strategies to improve the results obtained by BRKGA, such as running multiple times and adopting the best solution, using the OPTUNA tool to search for better hyper parameters and applying clustering in the choice of parents in the crossing phase for greater variability in the heirs, but maintaining its the elitist characteristic. Furthermore, we tested the use of the solutions obtained by BRKGA as an initial solution for CPLEX, that is, for the exact resolution of the model, since the solutions found by the proposed BRKGA were feasible for the problem. We were able to compare different resolution strategies using BRKGA and BRKGA + CPLEX. From the tests it was possible to conclude that using the result of a single execution of BRKGA or CPLEX with only a time limit are not the best options. Regarding computational time and solution quality, we concluded that, for the models without and with idle time of berths, the combination strategies with clustering and multiple executions in parallel with fixed parameters were the strategies with the best balance. In this work, we also bring a summary of the Berth Allocation Problem, along with a bibliographical review, a little about the theory of Genetic Algorithms, Random Key Genetic Algorithms and Biased Random Key Genetic Algorithms, as well as about clustering.

Keywords: Berth allocation problem; Biased random key genetic algorithm; Clusterization.

Sumário

Introdução

O crescimento do comércio inter e intra continental que movimenta a economia mundial e permite países exportarem produção interna e importarem produtos necessários, não supridos por produção interna, é seguido pela necessidade de aperfeiçoamento no transporte de cargas. Para comércio intra continental, o transporte pode ocorrer via rodovias, ferrovias, transporte aéreo ou aquaviário (marítimo ou fluviais). Mas, quando se trata de comércio inter continental, o transporte ocorre por vias aérea ou marítima [1].

As vantagens do transporte aéreo incluem maior rapidez na entrega de produtos e qualidade no transporte e possibilidade de entrega da carga em pontos mais próximos do destino final (localização de aeroportos não depende da existência de áreas navegáveis). Duas grandes desvantagens do frete aéreo são o custo de transporte, não sendo viável para alguns tipos de produtos, e uma maior limitação com relação ao peso e dimensões da carga [2].

O transporte aquaviário de produtos tem, entre suas vantagens, menor custo em trânsito e alta capacidade de carga. A não ser que seja demandada uma maior velocidade na entrega, o frete marítimo costuma ser preferido ao frete aéreo. Apesar de um menor custo e uma maior flexibilidade no tamanho, peso e dimensões das cargas, o frete marítimo é, geralmente, mais demorado [2].

Entre os países que lideram o *ranking* de importação e exportação mundial de 2014 até 2019, destacam-se Estados Unidos, China, Japão e países da União Europeia, como Alemanha, Holanda, França e Reino Unido [3]. Os dados consultados se estendem até o ano de 2019, no qual o Reino Unido ainda fazia parte da União Europeia.

As Tabelas 1 à 5 a seguir mostram dados das porcentagens de transporte de cargas em importação e exportação para alguns países com base nos relatórios *Trade Profiles* fornecidos por World Trade Organization (2016), World Trade Organization (2017), World Trade Organization (2018) e World Trade Organization (2019). Nas Tabelas 3 e 4, usamos a notação "..." para representar dados não disponíveis ou não reportados.

A Figura 1 mostra o crescimento do transporte marítimo de mercadorias nos anos de 2002 e 2019, diferenciando importação e exportação. Nesta figura é possível observar que, em 2019, o transporte marítimo representou cerca de metade de todos os bens comercializados entre a União Europeia (*EU*) e o resto do mundo (*extra EU-27*).

Com esses dados, é possível perceber a importância do transporte marítimo de

[1] OCT
[2] Fontes (link para site): CROWLEY, OCT, E REALTY CO.
[3] Fontes (link para site): Trendeconomy, Trademap, World Trade Organization

	Ano	Marítimo (%)	Aéreo (%)	Outros (%)	Relatório
Exportação	2015	79,5	15,9	3,9	2016/2017
	2017	81,1	14,0	3,9	2018
	2018	76,4	16,2	5,0	2019
Importação	2015	66,1/65,9	31,8/32,0	2,1/2,1	2016/2017
	2017	56,3	41,7	1,9	2018
	2018	60,6	37,4	2,0	2019

Tabela 1 – Brasil

	Ano	Marítimo(%)	Aéreo (%)	Outros (%)	Relatório
Exportação	2015	22,0	72,9	5,1	2016
	2016	21,4	73,2	5,4	2017
	2017	21,7	73,0	5,4	2018
	2018	21,1	73,5	5,4	2019
Importação	2015	38,5	57,3	4,1	2016
	2016	36,1	59,8	4,1	2017
	2017	36,6	59,4	4,0	2018
	2018	36,0	60,4	3,6	2019

Tabela 2 – Estados Unidos

	Ano	Marítimo(%)	Aéreo (%)	Outros (%)	Relatório
Exportação	2018	62,2	30,6	...	2019
Importação	2018	63,7	29,1	...	2019

Tabela 3 – China

	Ano	Marítimo(%)	Aéreo (%)	Outros (%)	Relatório
Exportação	2014	82,6	17,3	...	2016
	2016	77,7	21,5	0,7	2017
	2017	78,8	20,8	...	2018
	2018	73,1	26,5	...	2019
Importação	2014	75,0	24,7	...	2016
	2016	75,6	23,8	0,7	2017
	2017	75,4	24,3	...	2018
	2018	72,6	26,8	...	2019

Tabela 4 – Japão

	Ano	Marítimo(%)	Aéreo (%)	Outros (%)	Relatório
Exportação	2014	53,0	31,0	14,3	2016
	2015	51,6	32,4	14,6	2017
	2016	45,2	38,7	14,0	2018
	2017	48,9	35,9	13,3	2019
Importação	2014	44,3	36,8	16,5	2016
	2015	43,1	37,3	17,5	2017
	2016	43,6	36,9	16,8	2018
	2017	44,8	35,6	17,0	2019

Tabela 5 – União Europeia

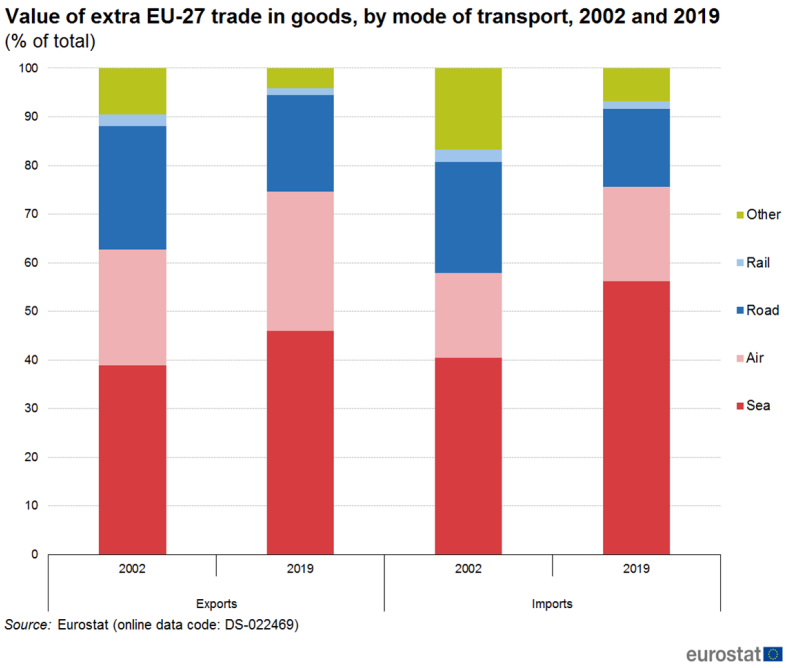
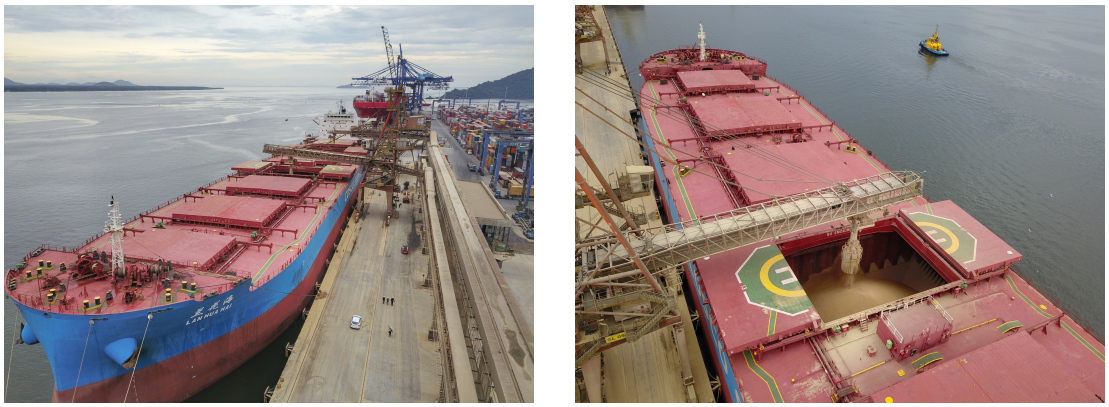


Figura 1 – Comércio de mercadorias entre a Europa e o resto do mundo por meio de transporte (fonte: Eurostat).

cargas para importação e exportação de mercadorias no mundo. Os portos são responsáveis por receberem os navios, manipularem e armazenarem suas cargas. Existem diferentes tipos de navios, para diferentes tipos de cargas, e que requerem estruturas específicas para serem atendidos nos portos. Alguns dos principais tipos de navios de transporte de mercadorias existentes são listados a seguir.

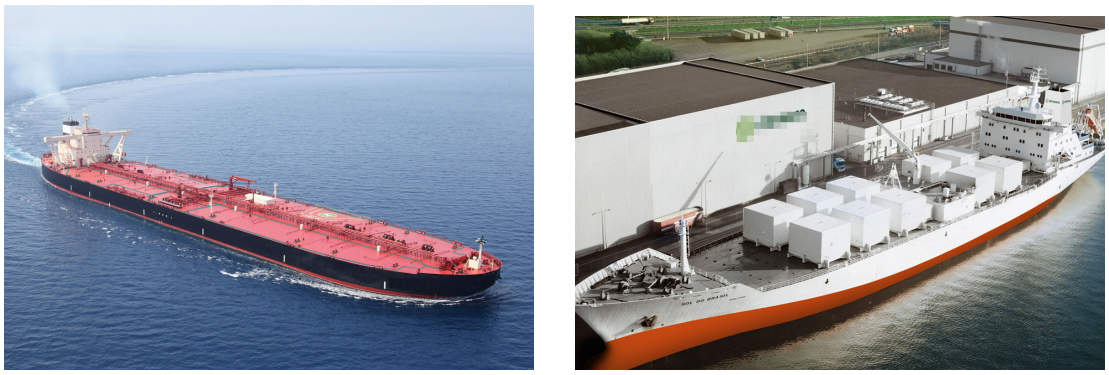
- Navios graneleiros (*Dry-bulk ships*)(Figura 2): navios responsáveis pelo transporte de granel sólido, isto é, de carga seca fragmentada, transportada em grandes quantidades diretamente nos porões dos navios, sem embalagem. O processo de carregar e descarregar esses navios é feito através dos chamados carregadores (*ship loaders*), localizados no terminal portuário, que carregam e descarregam continuamente os materiais sólidos.



(a) (b)

Figura 2 – Navios graneleiros (fonte:www.aen.pr.gov.br).

- Navios-tanque (*Tankers*)(Figura 3): navios que transportam carga líquida à granel diretamente nos porões dos navios, sem embalagem e em grandes quantidades. A carga é armazenada em tanques adaptados para características específicas de cada carga como, por exemplo, temperatura (tanques refrigerados) e pressão. A movimentação desse tipo de carga é feita, de maneira geral, em dutos e por meio de bombas. Exemplos de granel líquido incluem petróleo e seus derivados, óleos vegetais, suco de laranja etc.



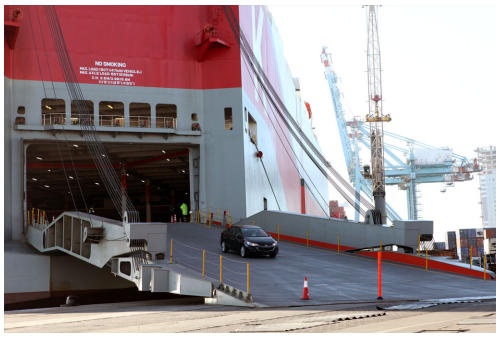
(a) Petróleo (fonte:https://gcaptain.com) (b) Suco (fonte: https://foodsafetybrazil.org/case-amostragem-em-navio-graneleiro/)

Figura 3 – Navio tanque que transporta petróleo e suco (respectivamente).

- Navios gaseiros e químicos (Figura 4): tipo de navio tanque que transporta gás liquefeito (como GPL, GNL, etileno etc.) em tanques especializados, geralmente de forma arredondada e levados acima do convés. Os navios químicos são parecidos com os gaseiros, mas transportam cargas químicas como enxofre líquido, soda cáustica, entre outros.

Figura 4 – Navio gaseiro (fonte:http://www.shippipedia.com/ships).

- Navios *Roll-on/roll-off* (Ro-Ro) (Figura 5): navios que realizam transporte de cargas unitizadas com rodas, como carros (*car carriers*), ou cargas que são carregadas e descarregadas por meio de veículos. A movimentação dessas cargas geralmente é feita por rampas, que costumam ser dobráveis e guardadas no próprio navio. Esses navios possuem portas largas para carregar/descarregar as cargas.



(a) Fonte: www.portosenavios.com.br.

(b) Fonte: www.agcs.allianz.com.

Figura 5 – Navios Ro-Ro.

Como pode-se observar na Figura 5 (b), alguns navios do tipo ro-ro podem levar outros tipos de cargas, como contêineres.

- Navios de carga geral: navios que transportam cargas gerais, isto é, cargas secas a granel, embaladas ou não, ou em contêineres. Como exemplos podemos citar pás eólicas, bobinas de papel e celulose, veículos, caixas etc. Os navios de carga geral também podem transportar cargas em contêineres, apesar de existir uma categoria específica de navios para esse tipo de carga.

- Navios porta-contêineres: navios que levam grandes quantidades de cargas compactadas em contêineres e que não precisam ser descompactadas em cada fase de transporte, caso a carga passe por diferentes meios de transporte até chegar ao destino final.

(a) Fonte: https://www.vesselfinder.com/ship-photos/552622

(b) Fonte: https://www.vesselfinder.com/ship-photos/515249

Figura 6 – Navio de carga geral.

Contêineres são caixas de metal com tamanhos padronizados e medidos em unidades de vinte pés (*Twenty-foot Equivalent Units - TEU*), que podem ser transferidos para outros tipos de transporte como caminhões e trens, por meio de guindastes. A capacidade de carga dos navios também é medida em *TEU*. Alguns navios de contêineres já vêm equipados com guindastes para movimentar a carga, mas em geral é necessário um terminal portuário equipado com guindastes.

Os tipos de contêineres podem ser diferenciados pelo tamanho ou pelo tipo de carga que levam. Diferentes tipos de contêineres foram desenvolvidos para transportarem diferentes tipos de cargas, desde cargas líquidas, sólidas e vivas, até cargas que podem necessitar de ventilação apropriada, refrigeração ou isolamento.

(a) Fonte: https://www.porttechnology.org/news/the_new_giants_of_container_shipping/

(b) Fonte: https://www.rivieramm.com/news-content-hub/news-content-hub/mega-box-ship-orders-ramp-up-57116

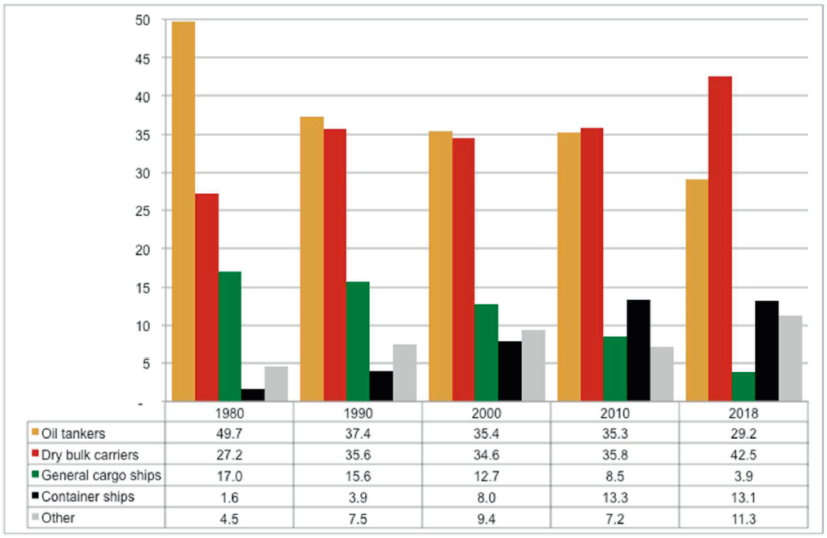
Figura 7 – Navios porta-contêineres.

Um mapa interativo que detalha a movimentação de navios no mundo no ano de 2012, foi desenvolvido pela empresa britânica Kiln [4]. Nesse mapa (https://www.

[4] https://www.kiln.digital/

shipmap.org/) é possível visualizar diferentes tipos de navios com diferentes cores com contadores para suas respectivas cargas máximas transportadas, pode-se observar as rotas dos navios, identificar os portos e visualizar um contador para a emissão de CO_2. No ícone de informações, é possível obter informações sobre os tipos de navios considerados no mapa e a fonte da base de dados. O *site* www.marinetraffic.com é outra fonte na qual pode ser observado um mapa interativo do tráfego marítimo.

Na Figura 8 podemos observar o desenvolvimento das frotas dos principais tipos de navios no comércio de produtos do ano de 1980 até 2018, a cada 10 anos.



	1980	1990	2000	2010	2018
Oil tankers	49.7	37.4	35.4	35.3	29.2
Dry bulk carriers	27.2	35.6	34.6	35.8	42.5
General cargo ships	17.0	15.6	12.7	8.5	3.9
Container ships	1.6	3.9	8.0	13.3	13.1
Other	4.5	7.5	9.4	7.2	11.3

Source: UNCTAD. Review of Maritime Transport 2018.

Figura 8 – Desenvolvimento da frota de navios de 1980 à 2018 (fonte: UNCTAD, 2018).

Podemos observar que a frota de navios responsável pelo transporte de petróleo e derivados sofreu a maior queda entre os tipos de navios considerados, e a quantidade de navios de carga seca a granel aumentou, uma das causas foi o desenvolvimento de demanda por minério e carvão.

A frota de navios porta-contentores aumentou enquanto a frota de navios de carga geral diminuiu. De acordo com o relatório UNCTAD (2018), nos anos de 1970 e 1980 intensificou-se a mudança do transporte convencional de carga geral para contêineres, junto com o desenvolvimento de transporte multimodal, movimentos que ocorrem até os dias de hoje, mas em um ritmo mais lento. O crescimento no uso de contêineres acarretou um processo de substituição de mão de obra, reduzindo custos no processo de transporte e manipulação de cargas, além do aumento na qualidade do serviço. Na Figura 9 podemos ver a razão entre a frota de navios de contêineres e de navios de carga geral.

O processo de conteinerização, iniciado em torno dos anos 1960 pelo norte-americano Malcom McLean, possibilitou o desenvolvimento do transporte intermodal e multimodal de cargas (sistemas que abrangem diferentes modalidades de transporte como o

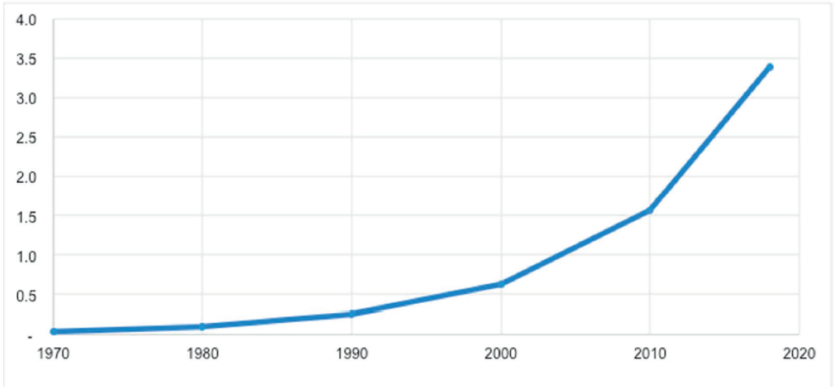


Figura 9 – Razão de contêineres para frota de carga geral (fonte: UNCTAD (2018)).

marítimo, fluvial, rodoviário e ferroviário, no percurso da origem até o destino das cargas), da automatização e redução de custos, de tempo e de danos às cargas na manipulação, movimentação e transporte, além de uma maior segurança, uma vez que os contêineres são dotados de sistemas de segurança[5].

A diferença entre os transportes intermodal e multimodal é que, para o segundo, cada modalidade é feita sob uma documentação fiscal diferente e a responsabilidade do transporte é dividida entre os transportadores de cada modal e trecho utilizado, especificamente o embarcador da carga. Já no multimodal, todo o transporte da mercadoria, da origem até o destino final, é feito sob um único documento fiscal, um único contrato de prestação de serviço e uma única empresa assume total responsabilidade pela operação [6].

O processo de conteinerização trouxe, também, adaptações físicas nos portos, como a implementação de guindastes nos terminais, espaço para armazenamento e formas de transporte dos contêineres dentro do porto (entre áreas de carga/descarga, armazenamento e transferência para outras modalidades de transporte). Além disso, diferentes problemas de logística passaram a fazer parte das decisões a serem tomadas nos terminais como, por exemplo, o local do terminal onde os navios devem atracar para serem atendidos, onde os contêineres de cada navio devem ser transferidos e/ou armazenados e, também, a decisão de alocação de guindastes para terminais que possuem guindastes móveis, definindo quantos, quais e quando deles serão alocados para os navios.

Diferentes problemas de logística e otimização dentro de um terminal portuário envolvendo cargas em contêineres envolvem diferentes áreas do terminal. Uma descrição mais detalhada de cada área de terminais de contêineres, bem como uma descrição detalhada dos respectivos problemas de logística e otimização podem ser encontrados em Steenken et al. (2004). Os autores separaram as operações em operações de cais, que envolvem o atendimento dos navios, e operações de terra, que envolvem armazenamento

[5] BBC-Conteinerização, The Geography of Transport Systems
[6] Inter and Multimodal

(em pátios ou armazéns) e movimentação das cargas dentro do porto, seja a movimentação para pátios e/ou armazéns, para outro navio ou outro meio de transporte (como caminhões e trens). Outras opções que descrevem os problemas de logística em portos podem ser encontradas em Vacca et al. (2008), Stahlbock and Voß (2008) e Rashidi and Tsang (2013), Bierwirth and Meisel (2010), Bierwirth and Meisel (2015).

Entre os problemas de logística e otimização de operações de cais são os problemas de alocação de navios em berços (ou Problema de Alocação em berços - PAB) e problema de distribuição e alocação de guindastes. O PAB consiste em decidir onde e quando os navios devem ser alocados para a manipulação de sua carga e o problema de guindastes consiste em decidir quais navios os guindastes irão atender, bem como o local e horário de atendimento (Bierwirth and Meisel, 2010).

O tempo de atendimento dos navios nos berços depende de quais e quantos guindastes são alocados para atendê-los. Os guindastes podem ser fixos ou móveis no cais e, para cada caso, há maneiras diferentes de lidar com o problema. Para o caso dos guindastes serem fixos, os tempos de atendimento de cada navio em cada berço são fixos e, para o caso de guindastes móveis, esse tempo pode variar de acordo com quantos e quais guindastes são alocados para os navios. Além da alocação e distribuição dos guindastes, o tempo de atendimento dos navios pode considerar a distância do berço até a área do pátio ou armazém destinado às cargas do navio.

Diferentes características do problema resultam em diferentes modelos matemáticos para representá-los. O caso do PAB no qual os navios chegam ao longo do horizonte de planejamento, por exemplo, pode ser descrito através de modelos lineares inteiros, que são difíceis de serem resolvidos computacionalmente pois o tempo de resolução do problema cresce exponencialmente com o aumento do tamanho do problema (Lim, 1998, Bierwirth and Meisel, 2010). Assim, diversas abordagens heurísticas e meta-heurísticas aparecem como uma forma viável de resolver o problema, para que seja possível sua aplicação em situações reais. Dentre as possíveis abordagens, temos os algoritmos genéticos, que são meta-heurísticas baseadas na teoria Darwiniana da evolução das espécies através da seleção natural dos indivíduos mais aptos ao meio (Holland, 1992, Gen and Lin, 2007).

O presente trabalho é focado no problema de alocação de berços considerando fixos os tempos de atendimento dos navios nos berços, que os navios chegam ao longo do horizonte de planejamento (dinâmico) e que os berços são considerados conjunto discreto e finito. Um modelo matemático de Programação Linear Inteiro Misto (PLIM), baseado no proposto por Cordeau et al. (2005), que tem como objetivo minimizar o tempo de permanência dos navios no porto, foi resolvido através do *software* CPLEX e por um algoritmo genético de chaves aleatórias viciadas (Biased Random Key Genetic Algorithm-BRKGA) proposto e implementado com foco em melhorar o desempenho computacional, através de uma implementação vetorizada. Desenvolvemos um estudo para identificar a fase

mais custosa computacionalmente e estratégias para buscar melhorar as soluções obtidas, equilibrando esforço computacional. Pela característica do BRKGA de gerar soluções factíveis sem a necessidade de uma busca local adicional, também testamos a utilização do BRKGA como solução inicial para o CPLEX.

O modelo proposto por Cordeau et al. (2005) busca minimizar o tempo total de permanência dos navios no porto, mas pode ser do interesse do gestor do porto liberar berços mais rápido ou reduzir seus os tempos de ociosidade. Assim, uma alteração no modelo foi proposta para considerar o tempo de ociosidade dos berços, buscando trazer uma visão de interesse do gestor do porto e não somente dos clientes (navios) na tomada de decisão. Os testes e análises computacionais também foram desenvolvidos para o modelo com ociosidade de berços.

O algoritmo genético de chaves aleatórias viciadas (BRKGA) é um algoritmo genético que vem sido bastante utilizado na literatura principalmente para problemas de sequenciamento, apresentando bons resultados (Londe et al., 2024). Além disso, é um algoritmo que consegue garantir a factibilidade alcançada na decodificação mesmo após a aplicação dos operadores genéticos como mutação e cruzamento, sem a necessidade de uma busca local adicional. O BRKGA proposto para os PABs consideram na decodificação das chaves aleatórias a decisão da ordem de alocação e o berço no qual os navios são alocados de maneira independente, buscando uma maior variabilidade nas soluções e possibilitando a alocação dos berços ser considerada diretamente na operação de *crossover*.

Com uma implementação mais simples do BRKGA proposto, investigamos quais os momentos que custam mais tempo de processamento e, a partir disso, desenvolvemos uma implementação vetorizada mais eficiente computacionalmente através da biblioteca *Numpy* para a fase do cálculo do *fitness*, que foi a fase mais custosa computacionalmente e também em outras fases como no cruzamento. A mudança na implementação da decodificação foi realizada para que os cálculos dos tempos fossem feitos de forma matricial, iterando nos berços e nas posições de atendimento, mas não nos indivíduos da população, ou seja, calculando para toda a população de maneira acumulada.

Além de uma implementação vetorizada, investigamos diferentes estratégias para buscar melhor desempenho do algoritmo, explorando o equilíbrio entre diversificação e intensificação. As estratégias testadas e analisadas incluem múltiplas execuções em paralelo e a utilização da ferramenta OPTUNA, mais comum de ser utilizada no contexto de aprendizagem de máquinas, para a busca de hiperparâmetros, e a proposta e implementação de uma clusterização na fase do cruzamento para aumentar a diversificação de maneira a não perder a característica elitista do algoritmo.

O trabalho está organizado da seguinte forma: uma apresentação do problema de alocação de navios nos berços é apresentada no Capítulo 2, incluindo uma revisão bibliográfica; um resumo sobre algoritmos genéticos e sobre algoritmos genéticos de

chaves aleatórias (RKGA) e de chaves aleatórias viciadas (BRKGA) é apresentado no Capítulo 3, incluindo trabalhos de aplicação de BRKGA à variações do PAB; no Capítulo 4 apresentamos um resumo sobre a teoria de clusterização, com ênfase no método selecionado para os testes; e no quinto capítulo são apresentados os modelos e métodos implementados e propostos, com os testes computacionais comentados no Capítulo 6.

1 Problema de alocação de navios em berços (PAB)

Neste capítulo apresentamos o Problema de Alocação de Berços (PAB) com suas variantes e uma breve revisão bibliográfica sobre o problema. Apresentamos, também, o modelo implementado, baseado nos trabalhos de Cordeau et al. (2005) e Mauri et al. (2008b) e descrito em Peteam (2016).

1.1 Descrição do Problema

O Problema de Alocação de Navios em Berços (PAB) busca alocar os navios que chegam no porto para carregarem e/ou descarregarem as cargas de maneira otimizada, estabelecendo a ordem em que os navios são atendidos, em qual localização (espaço do cais ou berço) e quando (ou em qual ordem). Essas decisões podem ser baseadas em diferentes restrições e objetivos a serem otimizados, de acordo com as características dos navios ou do cais.

Os problemas de alocação em berços são classificados com base no *layout* dos berços (contínuo ou discreto), com relação ao horário de chegada dos navios (estático ou dinâmico), ao tempo de atendimento dos navios nos berços (fixo ou variável) e de acordo com a medida de desempenho adotada (função objetivo) (Bierwirth and Meisel, 2010).

Com relação aos horários de chegada dos navios, os problemas podem ser classificados como estáticos ou dinâmicos. No primeiro caso, os navios já estão no porto no início do horizonte de planejamento e, no segundo caso, os navios podem chegar durante o horizonte de planejamento, restringindo o horário em que os navios podem ser alocados (Imai et al., 2001). O PAB estático pode ser resolvido em tempo polinomial, mas o caso dinâmico é difícil de ser resolvido computacionalmente (Bierwirth and Meisel (2010)).

Os horários de chegada e partida dos navios podem ser considerados determinísticos ou estocásticos, assim como os tempos de atendimento em cada berço. De acordo com Bierwirth and Meisel (2010), os tempos de atendimento dos navios são considerados determinísticos na maioria dos trabalhos na literatura.

Com relação ao *layout* dos berços, o problema pode ser classificado como discreto, no qual o cais é dividido em um conjunto finito e enumerável de berços, com tamanhos fixos, ou tratados como pontos (dimensão de espaço ignorada), ou pode ser classificado como contínuo, no qual os navios podem ser alocados em qualquer posição ao longo do cais. Há, também, os problemas híbridos, nos quais o cais é dividido em berços,

como no caso discreto, mas navios grandes podem ocupar mais de um berço e navios pequenos podem dividir berços. Os modelos para o caso discreto podem ser aplicados quando o tamanho do cais não é um fator limitante para a performance do terminal portuário. O caso contínuo é mais difícil de ser resolvido do que o discreto, mas com a vantagem de utilizar o espaço do cais mais eficientemente (Bierwirth and Meisel, 2010).

O objetivo do PAB consiste, geralmente, em minimizar o tempo total de serviço dos navios, considerando o tempo de espera até o navio ser atendido acrescido ao tempo de atendimento nos berços, podendo ou não considerar prioridade de atendimento através de diferentes pesos nos tempos de serviço. Outros objetivos podem ser considerados como, por exemplo, penalizar atrasos ou premiar adiantamentos nos atendimentos, considerar o tempo ocioso dos berços, minimizar o número de navios rejeitados para serem atendidos em um terminal ou até combinar diferentes objetivos na função de custo (Bierwirth and Meisel, 2010) podendo ocorrer, também, objetivos definidos por funções não lineares.

Cordeau et al. (2005) comentam que os tempos de atendimento dos navios nos berços dependem do ponto de atracação dos navios e podem ser descritos em função da distância do berço até a área de carga e descarga dos contêineres, dependência que afeta bastante o desempenho das operações no porto (Mauri et al., 2008b). Além da distância do berço, outra característica que influencia os tempos de atendimento é a quantidade de guindastes alocados em cada berço. No caso da quantidade de guindastes ser fixa em cada berço, os tempos de atendimento podem ser considerados fixos, caso contrário, pode-se integrar o problema de atribuição e alocação de guindastes ao problema de alocação de berços.

O problema pode considerar diferentes restrições espaciais (discreto e contínuo) e temporais (estático, dinâmico, com janelas de tempo etc.). O PAB discreto trata dos berços como um conjunto contínuo e discreto de pontos de atracação e não considera restrições espaciais dos navios e berços. Já no problema contínuo, as restrições devem considerar as dimensões espaciais, como os tamanhos dos navios (incluindo folgas para manobra) e calado (espaço ocupado pelo navio dentro da água), e os tamanhos dos berços, incluindo a profundidade da água.

Com relação às restrições de tempo, no problema estático os navios já estão no porto no início do horizonte de planejamento, não sendo necessário restringir seus atendimentos ao horário de chegada. Já no problema dinâmico, os navios só podem ser atendidos depois de chegarem ao porto. Em qualquer dos casos pode-se considerar um horário máximo de permanência no porto, criando uma janela de tempo para o tempo total de serviço dos navios (tempo de espera mais tempo de atendimento), resultando em restrições sobre o horário de saída dos navios. Além de janelas de tempo dos navios, pode-se considerar janela de tempo (abertura e fechamento) dos berços.

Para minimizar o tempo total de permanência dos navios no porto precisamos

do tempo de atendimento de cada navio em cada berço, que pode ser determinístico e depender do seu local de atracação, podendo considerar fatores como a distância berço até o ponto de armazenamento das cargas (Mauri et al., 2008b) do navio, quantidade de guindastes disponíveis no local, tempo de manobra dos navios ou até a combinação desses fatores. Em geral, considera-se que os navios não podem ser realocados durante seu atendimento, para não prolongar o tempo, mas, em algumas situações especiais como, por exemplo, portos navais, os navios podem ser realocados durante o atendimento, como em Brown et al. (1994).

O problema de alocação de berços pode ser integrado a outros problemas que envolvem a otimização de outras operações portuárias como, por exemplo, a alocação e planejamento da alocação de guindastes. A otimização de alocação (*Quay Crane Assignment Problem*) e planejamento (*Quay Crane Scheduling Problem*) de guindastes ao longo do cais influenciam bastante no tempo de atendimento dos navios em cada berço, como pode ser visto em Bierwirth and Meisel (2015).

Cordeau et al. (2005) comentam que o processo de decisão de alocação é hierárquico e, geralmente, o problema de atribuição de guindastes é resolvido antes do PAB. A atribuição de guindastes leva em conta regras que considerem o tamanho dos navios e suas prioridades de atendimento. Apenas no caso discreto e com um conjunto fixo de guindastes dedicados para cada berço que os problemas envolvendo guindastes não influenciam o PAB (Bierwirth and Meisel (2010)).

A prioridade de atendimento pode ser tratada, por exemplo, com uma penalização para cada berço de acordo com sua distância de um ponto ideal (Park and Kim, 2003), ou apenas penalizar o tempo de serviço de acordo com o berço no qual o navio irá atracar (Cordeau et al., 2005), ou pode ser tratado de acordo com uma urgência de atendimento independentemente do berço que o navio irá atracar.

Para as diferentes classificações do PAB, ou mesmo para problemas com mesmas características, diferentes formulações foram estudadas e propostas, muitas dessas baseadas em modelos de problemas combinatórios clássicos. Assim como no caso de muitos problemas combinatórios difíceis de serem resolvidos de maneira exata, diversas heurísticas também foram desenvolvidas e estudadas para sua resolução.

As aplicações do PAB vão além de alocação de navios em portos comerciais ou públicos, pode ser aplicado, por exemplo, para alocação de submarinos, que requerem serviços adicionais aos de carregamento/descarregamento, como serviços de manutenção.

1.2 Revisão bibliográfica

Um modelo para o problema de alocação para a base naval de Norfolk (Reino Unido) foi proposto por Thurman (1989), seguido de Brown et al. (1994). Além de carregar/descarregar carga (que pode ser carga perigosa), os navios podem precisar de serviços como manutenção, treinamento da tripulação, inspeções obrigatórias por lei etc. e, como não são todos os berços que oferecem todo tipo de serviço, as restrições de atendimento aumentam. O processo de alocação nessa base naval era, até então, feito à mão, o que resultava em muitas trocas de berços pelos navios e consequente tempo ocioso dos navios no porto. A troca de berços pelos navios em um porto naval é uma operação perigosa, trabalhosa e cara. Assim, o plano de alocação modelado busca minimizar os conflitos nos atendimentos, reduzindo a necessidade de troca de berços e garantindo que todos os serviços necessários para cada navio sejam realizados.

Imai et al. (1997) trataram do problema estático e discreto para um porto público na Ásia, trabalhando com um modelo de programação não linear inteiro e multiobjetivo, que busca o equilíbrio entre minimizar o tempo de serviço total dos navios e a insatisfação dos navios com relação à ordem de atracação. O problema é reduzido a um problema clássico de atribuição bidimensional, resolvido pelo método Húngaro. Os testes foram feitos com instâncias geradas para 5, 10 e 15 berços, considerando uma média de 2 ou 4 navios por berço. Os horários de abertura dos berços são iguais, assim como os horários de chegada dos navios são considerados idênticos. O tempo de atendimento é gerado aleatoriamente por uma distribuição uniforme e considerando que o tempo de carregamento/descarregamento de cada contêiner ocorre em torno de $1,5$ a $3,5$ minutos e que cada navio tem uma capacidade de, em média, 2000 contêineres. Esses dados foram baseados no porto de Kobe, no Japão.

A formulação de Imai et al. (1997) para o PAB discreto e estático foi estendida por Imai et al. (2001) para tratar o caso dinâmico, ainda discreto, e apresentaram uma heurística baseada em relaxação Lagrangiana, utilizando o método do subgradiente, para resolver o problema. Os autores trabalharam com o problema dinâmico, discreto no qual os navios podem ser atendidos em qualquer berço, com tempos de atendimento determinísticos e que dependem do berço onde são alocados (da distância do berço até o local de armazenamento da carga). O modelo apresentado trata do PAB como um problema de atribuição tridimensional com algumas restrições adicionais, que não pode ser resolvido de maneira exata em tempo polinomial. Os autores comentam que o PAB estático proposto em Imai et al. (1997) pode ser formulado como um problema de atribuição tridimensional mas que pode ser reduzido a um problema de programação linear clássico de atribuição bidimensional, resolvido em tempo polinomial.

As instâncias em Imai et al. (2001) também foram baseadas no porto de Kobe e geradas para 5, 7 e 10 berços, com 25 e 50 navios e as distribuições usadas para gerar o

horário de chegada e o tempo de atendimento dos navios são independentes. Os autores concluem que, além do método ser adaptável para problemas com dimensões práticas, pode ser útil para tomadas de decisão de quantos berços devem operar para não ultrapassar um determinado custo.

Nishimura et al. (2001) apresentaram uma formulação não linear para o PAB dinâmico no qual os berços podem atender mais de um navio simultaneamente, se houver espaço suficiente, e considerando restrições de profundidade da água nos berços. Como o problema não é resolvido em tempo polinomial, os autores apresentaram dois algoritmos genéticos para a sua resolução, que diferenciam-se entre si pelo tipo de representação do cromossomo. Os autores compararam o desempenho dos algoritmos genéticos (sem considerar atendimento simultâneo de navios) com a relaxação Lagrangiana proposta em Imai et al. (2001), propondo três heurísticas diferentes para obter uma solução factível a partir da solução obtida pela relaxação Lagrangiana. As instâncias utilizadas nessa parte foram geradas de maneira semelhante à proposta por Imai et al. (2001).

Além de comparar com a relaxação Lagrangiana, os autores fizeram testes para comparar o desempenho computacional e qualidade da solução entre os dois tipos de representação dos cromossomos, ainda sem considerar atendimento simultâneo de navios nos berços. As instâncias foram as mesmas usadas na comparação com a relaxação Lagrangiana.

Os testes considerando atendimento simultâneo nos berços com dados reais foram feitos com informações baseadas nas estatísticas de um mês de operação do porto de Kobe. O problema foi dividido em subproblemas com critério de tempo, e resolvido iterativamente, no qual o próximo subproblema herda a solução do anterior. O problema todo foi dividido, com período de planejamento de um dia (resultando em 29 subproblemas), 3 dias (10 subproblemas) ou 6 dias (5 subproblemas), para o número de berços disponíveis dentro do intervalo de 3 a 7.

Imai et al. (2003) abordaram o PAB dinâmico para um terminal definido como um longo berço que pode atender simultaneamente vários navios, os quais são alocados dinamicamente e nem sempre atribuídos a um local específico do berço (*Multi-User Container Terminal* - MUT), com foco em portos públicos do Japão. Os autores dividiram o cais em blocos para obter um conjunto de atribuições de navios para os blocos, referidos como berços. Foi desenvolvido um modelo para o PAB considerando como base o proposto em Imai et al. (2001), adaptando-o para considerar prioridade de atendimento através de variação nos tempos de serviço dos navios. O modelo resultante é um problema de programação não linear inteiro e os autores tentaram uma abordagem por relaxação Lagrangiana mas, devido a dificuldade de resolver, desenvolveram uma heurística baseada em algoritmo genético.

O horizonte de planejamento que Imai et al. (2003) usaram depende dos horários

de chegada dos navios disponíveis como dados de entrada. As instâncias usadas foram geradas de maneira aleatória mas sistematizada para quatro problemas básicos com 25, 50, 75 e 150 navios e 5 berços. Os tempos de atendimento dependem do berço e o tempo mínimo de atendimento de cada navio em um berço em particular foi calculado de acordo com o volume de contêineres que cada navio carrega multiplicado pelo tempo médio para carregar/descarregar um contêiner (com base em dados do porto de Kobe, no Japão). Os horários de atendimento em outros berços (tempos mais longos) foram gerados por números aleatórios. Para os horários de chegada foram gerados quatro padrões por distribuição exponencial com números aleatórios e as prioridades são definidas de acordo com o volume dos navios.

Hansen and Oğuz (2003) consideraram os casos estático e dinâmico para o PAB discreto e apresentaram os modelos propostos por Imai et al. (1997) (caso estático) e Imai et al. (2001) (caso dinâmico), que têm como objetivo minimizar o tempo total dos navios no porto, somando o tempo de espera e de atendimento. Nesses modelos são consideradas as ordens diretas de alocação dos navios nos índices das variáveis e Hansen and Oğuz (2003) comentaram os problemas dessa abordagem. Assim, os autores propuseram uma correção para os modelos, considerando a ordem reversa de alocação nos índices das variáveis, e uma formulação compacta para o caso dinâmico.

Os testes computacionais apresentados por Hansen and Oğuz (2003) foram feitos para problemas com 5 e 10 berços e 10, 15, 20, 25 e 30 navios, gerados computacionalmente, de maneira sistematizada, de acordo com o proposto em Imai et al. (2001). Os horários de chegada foram gerados com uma distribuição uniforme cujo intervalo depende da quantidade de navios e berços, os tempos de atendimento são gerados por uma distribuição uniforme levando em consideração o tempo médio de manuseio de um contêiner, fornecido em Imai et al. (2001) de acordo com um porto no Japão, e considerando em média 2000 contêineres para cada navio. Os horários de abertura dos berços gerados são frações do intervalo de tempo entre a chegada do primeiro e último navio e, para cada teste, os horários de abertura dos berços são considerados iguais. Uma das conclusões dos autores foi que o tempo para resolver o caso dinâmico diminui conforme aumenta a quantidade de navios que chegam ao porto antes do horário de abertura dos berços, fato que se torna mais aparente para as instâncias maiores.

Park and Kim (2003) introduziram um modelo não linear inteiro para os problemas de agendamento de berços e guindastes integrados, considerando o cais contínuo e a duração do atendimento nos berços inversamente proporcional ao número de guindastes atribuídos aos navios, variando linearmente de acordo com esse número. Os horários de chegada e partida dos navios são considerados como um horário esperado, acordado entre as transportadoras e os operadores do terminal portuário. Os autores propuseram a divisão em duas fases para a resolução do problema e as instâncias testadas consideraram até 40

navios. A primeira encontra uma solução quase ótima através uma relaxação Lagrangiana com o método do subgradiente, determinando a posição e o horário de alocação dos navios, assim como a quantidade de guindastes alocados para cada navio em cada segmento de tempo, buscando minimizar a soma ponderada do custo de manuseio dos contêineres, de um custo dependente da distância do local de atracação de uma embarcação até o pátio onde ficarão os contêineres e um custo decorrente de penalidade pelo atraso na chegada e partida das embarcações de acordo com o horário esperado de chegada e partida. Como uma relaxação Lagrangiana é resolvida, a solução obtida pode não ser factível e, assim, os autores propuseram um método para buscar uma solução factível a partir da encontrada pela relaxação. Na segunda fase, baseado na solução da primeira, um planejamento detalhado de cada guindaste é construído, determinando quais guindastes específicos atenderão cada navio, assim como o horário de início e término do atendimento. O objetivo nessa fase é minimizar o número de *setups* dos guindastes e atrasos para iniciar a preparação para atendimento dos navios. Essa fase foi resolvida por programação dinâmica.

Steenken et al. (2004) comentam o aumento no uso de contêineres nos portos ao longo dos anos, que foi modificando a forma de transporte de produtos, assim como a estrutura dos portos para se adaptarem ao carregamento e descarregamento de cargas em contêineres. O objetivo dos autores foi fornecer uma visão geral das operações dos terminais de contêineres (portos com estrutura para trabalhar com cargas em contêineres) e fornecer uma classificação para essas operações, levando em conta equipamentos específicos e instalações de empilhamento de contêineres. Os autores forneceram, também, uma revisão bibliográfica sobre modelos matemáticos e aplicações em operações portuárias, com base na classificação proposta.

Cordeau et al. (2005) apresentaram duas propostas para o PAB dinâmico, discreto e com janela de tempo para os berços e navios. O primeiro modelo apresentado foi o proposto por Imai et al. (2001) e, para o segundo modelo, os autores propuseram um modelo baseado em um Problema de Roteamento de Veículos com Janela de Tempo e múltiplas garagens (PRVJT) (Cordeau et al., 2001), com os depósitos possuindo um único veículo cada. No modelo baseado em PRVJT, os berços são equivalentes aos depósitos e os navios aos clientes e, na função objetivo, além de considerar o tempo de total de permanência dos navios no porto, também foi tratada a prioridade de atendimento dos navios, o que não é possível no modelo proposto por Imai et al. (2001). Os autores comentam que, para instâncias pequenas, ambos modelos foram resolvidos pelo CPLEX, com o primeiro modelo melhor que o segundo e, para instâncias maiores, foi necessário utilizar metaheurística. Foram propostas metaheurísticas baseadas em Busca Tabu para o BAP dinâmico e discreto, para a variante do PRVJT. Depois, desenvolveram uma variação do método para lidar com o problema contínuo, considerando não somente a dimensão temporal mas, também, dimensão espacial do problema. Os autores comentam que os

modelos discretos para o problema podem ser aplicados quando o tamanho dos berços não for um limitante para a performance do porto.

Para os testes computacionais, Cordeau et al. (2005) buscaram gerar instâncias realistas, baseadas em estatísticas obtidas de análise de dados do tráfego e alocação em berços do porto Gioia Tauro, na Itália. Para o caso discreto, consideraram 25 navios com 5, 7 e 10 berços, e 35 navios com 7 e 10 berços, considerando prioridade de atendimento igual a 1 para todos os navios para efeito de comparação. Os horários de abertura para os berços foram gerados como proposto por Imai et al. (2001) mas os tempos de atendimento dos navios nos berços foram gerados de forma diferente, por estarem baseados no porto de Gioia Tauro e não de Kobe. O horizonte de planejamento foi definido para uma semana, com o planejamento dos berços atualizado diariamente.

Uma relaxação do modelo proposto por Cordeau et al. (2005) foi considerada por Mauri et al. (2008b), que propuseram uma metaheurística baseada em *Simulated Annealing* para sua resolução. Os autores relaxaram as restrições de janela de tempo, considerando um modelo do problema de roteamento de veículos com múltiplas garagens e sem janela de tempo, e transferiram as restrições para a função objetivo com penalizações. Após a execução do *Simulated Annealing*, foi aplicado um re-aquecimento para melhorar a solução, intensificando a busca na região próxima à solução encontrada pela heurística. Os autores obtiveram soluções melhores em um tempo computacional menor do que o método baseado em Busca Tabu apresentado por Cordeau et al. (2005), utilizando 30 instâncias com 60 navios e 13 berços geradas por Cordeau et al. (2005).

Uma atualização da revisão bibliográfica sobre operações em terminais portuários que trabalham com contêineres feita por Steenken et al. (2004) foi apresentada em Stahlbock and Voß (2008), seguindo a mesma classificação para os problemas. Os autores comentaram sobre a estrutura dos terminais portuários citando, por exemplo, equipamentos de manuseio de cargas e recursos humanos, bem como problemas de otimização, como a alocação de navios em berços, armazenamento e empilhamento de contêineres e problemas que envolvem guindastes, como distribuição e movimentação.

Um método de geração de coluna para a resolução do PAB discreto e dinâmico com base no modelo de Cordeau et al. (2005) foi proposto por Mauri et al. (2010). O método aplica iterativamente o Algoritmo de Treinamento Populacional (ATP) junto com Programação Linear (PL). A relaxação linear (PL) fornece informações para o ATP gerar boas colunas (baixo custo e boa cobertura dos navios) e, com as colunas, gera-se um Problema de Particionamento de Conjuntos com restrição adicional (PCC^+), resolvido através de um PL. A partir daí, novas colunas são geradas pelo ATP considerando os valores das variáveis duais. Tais colunas, juntamente com o custo reduzido negativo (variáveis duais), são adicionados ao PCC^+ que novamente é resolvido como um problema de programação linear. Esse processo é repetido por uma quantidade máxima de iterações

ou até o número máximo de colunas ser gerado. Os testes computacionais foram feitos com 30 instâncias geradas por Cordeau et al. (2005), cada instância com 60 navios e 13 berços.

Em Bierwirth and Meisel (2010), os autores apresentaram uma pesquisa sobre os problemas de alocação de berços e de atribuição e alocação/agendamento de guindastes, *Quay crane assignment problem* e *Quay crane scheduling problem*, respectivamente, citando diversas referências importantes. Os autores elaboraram um esquema de classificação para os problemas de alocação de berços e agendamento de guindastes de acordo com suas características, analisando cada problema separado, as relações que existem entre eles e analisaram, também, os três problemas integrados, apresentando abordagens de integração encontradas na literatura. Os autores expõem alguns trabalhos nos quais foi mostrado que o PAB é um problema NP-difícil, relacionando-o com outros problemas clássicos como um problema de particionamento de conjunto (Lim, 1998) ou um problema de corte e estoque bidimensional (Imai et al., 2005b).

Bierwirth and Meisel (2010) comentam que os problemas são classificados de acordo com quatro atributos: atributo espacial (*layout* do berço), atributo temporal (restrições temporais para o processo de serviço dos navios), atributo do tempo de atendimento (como os tempos de atendimento dos navios são considerados no problema) e medida de performance (forma de avaliar soluções possíveis do problema). O último atributo reflete a qualidade do serviço oferecido e é medido pela função objetivo do modelo.

Buhrkal et al. (2011) descrevem modelos importantes para o PAB dinâmico e discreto encontrados na literatura: o modelo proposto por Imai et al. (2001) e a melhoria proposta por Monaco and Sammarra (2007). Monaco and Sammarra (2007) melhoraram o modelo ao perceberem que os tempos ociosos dos berços são independentes de qual navio é servido na p-ésima posição do berço k e, então, pode-se reduzir o índice do navio e trabalhar apenas com os índices de posição de atendimento e de berço nas variáveis que representam os tempos ociosos dos berços. Além disso, Buhrkal et al. (2011) descreveram o modelo proposto Cordeau et al. (2005) e propuseram melhorias para o modelo.

Para o modelo proposto por Cordeau et al. (2005), Buhrkal et al. (2011) incluíram um conjunto de restrições para considerar os casos nos quais determinados navios não podem ser atendidos por determinados berços devido a infactibilidades nas dimensões de espaço, que ocorrem nas instâncias geradas por Cordeau et al. (2005) e também propuseram melhorias no modelo, considerando conjuntos de berços idênticos e fixando valores de variáveis de decisão de modo a não perder soluções ótimas. Além disso, os autores acrescentaram inequações válidas (equivalentes ao proposto em Desrochers et al. (1992)), para melhorar os limitantes inferiores de algumas variáveis que possuem bastante importância na função objetivo do problema.

Outro modelo que Buhrkal et al. (2011) descreveram e testaram um modelo proposto na literatura baseado em um problema de particionamento de conjuntos ge-

neralizado (*Generalized Set-Partitioning Problem* - GSPP), no qual consideraram que todas as medidas de tempo são números inteiros. Nesse modelo, cada coluna (variável) representa uma atribuição factível de um único navio à um berço, em um tempo específico, e as primeiras n linhas correspondem aos n navios. A função objetivo busca minimizar o tempo de serviço dos navios. Os autores comentaram que esse modelo oferece algumas vantagens comparado com os outros, como incorporar restrições mais avançadas com relação à alocação dos navios, contanto que considerem apenas um navio para poderem ser tratadas na geração de colunas e, contanto que a função objetivo possa ser escrita como uma soma de custos de colunas, é possível lidar com uma função objetivo mais complicada. Por outro lado, esse modelo não consegue lidar facilmente com restrições que envolvem diversos navios como, por exemplo, o caso de o horário de atendimento de um navio em um berço depender do término de atendimento do navio precedente neste berço. Esse modelo exige a geração das colunas antes de sua resolução e, nos testes feitos por Buhrkal et al. (2011), os tempos de geração das colunas foram sempre menores que um décimo de segundo e, então, nem foram considerados no tempo computacional de resolução do problema.

Buhrkal et al. (2011) concluíram que o modelo proposto por Imai et al. (2001) (denotado por DBAP - *Dynamic Berth Allocation Problem*) supera o proposto por Cordeau et al. (2005) (denotado por HVRPTW - *Heterogeneous Vehicle Routing Problem with Time Windows*) no sentido de resolver instâncias maiores. Contudo, o modelo com os melhoramentos propostos por Buhrkal et al. (2011) (HVRPTW+), tornou-se bastante competitivo com o DBAP, apresentando melhores *gaps* na maioria dos casos e resolvendo mais instâncias até a otimalidade, embora o DBAP tenha fornecido bons limitantes superiores. Além disso, o HVRPTW+ forneceu melhoramentos significativos quando comparado com o HVRPTW, devido ao tipo de desigualdades válidas introduzidas.

O modelo proposto por Monaco and Sammarra (2007) (DBAP+), com melhorias no modelo *DBAP*, apresentou-se superior a este último mas não puderam concluir superioridade quando comparado ao HVRPTW+. Nos experimentos feitos, Buhrkal et al. (2011) observaram que HVRPTW+ é mais rápido que o DBAP+, além de ter resolvido até a otimalidade duas novas instâncias, mas o DBAP+ forneceu melhores limitantes superiores, principalmente para instâncias com maior razão navios/berços. O modelo *GSPP* foi superior aos outros quatro modelos apresentados. Buhrkal et al. (2011) comentaram que, em geral, o horizonte de planejamento utilizado é de uma semana, mas que pode ser atualizado de acordo com mudanças nos horários de chegada e partida dos navios.

Os casos dinâmico e discreto do problema de alocação de berços também foram tratados por Lalla-Ruiz et al. (2012), que desenvolveram uma meta-heurística híbrida combinando uma Busca Tabu (baseada no desenvolvido por Cordeau et al. (2005)) com *path-relinking*, com o objetivo de minimizar o tempo total de permanência dos navios

no porto. Os autores compararam suas propostas com o desempenho obtido pelo *GSPP*, apresentado em Buhrkal et al. (2011), e com a Busca Tabu desenvolvida por Cordeau et al. (2005). Apesar da Busca Tabu dos autores ter sido baseado no desenvolvido por Cordeau et al. (2005), possui algumas diferenças, com a principal delas sendo o fato de terem usado duas estruturas de vizinhança diferentes, enquanto Cordeau et al. (2005) usaram uma.

Antes de testarem instâncias dos problemas, Lalla-Ruiz et al. (2012) testaram diferentes combinações para os parâmetros e utilizaram o teste não paramétrico de *Friedman* para detectar as diferenças nas combinações. Os testes computacionais da abordagem híbrida foram feitos com problemas cujo número de navios varia entre 25 e 60 e o de berços entre 5 e 13, utilizando as instâncias propostas por Cordeau et al. (2005) e outras que os próprios autores geraram, com horizonte de planejamento mais longo, maior tráfego e menor número de berços disponíveis.

O algoritmo híbrido de Lalla-Ruiz et al. (2012) superou a meta-heurística proposta por Cordeau et al. (2005) tanto em qualidade de solução como em tempo computacional. Para instâncias menores, obtiveram soluções ótimas ou quase ótimas na maioria dos casos em uma quantidade menor de tempo quando comparado com o *GSPP* e, para instâncias médias e grandes, para as quais o *GSPP* não chegou à otimalidade, o algoritmo híbrido superou significativamente o algoritmo de Busca Tabu proposto por Cordeau et al. (2005).

Um método de *clustering search* (CS) para resolver o PAB discreto e dinâmico foi proposto por de Oliveira et al. (2012). O CS é um método iterativo que divide o espaço de busca em *clusters* e busca por regiões promissoras. O método proposto conta com a metaheurística *Simulated Annealing* (SA) para geração de soluções, com um processo de agrupamento e uma heurística de busca local. Os autores utilizaram a formulação baseada no PRVJT, proposta por Cordeau et al. (2005), com a aplicação da metaheurística SA proposta por Mauri et al. (2008b). Os autores comentaram que o resultado obtido com a aplicação do algoritmo de treinamento populacional junto com um modelo de programação linear, utilizando a técnica de geração de colunas proposto por Mauri et al. (2008a) superam os resultados obtidos com a meta-heurística SA de Mauri et al. (2008b). Para os testes computacionais, os autores utilizaram 30 instâncias distintas, com 60 navios e 13 berços, baseadas nos dados do porto de *Goia Tauro* (Itália) e geradas aleatoriamente por Cordeau et al. (2005). Os autores concluíram que o método CS foi capaz de fornecer soluções com boa qualidade e com tempos computacionais significativamente baixos.

Elwany et al. (2013) trataram do problema de alocação dinâmico e contínuo integrado ao problema de alocação de guindastes (definir quantos e quais guindastes vão atender cada navio). Nesse trabalho, os autores desconsideram os tempos de manobra dos navios e de movimentação dos guindastes ao longo do cais e consideraram que, uma vez começado o atendimento de um navio, o processo não é interrompido até que o

atendimento seja completado. Os guindastes foram considerados idênticos e sem restrição de atendimento aos navios, ou seja, todos os guindastes podem atender todos os navios. A profundidade da água também é considerada e todos os parâmetros de entrada são considerados determinísticos.

A versão tática do PAB dinâmico e discreto que busca determinar a posição e o horário de alocação dos navios e, também, a alocação de perfis guindastes, foi tratada por Lalla-Ruiz et al. (2014), que propuseram um algoritmo genético de chaves aleatórias viciadas (BRKGA) para a sua resolução. Cada perfil de guindaste define uma alocação de quantos e quais guindastes são alocados em cada passo de tempo para cada navio em cada berço, definindo os tempos de atendimento. Na função objetivo, os autores buscam maximizar o valor total dos perfis de guindastes escolhidos e minimizar os custos relacionados ao fluxo de contêineres que passam de um navio para outro, ou seja, quando chegam por um navio e serão carregados em outro navio para serem transportados novamente. As operações que envolvem a transferência de contêineres de uma área do cais até outra (para troca de navios) são chamadas de *housekeeping*. Os autores também consideraram prioridade e importância dos berços.

Lalla-Ruiz et al. (2014) compararam a abordagem proposta com o modelo proposto por Giallombardo et al. (2010), que usaram de base para o BRKGA que propuseram, e, também, com os resultados obtidos por Vacca et al. (2011) e Vacca et al. (2013).

Bierwirth and Meisel (2015) continuam a revisão bibliográfica feita em Bierwirth and Meisel (2010), comentando estudos para melhoria de diversas operações portuárias mas com foco no problema de alocação de berços e de alocação e agendamento de guindastes, principais operações ao longo do cais. Os autores utilizaram o mesmo esquema de classificação que Bierwirth and Meisel (2010) para esses problemas, de acordo com suas características, para classificar o problema de alocação de berços, o problema de agendamento de guindastes, o problema de alocação de berços e alocação de guindastes integrado, o problema de alocação e agendamento integrado e o problema de alocação de berços e alocação e agendamento de guindastes integrado. Os três problemas integrados determinam o tempo de permanência dos navios no porto, o que reflete a qualidade do serviço prometido para as companhias de transporte e, assim, reflete a competitividade do porto.

Os autores também apresentam uma revisão bibliográfica de abordagens que integram operações de planejamento no cais com os tempos de navegação dos navios, reduzindo o tempo de espera dos navios em portos congestionados. Problemas de otimização de operações no pátio, considerando recursos como caminhões que transportam carga dentro do porto, guindastes no pátio e locais de armazenamento de carga também influenciam no desempenho do serviço oferecido pelo porto.

Saadaoui et al. (2015) propõem uma forma dinâmica de gerar as colunas

necessárias para resolver o PAB dinâmico e discreto, com horizonte de planejamento fixo e particionado em intervalos de tempo discretos, modelado como particionamento de conjuntos e que, como comentam os autores, pode ser facilmente adaptado para outras variantes do PAB. O objetivo do PAB considerado é minimizar o custo total de serviço dos navios no porto. Os autores argumentam que a geração de colunas estática e *a priori* à resolução do PAB pelo modelo de geração de colunas, como descrito em Buhrkal et al. (2011) e Umang et al. (2013), pode causar problemas de memória ou ter um tempo de resolução computacional alto. Os autores iniciam o procedimento com um conjunto de colunas ativas geradas a partir da ordenação primeiro a chegar - primeiro a ser atendido, de acordo com os horários de chegada fornecidos. Com essas colunas, obtêm as variáveis duais (custos relativos), resolvem a relaxação linear do problema formulado como particionamento de conjuntos e resolvem n (quantidade de navios) subproblemas, um para cada navio da instância. As colunas com custo relativo negativo (potencial de melhorar a solução) da solução de cada subproblema são adicionadas ao conjunto de colunas ativas. Com o conjunto de colunas ativas atualizado, resolvem novamente a relaxação linear do problema de particionamento e seguem as iterações até não encontrarem mais colunas com custo relativo negativo nas soluções dos subproblemas.

Nos experimentos computacionais, Saadaoui et al. (2015) geraram instâncias artificiais, com a maior instância contendo 120 navios e todas com a quantidade de berços fixa em 10, mas cada berço com dimensões de comprimento e profundidade diferentes, considerando um congestionamento médio e alto, definido de acordo com o tamanho do intervalo de tempo dos horários de chegada do navio. Os autores obtiveram resultados nos quais o conjunto de colunas ativas factíveis correspondem à solução ótima ou quase ótima, não sendo necessário implementar *branch-and-bound* para reduzir o *gap* entre os limitantes inferiores e superiores.

Barbosa et al. (2016) descrevem os diferentes tipos problemas de alocação de berços tratados na literatura. Os tipos de problema diferenciam-se pela estrutura dos portos que recebem os navios, alguns portos tratam berços que podem atender múltiplos navios, portos cujo cais é contínuo e os navios podem ser alocados em qualquer espaço, satisfazendo algumas restrições dimensionais, disponibilidade de guindastes etc. Além da estrutura dos portos, há diferenças em como lidar com os horários de chegada dos navios, as dimensões dos mesmos e o tempo de atendimento de cada navio em cada berço, que pode depender do tamanho do navio, da quantidade de carga que carrega, da quantidade de guindastes alocados para seu atendimento, da distância até o depósito onde a carga do navio é estocada etc. Comentam que os diferentes problemas de logística nos portos, como alocação de navios e alocação de guindastes, influenciam nos resultados um do outro e, assim, uma tendência nas pesquisas é tratar os problemas integrados.

Além das diferenças nas características do problema, Barbosa et al. (2016)

também tratam de diferentes modelos e formas de resolução para os problemas, enfatizando, devido a dificuldade de resolver o problema de maneira exata, o uso frequente de algoritmos genéticos devido seu potencial e que cada modelo se beneficia de um método de solução específico.

O PAB dinâmico e contínuo integrado com o problema de atribuição de guindastes invariante no tempo (guindastes ficam fixo durante todo o atendimento de cada navio) foi abordado por Correcher and Alvarez-Valdes (2017). Os autores trabalharam com a definição de quantos e quais guindastes atendem cada navio. Os tempos de atendimento dos navios são estimados de acordo com a quantidade de guindastes atribuídos para atendê-los, sendo independe da posição em que o navio é alocado no cais. O horizonte de planejamento é previamente dado e os autores consideram soluções nas quais alguns navios possam terminar seus atendimentos além do horizonte de planejamento, mas com um determinado custo. Os autores consideraram, também, a prioridade de atendimento nos navios.

Correcher and Alvarez-Valdes (2017) comentaram que uma das limitações em resolver esses problemas é o tamanho das instâncias, e da dificuldade de métodos de resolução para problemas com mais de 50 navios com um horizonte de planejamento de uma semana. Assim, os autores propuseram um método de solução baseado no algoritmo genético de chaves aleatórias viciadas (*Biased Radom Key Genetic Algorithm* - BRKGA) com melhoramento memético, um algoritmo construtivo baseado em listas ordenadas de navios (para construir soluções factíveis) e três procedimentos de busca local. Através dos testes computacionais, concluíram que a abordagem foi capaz de encontrar soluções ótimas para instâncias de até 40 navios e de encontrar boas soluções para até 100 navios.

Agra and Oliveira (2018) trataram do problema de alocação de berços dinâmico (com horário de chegada determinístico), contínuo e sem prioridade de atendimento, integrado com o problema de atribuição e alocação guindastes de diferentes capacidades e variantes no tempo (guindastes podem mudar de navio durante atendimento), o horizonte de planejamento foi dividido em períodos de tempo de tamanhos iguais. Propuseram duas formulações, uma baseada na formulação de posição relativa (*Relative Position Formulation* - RPF) e outra que resulta da discretização das variáveis do tempo e espaço para evitar as restrições com o *big-M*. Utilizaram uma abordagem *brach-and-cut* com horizonte rolante para obter limitantes superiores.

Kovač et al. (2018) trataram com o problema de alocação híbrido, que mistura as duas propriedades do cais (discreto e contínuo), considerando o tamanho dos navios. O problema é classificado como estático mas considera horários previstos de chegada e horários desejáveis de saída nos berços, penalizando adiantamentos na alocação e atrasos na partida e os tempos de serviço nos berços são considerados fixos. A função objetivo é uma soma ponderada de posições favorecidas nos berços, de adiantamento de alocação,

atraso na saída dos navios e tempo de espera dos navios. Os autores propuseram uma meta-heurística de pesquisa em vizinhança variável, o modelo utilizado como base foi inspirado no proposto por Rashidi and Tsang (2013) e foram apresentadas três abordagens metaheurísticas existentes na literatura para comparação.

Um PAB contínuo e dinâmico, com tempos de atendimento considerados fixos e com objetivo de minimizar o tempo total de serviço (ponderado) e o custo de desvio da posição de alocação de preferência foi tratado por Lin et al. (2018). Os autores formularam o problema como um problema de programação linear inteira e apresentaram duas propostas de *Simulated Annealing*, uma que aloca os navios da esquerda para a direita no cais e outra que alocaça os navios pelos dois lados. Testaram instâncias encontradas na literatura (Cordeau et al., 2005, Lee et al., 2010 e Frojan et al., 2015) e resolveram o modelo linear inteiro pelo *software* Gurobi.

Uma variação do PAB discreto, que considera que múltiplos navios podem ser atendidos em um mesmo berço, desde satisfaçam as restrições espaciais, foi tratado por Hammouti et al. (2019). A profundidade da água é considerada a mesma para todos os berços, então são apenas consideradas as dimensões de tamanho e não de profundidade dos navios. O tempo de atendimentos dos navios é conhecido e depende do berço, o problema considerado é o dinâmico, e o horizonte de planejamento trabalhado foi de uma semana. Os autores propuseram uma meta-heurística *Modified Sailfish Optmizer* baseada no comportamento de caça em grupo do peixe-vela, estes como predadores e as sardinhas como presas. Para efeito de comparação os autores aplicaram o método para o problema discreto e compararam com abordagens de solução encontrados na literatura, como a Busca Tabu de Cordeau et al. (2005), o Problema de Particionamento de Conjuntos Generalizado de Buhrkal et al. (2011) e outros. Os autores basearam as comparações em um conjunto de instâncias usadas por Cordeau et al. (2005), com 60 navios e 13 navios.

Os autores compararam os resultados para cais discreto e berços que podem atender múltiplos navios (*conventional layout*) e concluíram que este último é mais eficiente com relação a tempo de permanência dos navios nos portos. Além disso, os autores testaram casos para fatores que podem atrapalhar a eficiência no atendimento dos navios, testando cenários como tráfego congestionado, problemas em guindastes e manutenção nos berços. Dos casos testados para cada cenário e comparando os cenários entre si, os autores concluíram que o fator que mais custa para o porto é o tráfego congestionado.

Correcher et al. (2019) propuseram um modelo de programação linear inteira para o problema de alocação de navios contínuo junto com o problema de atribuição de guindastes invariante no tempo, isto é, o número de guindastes atribuídos a uma embarcação não muda durante sua permanência no berço (Türkoğulları et al., 2014). Além disso, o modelo foi desenvolvido para o problema de atribuição de guindastes de modo que cada navio é atribuído a uma quantidade de guindastes com características específicas.

Os autores estendem o modelo para incluir a decisão de quais são esses guindastes com características específicas.

Três modelos de programação linear inteira para resolver o BAP discreto, dinâmico e determinístico foram propostos por Jos et al. (2019), com o objetivo de minimizar a soma da penalização pelo tempo de espera dos navios até serem atendidos, a penalização por atraso no atendimento, o custo de atendimento nos berços e beneficiar a conclusão antecipada do serviço dos navios. Os autores usam como modelo de referência para comparação o modelo proposto em Hansen et al. (2008), com modificações na definição dos valores do *Big M* utilizado na formulação. As instâncias foram geradas baseadas nas gerações propostas por Imai et al. (2001), Hansen and Oğuz (2003) e Hansen et al. (2008).

Kramer et al. (2019) propuseram duas novas formulações para o problema de alocação de berços, uma formulação indexada no tempo e uma de fluxo de arco. Para melhorar a performance computacional, os autores ainda propuseram melhorias no modelo e um procedimento de fixação de variáveis que permite descartar algumas variáveis do problema, considerando seus custos reduzidos (variáveis duais). Os autores apontam diferentes formas de resolução do PAB dinâmico, baseados na literatura. A forma mais comum encontrada é através de heurísticas e meta-heurísticas, que permitem encontrar soluções factíveis em tempo computacional viável. Esses métodos são indicados em situações nas quais um ambiente dinâmico requer rápidas soluções ou em sistemas nos quais a solução do problema é um dado de entrada para outros problemas. Outra forma é utilizar métodos que misturam abordagens exatas e heurísticas ou meta-heurísticas (denominadas mateurísticas), explorando as capacidades dessas abordagens de maneira integrada. Por fim, tem-se os métodos exatos de resolução, que possuem a vantagem de buscar a otimalidade por meio de limites mas com um custo computacional que cresce exponencialmente com o aumento da dimensão do problema.

Para efeito de comparação com a formulação indexada no tempo (*Time-Indexed formulation* - TI) e com a formulação de fluxo de arco (*Arc-Flow formulation* - AF) e as melhorias propostas, Kramer et al. (2019) descreveram o modelo proposto na literatura baseado em particionamento de conjuntos generalizado, avaliado como o mais eficiente dentre os testados por Buhrkal et al. (2011). As instâncias usadas pelos autores foram as propostas por Cordeau et al. (2005) (para os problemas com 60 navios e 13 berços), as propostas por Lalla-Ruiz et al. (2012) e por Nishi et al. (2020). Os resultados obtidos apresentaram redução do tempo computacional e soluções ótimas ainda não relatadas para instâncias de problemas de grande porte propostas na literatura.

Nishi et al. (2020) propuseram um método que combina relaxação Lagrangiana, com o método do subgradiente e um algoritmo exato para resolver os subproblemas, para obter limitantes inferiores e a abordagem *dynasearch* para fornecer bons limitantes superiores para o BAP dinâmico e discreto. A abordagem *dynasearch* considera o uso de

mais de uma estrutura de vizinhança e também um mecanismo de perturbação para evitar ótimos locais. O modelo que os autores se basearam foi o proposto por Cordeau et al. (2005) (problema de roteamento de veículos com janelas de tempo e múltiplos depósitos), salientando a vantagem de poder facilmente considerar prioridade de atendimento nos navios. Para considerar as situações em que algum navio não pode ser atendido em algum berço por restrições dimensionais, os autores consideraram um tempo de atendimento alto para que não ocorra a alocação. Os autores realizaram testes computacionais com as instâncias de 60 navios e 13 berços geradas por Cordeau et al. (2005), com instâncias propostas em Lalla-Ruiz et al. (2012), e também geraram instâncias maiores, com 100 a 150 navios e 15 berços.

Um problema de seleção de algoritmos foi proposto por Wawrzyniak et al. (2020) para um PAB que considera horários de chegada, tamanho, tempo de atendimento e importância, considerando o modelo híbrido no qual os locais de atracação são considerados um conjunto discreto de berços mas cada berços pode acomodar mais de um navio por vez. O problema de seleção de algoritmos consiste em escolher um algoritmo dentro de um portfólio de opções que possui melhor desempenho para um conjunto de instâncias do problema através de uma programação linear que minimiza a perda de qualidade da solução (*solution quality loss*), limitando o tempo de processamento do algoritmo. A *performance* dos algoritmos foi medida considerando o tempo de execução de cada um e a qualidade da solução obtida. O portfólio selecionado considera algoritmos gulosos, *hill climbing*, *GRASP* e busca local iterada (*iterated local search*).

Um modelo adaptado de Nishi et al. (2020), com ajustes para lidar com especificidades de portos no Brasil, de cenários reais da Administração dos Portos de Paranaguá e Antônia (APPA) foi desenvolvido por Bacalhau et al. (2021). Os autores propuseram duas meta heurísticas compostas pela combinação de um algoritmo genético e uma programação dinâmica aproximada (*approximated dynamic programming*), aplicada como uma busca local e também aplicaram heurísticas para redução do espaço de busca na busca local.

Xiang and Liu (2021) trabalharam com o PAB tático com incerteza no tempo de atendimento dos navios através de um modelo de otimização robusto expandido baseado em dados (*data driven*) buscando minimizar o custo dos desvios entre os tempos planejado e esperados de atendimento dos navios, resolvendo com um algoritmo de geração e colunas e restrições, aplicando antes uma clusterização *K-means* para dividir o conjunto de incerteza em *clusters* que não se sobrepõem.

Um *survey* sobre os problemas de alocação em berços e de alocação e agendamento de guindastes considerando incertezas foi feito por Rodrigues and Agra (2022), que classificaram as publicações apresentadas em três principais classes de abordagens, a proativa, a reativa e a proativa/reativa. Os trabalhos apresentados incluem abordagens de

programação estocástica, *fuzzy*, programação robusta, entre outras, trazendo visibilidade para a representação dos parâmetros de incerteza.

Das diversas variações do PAB, podendo considerar variações nos parâmetros do navio (como os horários de chegada e janela de tempo), do porto (como representação dos berços e a consideração de guindastes), há, também, uma variação que considera a cooperação entre portos vizinhos que formam um grupo e compartilham berços para atenderem os navios que chegam, denominado PAB multi-porto (Guo et al., 2023). Guo et al. (2023) trabalham com o PAB multi-porto, considerando também o problema de estabilidade na cooperação portuária (*Port Cooperation Stability Problem* - PCSP), tratando do desvio de embarcações com alto tempo de espera para portos vizinhos. Os autores primeiro definem agrupamentos de portos vizinhos para depois determinar qual o melhor agrupamento, através de um modelo de programação inteira mista com uma abordagem de geração de colunas para sua resolução.

Cheimanoff et al. (2023) trataram do problema de agendamento de produção, alocação em berços e alocação no pátio de armazenamento integrados para um complexo industrial especializado em exportar fertilizantes e propuseram um método exato com *branch-and-cut* e heurísticos para a resolução do problema integrado, para o qual propuseram um modelo de programação linear inteira misto.

Uma revisão com trabalhos recentes sobre o problema de alocação em berços, categorizando os trabalhos na literatura de acordo com características espaciais, apresentando um modelo para cada categoria definida e métodos de resolução foi apresentada por Li et al. (2023).

2 Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA)

2.1 Introdução

Os Algoritmos Genéticos de Chaves Aleatórias Viciadas (*Biased Random Key Genetic Algorithms* - BRKGA) são uma variação dos Algoritmos de Chaves Aleatórias (*Random Key Genetic Algorithms* - RKGA) que, por sua vez, são meta-heurísticas baseadas em algoritmos genéticos (Gonçalves and Resende, 2011).

As meta-heurísticas são procedimentos que coordenam heurísticas e buscam um equilíbrio entre diversificação (exploração do espaço de busca) e intensificação (exploração de vizinhanças de soluções promissoras). Tais métodos não garantem encontrar a melhor solução possível para o problema, nem possuem garantia da qualidade da solução que encontram. Contudo, são métodos que utilizam menores tempos computacionais e são bastante aplicados em problemas cujo tempo computacional para encontrar uma solução exata aumenta exponencialmente com a dimensão do problema.

O processo de diversificação se caracteriza pela tentativa de escapar de mínimos locais, situação que ocorre em heurísticas simples como heurísticas construtivas (algoritmo guloso, por exemplo) e de busca local. Já o processo de intensificação é a busca por melhores soluções na vizinhança de regiões promissoras, na tentativa de melhorar soluções encontradas. A busca do equilíbrio entre diversificação e intensificação pode ser feito de maneira dinâmica, quando a decisão de qual mecanismo usar depende do progresso alcançado, ou de maneira estática, quando os passos de intensificação e diversificação são fixos, independente do progresso no espaço de busca.

A intensificação e a diversificação podem ser considerados objetivos conflitantes, uma vez que, quando a intensificação é favorecida no algoritmo, a busca é focada apenas nas soluções mais promissoras encontradas até então, negligenciando a exploração de outras vizinhanças e, assim, podendo causar a convergência prematura do método para algum ótimo local. Quando a diversificação é favorecida, a busca é focada em soluções aleatórias pelo espaço de busca, negligenciando a exploração de vizinhanças promissoras. Os algoritmos genéticos são uma classe de métodos de busca que possuem um notável equilíbrio entre intensificação e diversificação (Michalewicz, 1996).

Utilizar métodos heurísticos e meta-heurísticos pode ser vantajoso para problemas cujo método exato ou não está disponível ou exige um custo computacional além do disponível ou necessário para aplicação, quando o custo para encontrar a solução ótima

não apresenta, na prática, um ganho relevante em relação a uma solução heurística. Para alguns casos, o problema pode ser tão difícil ou até impossível de ser resolvido computacionalmente, que apenas encontrar uma solução factível já é o suficiente. As meta-heurísticas têm alcançado grande sucesso para problemas difíceis de otimização combinatória com aplicações em diversas áreas (Osman and Laporte, 1996). Outra vantagem desses métodos é quando os dados de entrada são incertos ou pouco confiáveis, quando não se entende completamente as estruturas dos problemas e para sistemas complexos.

As meta-heurísticas podem ser determinísticas ou estocásticas, podem ser inspiradas em processos naturais (como os algoritmos evolutivos), podem considerar uma população de soluções (algoritmos genéticos) ou trabalhar com uma solução por vez (busca Tabu) enquanto explora o espaço de busca, e podem ser iterativos (melhoram iterativamente soluções conhecidas) ou construtivos (constroem soluções).

Os algoritmos genéticos são meta-heurísticas baseadas em processos naturais, mais especificamente na teoria de evolução de Darwin e possuem diversas variações, como os algoritmos genéticos de chaves aleatórias viciadas, que são um tipo de algoritmo genético com estratégia de evolução elitista.

2.2 Algoritmos Genéticos

O BRKGA é um tipo de algoritmo genético e, então, possui características de estrutura e comportamento básicos de algoritmos genéticos, como a forma de convergência, a possibilidade de convergir para ótimos locais, a necessidade de definição de parâmetros de busca que influenciam o equilíbrio entre intensificação e diversificação etc.

Os algoritmos genéticos são algoritmos probabilísticos evolutivos, que misturam elementos de métodos de busca direta com métodos estocásticos e baseiam-se na teoria de evolução das espécies, de seleção natural, proposta por Darwin. Nessa teoria, a população evolui sob uma pressão seletiva que favorece a sobrevivência e reprodução dos indivíduos mais adaptados ao meio, garantindo uma maior probabilidade desses indivíduos passarem seus genes para gerações futuras.

A ideia dos algoritmos genéticos é adaptar para sistemas artificiais os processos de adaptação que acontecem em sistemas naturais de evolução das espécies (Iyoda, 2000; Holland, 1992). O princípio de sobrevivência do indivíduo mais adaptado garante que a qualidade geral das soluções aumente conforme o algoritmo avança de uma geração para a próxima. Esses algoritmos abordam a evolução pelas características genéticas, ou seja, pelo genótipo dos indivíduos.

De acordo com Goldberg et al. (1989), dois trabalhos marcam o início do desenvolvimento e implementação dos algoritmos genéticos, De Jong (1975) e Holland

(1975). As bases teóricas para esses algoritmos e algoritmos genéticos subsequentes foram formalizadas por Holland (1975) com a teoria e o Teorema dos Esquemas (Holland, 1992).

De Jong (1975) foi o primeiro a considerar cadeias de *bits* simples (algoritmos genéticos simples), de comprimento fixo, e operadores de recombinação e mutação, definindo formas de avaliar o desempenho do sistema adaptativo desenvolvido em um conjunto de experimentos computacionais, guiado pelo conhecimento do trabalho de Holland (1975) (Goldberg et al., 1989). Nesses algoritmos, os operadores de recombinação e mutação são uma classe de sistemas adaptativos genéticos e são chamados de planos reprodutivos, que geram respostas (novos indivíduos) adaptativas que simulam os mecanismos de hereditariedade e evolução.

De Jong (1975) comenta que mesmo os planos de adaptação genética mais simples demonstram desempenho melhor que uma busca aleatória pura no espaço de soluções, mas são facilmente afetados por efeitos colaterais estocásticos, resultantes de processos aleatórios internos (definição de parâmetros). Contudo, com alguns ajustes de parâmetros e modificações no plano reprodutivo básico, o autor comenta que foi possível obter uma considerável melhora no desempenho do algoritmo.

Esses algoritmos podem ser aplicados em diferentes problemas como problemas de otimização irrestritos ou restritos, teoria dos jogos e aprendizagem de máquinas (Goldberg and Holland, 1988; Holland, 1992; Michalewicz and Janikow, 1991; Michalewicz, 1995; Michalewicz and Schoenauer, 1996).

Os algoritmos genéticos usam termos da genética (biologia) para denominar seus elementos e operações. Cada solução do problema a ser resolvido é denominada indivíduo e cada indivíduo é representado por um cromossomo (vetor ou cadeia de *bits*). O conjunto de todas as configurações possíveis que os cromossomos podem assumir forma o espaço de busca (de Lacerda and De Carvalho, 1999) e um subconjunto de tamanho fixo de indivíduos do espaço de busca é chamado de população. As características genéticas (genótipo) dos indivíduos, que podem ser passadas de uma geração para outra, são os elementos que compõem cada solução do problema como, por exemplo, as posições de um vetor para problemas cujas soluções são representadas por vetores.

O valor *fitness* representa a aptidão, ou adaptação, do indivíduo ao meio em que está inserido e esse valor é definido pela função *fitness*. A função *fitness* é definida de forma a avaliar as soluções do problema ou, em termos genéticos, avaliar a adaptabilidade do indivíduo no meio (espaço de soluções do problema). Os operadores genéticos são operações aplicadas às soluções, que causam modificações nos indivíduos da população ou até mesmo geram novos indivíduos para construir a próxima geração. As gerações são as iterações do algoritmo, a cada iteração (geração) um conjunto de soluções (população) é analisado (*fitness*) e utilizado para gerar (através de operadores genéticos) um outro conjunto de soluções para a próxima iteração.

No caso de um problema de otimização, por exemplo, a função *fitness* pode ser a própria função objetivo do problema. Quando as soluções são representadas por vetores, as características de cada indivíduo podem ser os valores carregados em cada posição dos vetores e o operador genético de recombinação (*crossover*), por exemplo, pode ser a combinação de elementos (posições) de dois vetores diferentes, chamados de pais, ou seja, um vetor resultante desse operador tem cada posição com valor herdado de um ou de outro pai. A estrutura básica dos algoritmos genéticos consiste de três fases: avaliação do *fitness* dos indivíduos da população, seleção de indivíduos que transmitirão seus genes e aplicação dos operadores genéticos (mutação e *crossover*, por exemplo) a esses indivíduos selecionados, gerando a próxima geração.

A implementação clássica de algoritmos genéticos parte de uma população inicial (conjunto de soluções iniciais), na qual cada cromossomo (indivíduo/solução) é avaliado de acordo com alguma função (*fitness*), que avalia a adaptabilidade do indivíduo ao meio, ou seja, avalia a qualidade da solução no problema. Alguns indivíduos da população são selecionados para gerar indivíduos da próxima geração (iteração) através de operadores genéticos, como mutação e recombinação (*crossover*). Quanto mais apto o indivíduo for, isto é, quanto maior o valor do seu *fitness*, maior a probabilidade do indivíduo ser selecionado para gerar herdeiros e passar suas características genéticas para as próximas gerações. Com a nova geração formada, avalia-se de novo seus indivíduos e o processo é repetido até que algum critério de parada seja alcançado.

O critério de parada pode ser, por exemplo, um número máximo de gerações produzidas, algum valor limitante para o *fitness* do melhor indivíduo ou quando o melhor *fitness* de gerações consecutivas variam menos que algum valor determinado. Além disso, não é preciso se limitar a apenas um critério de parada, pode-se combinar diferentes critérios.

Entre os diversos algoritmos genéticos existentes, há diversas variações na sua elaboração que podem ser feitas, como diferentes formas de definir os operadores genéticos, critérios de parada e parâmetros como tamanho da população (Grefenstette, 1986). Contudo, as componentes básicas dos algoritmos genéticos são (Gen and Lin, 2007):

- representação genética de possíveis soluções do problema;
- uma forma de criar soluções (formar população inicial);
- uma função de avaliação para classificar as soluções (*fitness*);
- operadores genéticos (como seleção, mutação e recombinação/*crossover*) que geram os filhos para a próxima geração (iteração);

- valores para os parâmetros usados no algoritmo genético (tamanho da população, probabilidade de um indivíduo sofrer mutação, recombinação ou de ser escolhido para compor a próxima geração etc.).

2.2.1 Representação genética (codificação)

Cada indivíduo da população é representado por um único cromossomo e cada cromossomo possui componentes denominados genes. Cada gene possui um valor, que é denominado alelo. A estrutura do cromossomo é definida pela representação de possíveis soluções do espaço de busca do problema a ser tratado mas, em geral, os cromossomos são representados por vetores (representação genética), os genes são as posições do vetor e o valor que cada posição carrega é o alelo.

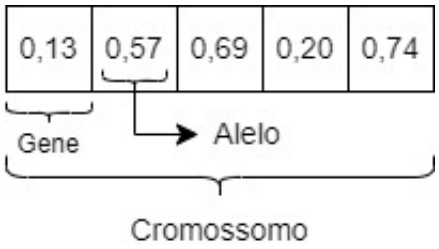


Figura 10 – Representação genética de uma solução (representação por ponto flutuante).

No exemplo da Figura 10, a codificação do indivíduo é feita com números decimais (ponto flutuante), mas os algoritmos genéticos clássicos usam codificação binária (Holland, 1992), denominados *algoritmos genéticos canônicos*. A forma como a codificação é feita influencia na qualidade e convergência do algoritmo, e pode ser classificada de acordo com os símbolos usados (codificação binária ou de ponto flutuante, por exemplo), de acordo com a estrutura das codificações (codificação unidimensional ou multidimensional) ou de acordo com os tipos de conteúdo codificados (apenas a solução ou solução + parâmetros, por exemplo) (Gen and Lin, 2007). Os valores nos genes devem representar alguma(s) característica(s) do indivíduo que será avaliada pela função *fitness*.

Na literatura encontram-se alguns trabalhos que discutem a importância da codificação para a eficiência dos algoritmos genéticos, como em Michalewicz and Schoenauer (1996), que comparam algoritmos com representação binária e de ponto flutuante para problemas de otimização numérica restrita, e concluem que os algoritmos com representação em ponto flutuante foram mais eficientes. Holland (1992) desenvolveu uma teoria para algoritmos genéticos utilizando representação binária, que diz favorecer o paralelismo implícito; Michalewicz (1996) argumenta que a codificação binária para problemas com espaço de busca de alta dimensão não é tão eficiente, argumento que é completamente aceito na literatura; Fogel (1994), por exemplo, argumenta que a dimensão do espaço

de busca sem considerar a representação não determina a eficiência do algoritmo (Iyoda, 2000).

Holland (1992) afirma que a representação binária favorece o paralelismo implícito dos algoritmos genéticos mas, em diversos problemas reais, como otimização numérica com parâmetros reais, a representação em ponto flutuante apresenta melhor desempenho e, de acordo com Michalewicz (1996), a representação binária não é tão eficiente em problemas numéricos com grande dimensão e alta precisão requerida (Iyoda, 2000).

Para a escolha da codificação a ser utilizada é importante levar em conta (Gen and Lin, 2007):

- espaço: os cromossomos não devem requerer uma quantidade muito alta de memória;
- tempo: o tempo para avaliar, recombinar e fazer a mutação dos cromossomos não pode ser alto;
- factibilidade: todos os cromossomos, principalmente os gerados pela recombinação e mutação, devem representar soluções factíveis;
- unicidade: a decodificação dos cromossomos para soluções pode ser um dos três casos: mapeamento de 1 para 1, mapeamento de n para 1 e mapeamento de 1 para n. O melhor caso entre os três é o primeiro e o menos desejável é o último;
- hereditariedade: a prole resultante de combinação deve representar soluções que combinam a estrutura de seus dois pais;
- localidade: um cromossomo que sofre mutação deve representar, em geral, uma solução similar à solução original.

2.2.2 População e população inicial

Os algoritmos genéticos partem de uma população inicial e, assim, é preciso gerar os indivíduos iniciais. A população inicial pode ser gerada, por exemplo, de maneira aleatória ou através de alguma heurística, contanto que, no caso de problemas restritos, a população gerada seja composta por indivíduos factíveis.

O tamanho da população é um parâmetro que, em geral, é constante em todas as gerações e definido previamente. Mas é possível identificar casos em que o tamanho da população varia. Um exemplo disso é o algoritmo ATP aplicado ao Problema de Alocação de Berços, apresentado por (Mauri et al. (2010)).

2.2.3 Função de avaliação ou função *fitness*

Os algoritmos genéticos permitem uma busca multidirecional, mantendo uma população de soluções em potencial, incentivando a troca de características entre essas soluções e gerando, também, indivíduos com novas características. A cada geração, "boas" soluções se reproduzem enquanto as soluções relativamente "ruins" morrem. Essa avaliação de soluções "boas" e "ruins" é feita por uma função que simula um ambiente (Michalewicz, 1996).

O valor da função *fitness* de cada indivíduo (cromossomo) retorna o quão adaptado o indivíduo está ao meio. Em termos de programação, é uma função (função *fitness*) que retorna uma medida de qualidade da solução no problema. De acordo com Goldberg and Holland (1988), a função *fitness* mede o lucro ou a utilidade que queremos maximizar (pegar os indivíduos mais aptos a sobreviverem na natureza). No caso de problemas de otimização, por exemplo, não precisa ser necessariamente a função objetivo do problema, pode-se considerar diferentes medidas como a função objetivo com penalidades para soluções infactíveis, caso o problema seja restrito, e diferentes funções objetivo no caso de problemas de otimização multiobjetivo.

Escolher a forma de avaliar a qualidade do indivíduo não é uma tarefa simples pois afeta todo o processo evolutivo e a convergência do método. Kramer (2017) comentam que a escolha das penalizações para infactibilidade e a escolha de pesos apropriados para otimização multiobjetivo são decisões importantes a serem feitas para uma boa qualidade da solução e convergência do algoritmo. Como, por exemplo, decidir se uma solução próxima de um ótimo global mas infactível tem melhor qualidade que uma solução ruim mas factível em um problema de otimização restrito ou decidir qual objetivo em um problema multiobjetivo tem maior importância (peso) para o *fitness* de um indivíduo, para sua adaptação ao meio.

A quantidade de chamadas da função *fitness* até uma solução ótima ou até uma determinada precisão ser alcançada é uma forma de medir o desempenho do algoritmo genético (Kramer, 2017). Chamar a função *fitness* pode ser, por exemplo, caro computacionalmente e, assim, chamar a função muitas vezes pode tornar o algoritmo lento.

2.2.4 Operadores genéticos

A troca de características entre as soluções e a passagem de características para gerações futuras são representadas pelos operadores genéticos, que influenciam bastante no equilíbrio entre a diversificação e a intensificação. Em geral, a intensificação é feita por algum mecanismo de seleção (seleção dos indivíduos para sobreviverem e se reproduzirem), enquanto a exploração de novas regiões no espaço de busca é feita por operadores genéticos como a mutação (Gen and Lin, 2007).

Os operadores genéticos são usados no processo de herança genética, para alterar a composição genética dos indivíduos ao gerar descendentes e podem ser definidos de forma independente ou dependente do problema. No primeiro caso, os operadores não levam em conta características particulares do problema, sendo aplicáveis em diferentes problemas sem adaptações (operador *crossover* de um ponto, por exemplo). No segundo caso, os operadores levam em conta especificidades do problema e não podem ser aplicados a outros tipos de problemas como, por exemplo, operadores definidos de maneira a manter a factibilidade das soluções no problema.

Os operadores clássicos mais usados são o *crossover* (recombinação) e a mutação para representação binária, Gen and Lin (2007) descrevem a seleção de indivíduos como um operador genético bastante utilizado também mas, para alguns autores como Yao (1993), a seleção é tratada como um mecanismo separado dos operadores genéticos.

O ***crossover*** opera, em geral, em dois cromossomos (chamados de pais), combinando seus genes para gerar descendentes com características similares. Existem diferentes formas de combinar os genes dos pais para gerar os filhos e a forma pode ser dependente ou independente do problema.

Um tipo clássico é o *crossover* de um ponto, no qual seleciona-se um ponto de corte aleatoriamente nos pais, dividindo o cromossomo de cada pai em duas partes e os filhos são gerados pelas combinações dessas partes. Esse tipo de *crossover* é independente do problema.

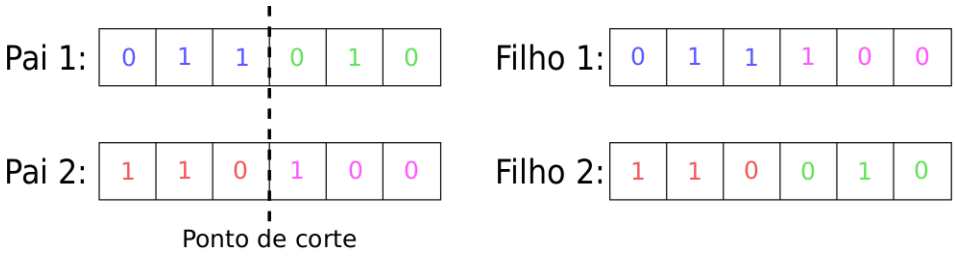


Figura 11 – *Crossover* de um ponto.

O *crossover* de dois pontos é uma variação, na qual são selecionados dois pontos de corte nos pais.

Outro exemplo é o *crossover* uniforme (Michalewicz and Schoenauer, 1996), no qual um filho $z = [z_1, \ldots, z_n]$ é gerado pelos pais $x = [x_1, \ldots, x_n]$ e $y = [y_1, \ldots, y_n]$ de forma que $z_i = x_i$ ou $z_i = y_i$ com igual probabilidade, para $i = 1, \ldots, n$. Um segundo herdeiro é gerado revertendo todas as decisões tomadas para gerar o primeiro, isto é, se o primeiro herdou determinado gene de x, então essa mesma posição de gene no segundo filho receberá o gene de y.

Cada gene de um filho é decidido de qual pai vai ser herdado com uma probabilidade fixa p. O segundo filho é gerado revertendo todas as decisões tomadas

para gerar o primeiro. Essa recombinação tem a característica de que a herança de genes independe da posição do gene no cromossomo.

O operador de *crossover* heurístico usa os valores da função objetivo para determinar a direção de busca e produz apenas um filho, z, a partir de dois pais, x e y, de acordo com a regra (Michalewicz and Schoenauer, 1996):

$$z = r * (x - y) + x,$$

em que $r \in [0, 1]$ e o pai x não é pior que o pai y, isto é, se for um problema de maximização, então $f(x) \geqslant f(y)$ e se for um problema de minimização, então $f(x) \leqslant f(y)$, em que f é a função objetivo do problema.

Michalewicz and Schoenauer (1996) definiu o operador *crossover* aritmético para problemas de otimização restrita convexos, no qual o espaço de busca factível é convexo. Aproveitando a característica de conjunto convexo de que a combinação linear convexa de dois indivíduos no conjunto gera um indivíduo dentro do próprio conjunto, considerando dois indivíduos da população (soluções factíveis) x e y, os herdeiros p e q podem ser definidos da seguinte maneira

$$p = \alpha * x + (1 - \alpha) * y \quad \text{e} \quad q = \alpha * y + (1 - \alpha) * x, \ a \in [0, 1].$$

Uma extensão que pode ser feita para múltiplos pais que podem formar uma família de herdeiros:

$$y = a_1 * x_1 + a_2 * x_2 + \cdots + a_k * x_k, \ a_i \in [0, 1], \ i \in \{1, \ldots, k\}, \ \text{tal que } a_1 + a_2 + \cdots + a_k = 1.$$

A seleção dos indivíduos pais para passarem por esse operador pode ser condicionada a uma probabilidade de *crossover*: para cada indivíduo pode-se gerar um valor aleatório que vai definir se o indivíduo vai ou não ser selecionado, de acordo com essa probabilidade (Michalewicz, 1996). Pode-ser, também, escolher sempre o melhor indivíduo da população e um indivíduo aleatório para serem os pais.

O operador **mutação** é um operador que produz mudanças aleatórias em um ou mais genes de indivíduos da população, procurando criar uma variabilidade extra na população. A seleção dos indivíduos que passarão por mutação geralmente é determinada por uma probabilidade de mutação e, em geral, é atribuído um pequeno valor a essa probabilidade (Iyoda, 2000).

Existem diferentes formas de definir operadores de mutação, para diferentes formas de representação dos indivíduos. Para a representação binária, por exemplo, um operador clássico é a mutação simples, que é independente do problema e consiste em trocar o valor do gene selecionado, isto é, se o gene tiver valor 0, troca para 1 e vice-versa.

Para representação em ponto flutuante, encontram-se alguns operadores definidos em Michalewicz (1996), como a mutação Gaussiana que modifica todos os componentes

de um vetor solução x de acordo com a regra $x^{k+1} = x^k + N(0, \sigma)$, em que $N(0, \sigma)$ é um vetor de variáveis aleatórias gaussianas independentes, com média 0 e desvio padrão σ.

Outro exemplo de mutação para representação em ponto flutuante é a mutação uniforme, que muda um único componente do vetor solução por um número aleatório com distribuição de probabilidade uniforme dentro do domínio da variável. Um caso especial dessa mutação é quando as variáveis têm limitantes superior e inferior. O operador mutação não uniforme, definido por Michalewicz (1996), é um operador dinâmico que explora o espaço de busca uniformemente no começo e mais localmente nas iterações mais avançadas, aplicado também para representação em ponto flutuante. Os autores apresentam outros operadores de mutação, alguns dependentes do problema.

Holland (1992) comenta que a mutação sozinha geralmente não melhora a busca por uma solução, mas fornece uma garantia contra o desenvolvimento de uma população uniforme, incapaz de evoluir.

O operador de **seleção** aumenta as chances de cromossomos de melhor qualidade, ou pelo menos algumas de suas características, serem copiados para a próxima geração e, assim , aumentar a qualidade média da população e guiar a busca para regiões promissoras. A forma de seleção do indivíduos influencia muito no equilíbrio entre diversificação e intensificação, se a seleção dos indivíduos for muito restrita aos melhores indivíduos (seleção mais forte) o algoritmo converge prematuramente mas, se for uma seleção mais uniforme, o progresso evolutivo do algoritmo pode ser mais lento do que o necessário (Gen and Lin, 2007). Um bom equilíbrio pode ser alcançado mantendo uma seleção mais fraca no início do algoritmo para favorecer uma maior diversificação e uma seleção mais forte no final para reduzir o espaço de busca em regiões mais promissoras.

A probabilidade de seleção dos indivíduos pode ser baseada diretamente no valor do *fitness* dos indivíduos, como o método clássico de seleção roleta que atribui a cada indivíduo da população uma probabilidade de passar para a próxima baseada na razão entre seu valor de *fitness* e o valor da soma do *fitness* de todos os indivíduos (Michalewicz, 1996). A probabilidade pode ser baseada, também, na classificação dos indivíduos de acordo com seu *fitness* em vez de depender diretamente desse valor.

Bäck and Hoffmeister (1991) comentam diferentes classificações de seleção, baseadas em diferentes características da definição da probabilidade de seleção. Os métodos que utilizam uma probabilidade de seleção dos indivíduos fixa para qualquer iteração (como os métodos cuja probabilidade está baseada na classificação dos indivíduos) são chamados métodos estáticos e os métodos nos quais a probabilidade varia a cada iteração (como as probabilidades que dependem do valor de *fitness* dos seus indivíduos) são considerados dinâmicos.

A seleção pode ser classificada também como extintiva ou preservativa. A

seleção preservativa garante que todo indivíduo possui uma probabilidade positiva de ser selecionado, ou seja, todo indivíduo possui chance de contribuir com a próxima geração. Já na seleção extintiva alguns indivíduos podem ter probabilidade nula de serem selecionados para contribuírem com a próxima geração. A seleção extintiva ainda pode ser subdividida em seleção extintiva direita, no qual os piores indivíduos têm taxas nulas de reprodução, e a seleção extintiva esquerda, na qual alguns dos indivíduos com melhor desempenho possuem taxa de reprodução nula para evitar convergência prematura (Bäck and Hoffmeister, 1991, Michalewicz, 1996).

A seleção pode ser, também, pura ou elitista. Na seleção pura, os indivíduos têm um tempo de vida de uma geração independente do valor de *fitness*, isto é, os pais passam por mutação ou recombinação uma vez e depois morrem. Já na seleção elitista, alguns ou todos os pais podem ser selecionados juntos com sua prole, podendo resultar em "tempo de vida ilimitado"ou indivíduos super adaptados.

Exemplos de diferentes rotinas de seleção e diferentes definições de seleção podem ser encontrados em Bäck and Hoffmeister (1991), Michalewicz (1996), Shukla et al. (2015) entre outros.

2.2.5 Parâmetros

O desempenho dos algoritmos genéticos depende muito dos valores definidos para seus parâmetros de controle, como o tamanho da população e as probabilidades de mutação e de recombinação. Esses parâmetros influenciam no equilíbrio entre diversificação e intensificação e, assim, na eficiência da busca e convergência do método. Nos algoritmos genéticos clássicos, esses parâmetros podem ser definidos pelo usuário e permanecerem fixos durante toda a execução dos algoritmos, ou podem ser alterados em momentos específicos do processo evolutivo (Iyoda, 2000). Quando definidos manualmente pelo usuário, diferentes valores podem ser testados e os que apresentarem melhores resultados são utilizados. Pode-se, também, utilizar valores sugeridos na literatura mas como cada problema e cada algoritmo genético têm, em geral, comportamentos diferentes com a variação dos parâmetros, pode não ser uma forma viável de defini-los. Outras formas podem ser através de técnicas de otimização, aprendizado de máquina ou mesmo heurísticas e meta-heurísticas.

O tamanho da população é um parâmetro que geralmente permanece fixo em todas as iterações e pode influenciar na eficiência ou na convergência do algoritmo. Uma população com um tamanho menor favorece, por exemplo, o cômputo do *fitness* dos indivíduos e a aplicação dos operadores genéticos, mas diminui a diversidade genética da população, podendo levar a uma convergência prematura do método (Gendreau et al., 2010). Uma população com mais indivíduos favorece uma maior variabilidade genética e uma exploração mais ampla do espaço de busca, mas possui um custo maior para aplicar

operadores e calcular o *fitness* de cada indivíduo, tornando a busca mais lenta.

A probabilidade de mutação influencia na geração de variabilidade genética da população, uma maior probabilidade de mutação gera maior variabilidade genética. Uma forma de variar esse parâmetro durante o algoritmo pode ser manter uma maior variabilidade genética no início do algoritmo e, mais no final do algoritmo, quando as soluções estão convergindo, reduzir para alcançar uma maior intensificação na busca.

No operador de recombinação (*crossover*), tem-se a probabilidade dos indivíduos serem selecionados para passarem pelo processo de recombinação (probabilidade de serem escolhidos para pais) e a probabilidade de um filho herdar as características de um pai ou de outro. As definições dessas probabilidades podem dar maior peso para uma elitização da população, priorizando a escolha de pais com melhores *fitness* e/ou priorizando os genes do pai com o melhor *fitness* dentre os pais selecionados, intensificando a busca na vizinhança de soluções mais promissoras.

Técnicas para resolver os problemas de ajuste (*parameter tuning problem*) e controle (*parameter control problem*) de parâmetros vêm sendo estudadas e desenvolvidas. O problema de ajuste de parâmetros busca melhores valores para definir os parâmetros, e o problema de controle de parâmetros lida com a melhor forma de variar os parâmetros durante a execução do algoritmo (Karafotias et al., 2014).

Um ajuste de parâmetros eficiente é indispensável para os algoritmos genéticos e já existem diversos métodos eficientes bem aceitos na comunidade de computação evolutiva. Já o controle de parâmetros não é indispensável mas possui vantagens de ser utilizado. Karafotias et al., 2014 cita algumas vantagens como permitir o uso de valores apropriados para os parâmetros em diferentes estágios do processo de busca, resolver implicitamente o problema de ajuste de parâmetros e permitir acumular informações sobre o panorama do *fitness* e usá-las para melhorar a performance do algoritmo.

Grefenstette (1986), por exemplo, aplicou um algoritmo genético para ajustar parâmetros de um algoritmo genético, técnica definida como meta-algoritmo genético (Meta-AG). O controle automático dos parâmetros baseado em processos de otimização genética é denominado processo auto-adaptativo (Kramer, 2017).

Grefenstette (1986) implementou um algoritmo genético para otimizar os parâmetros de tamanho de população, taxa de *crossover*, taxa de mutação, *gap* de geração (porcentagem da população que será substituída a cada geração), *scaling window* (define forma de calcular o desempenho de cada indivíduo) e a estratégia de seleção (pura ou elitista). O autor comenta que, em alguns casos, é possível predizer como variações de um único parâmetro (assumindo os outros fixos) afeta a performance do algoritmo, mas é difícil de predizer como diferentes parâmetros interagem juntos.

2.2.6 Teorema dos Esquemas

A ideia da evolução das populações nos algoritmos genéticos é que as representações associadas à boas soluções do problema irão predominar com o passar das gerações e os processos de seleção e combinação de soluções tendem a gerar soluções ainda melhores. Em outras palavras, os algoritmos genéticos exploram regiões do espaço solução com melhor desempenho. Para representações através de *strings* binárias, Holland (1992) comenta que regiões no espaço solução podem ser definidas olhando *strings* que possuem 1 ou 0 em posições específicas. O conjunto de todas as *strings* que possuem 1 na primeira posição, por exemplo, constituem uma região.

Holland (1992) comenta que a capacidade dos algoritmos genéticos de focarem sua atenção em regiões mais promissores do espaço de soluções é um resultado direto da sua capacidade de combinar *strings* contendo soluções parciais. O algoritmo genético favorece *strings* com melhores valores *fitness* para serem submetidos aos operadores de seleção, reprodução e mutação garantindo, assim, que a população seguinte tenha mais *strings* com *fitness* acima da média.

O Teorema dos Esquemas, desenvolvido por Holland (1975), é o fundamento teórico matemático do comportamento de algoritmos genéticos simples, mostrando que as populações vão melhorando, em média, de uma iteração para a seguinte. O teorema foi desenvolvido a partir de um modelo de similaridade binário que descreve um subconjunto de *strings* com similaridades em determinadas posições (Holland, 1975). Essas *strings* são denominadas esquemas (*schemas*), um padrão com capacidade de representar diferentes soluções do problema.

Holland (1975) utilizou a representação de soluções do problema por meio de esquemas binários, sequências de *bits* cujas posições podem assumir valores do conjunto $\{0, 1, *\}$. O símbolo $*$ indica que a posição pode assumir tanto o valor 0 quanto o 1, o que permite um esquema descrever todas as possíveis similaridades entre *strings* de um mesmo tamanho (Iyoda, 2000), ou seja, um esquema é capaz de representar diversos cromossomos simultaneamente. O conjunto $\{0, 1, *\}$ é denominado alfabeto ternário, mas existem diferentes modelos que podem considerar mais elementos. Um subconjunto de similaridades é denominado *schemata*.

Um esquema é similar a uma *string* se as posições com 0 e 1 no esquema correspondem às posições com 0 e 1 na *string* e as posições com $*$ no esquema correspondem ao valor 0 ou 1 na *string*. Por exemplo, o esquema $[*\,1\,1\,1\,1]$ é similar às *strings* {01111, 11111}. De maneira equivalente, o esquema $[*\,1\,0\,1\,*]$ é similar às *strings* {01010, 01011, 11010, 11011}. Outra forma de interpretar seria, por exemplo, os esquemas $[*\,1\,*\,1*]$ e $[1\,1\,*\,*1]$ estão contidos nos cromossomos $[1\,1\,1\,1\,1]$ e $[1\,1\,0\,1\,1]$. O esquema $[*\,*\,*\,*\,*]$ representa todos os cromossomos de tamanho 5, ou seja, representa 2^5

cromossomos.

Com esse esquema de similaridades, um algoritmo genético consegue representar e manipular um número maior de regiões do que a quantidade de *strings* da população. Holland (1992) chama essa característica de paralelismo implícito e comenta que esse comportamento é o que dá ao algoritmo genético sua principal vantagem sobre outros processos de resolução de problemas. Contudo, a operação de *crossover* pode complicar o efeito de paralelismo implícito, pois pode mover um filho para uma região diferente das regiões dos pais, fazendo com que a taxa de amostragem de diferentes regiões se afaste de uma proporcionalidade para a aptidão média (Holland, 1992). A probabilidade de um filho deixar a região de seus pais depende da distância entre os zeros e uns que define a região.

Para cada esquema, são definidos o comprimento definitório e a ordem. O comprimento definitório de um esquema é a distância entre as posições do último elemento diferente de $*$ para o primeiro elemento diferente de $*$, e a ordem de um esquema é a quantidade de elementos diferentes de $*$ que o esquema possui, isto é, o número de elementos fixos do esquema (de Lacerda and De Carvalho, 1999, Iyoda, 2000). Por exemplo o esquema $[*\,1\,1\,1\,1]$ possui comprimento $5-2=3$ e ordem 4, já o esquema $[*\,1\,0\,1*]$ possui comprimento $4-2=2$ e ordem 3. Quanto maior a ordem, mais específico é o esquema e, quanto maior seu comprimento, maior a chance do esquema ser quebrado/rompido pelo *crossover*. Para cada esquema em uma determinada geração, o valor *fitness* do esquema é definido pela média dos valores *fitness* de todos os cromossomos na população da geração que são representados por esse esquema (Iyoda, 2000).

Com as informações de ordem e comprimento de um esquema, da quantidade de cromossomos que esse esquema representa na população, seu valor *fitness*, o valor médio do *fitness* da população, o tamanho da população e as probabilidades de mutação e *crossover* em uma determinada geração t, é possível definir uma quantidade mínima de cromossomos representados por esse esquema na próxima geração, pela *equação de crescimento reprodutivo do esquema*. Com a equação do crescimento reprodutivo, é possível analisar que a seleção aumenta a quantidade de esquemas com valor *fitness* acima da média do *fitness* da população (Iyoda, 2000). Contudo, a seleção não introduz novos indivíduos na população, daí a utilização dos operadores de *crossover* e mutação.

Para a probabilidade de sobrevivência na operação de mutação, é importante a noção de ordem do esquema (Correia, 2004).

Na operação de *crossover*, esquemas de maior comprimento têm mais chance de serem quebrados e, assim, um filho pode ser movido para uma região diferente da dos pais. Considerando, por exemplo, o *crossover* de um ponto ilustrado na Figura 12. O primeiro esquema tem mais chance de ser quebrado do que o segundo no processo de *crossover*, ou seja, o segundo esquema tem mais chance de ser passado para algum dos filhos.

Figura 12 – Crossover de um ponto - esquemas.

Pode-se observar na Figura 12, que o esquema [1 1] do segundo pai, que possui um comprimento menor, é herdado por completo pelo primeiro filho e, se for um esquema com um bom valor *fitness*, esse bom esquema vai ser passado de geração para outra, combinando com outros possíveis esquemas. Já com relação ao Pai 1, o Filho 1 saiu e o Filho 2 também pode sair da região de soluções desse pai. Assim, esquemas de tamanho definido pequeno costuma ficar inalterado na operação de *crossover*, sendo reproduzido com uma taxa alta.

O teorema dos esquemas afirma que esquemas com comprimento definitório e ordem pequenos e valor *fitness* acima da média da população, chamados de *blocos de construção*, têm aumento exponencial de sua participação em gerações consecutivas (Iyoda, 2000), enquanto esquemas contidos em cromossomos com baixa aptidão tendem a desaparecer nas gerações seguintes (de Lacerda and De Carvalho, 1999).

Assim, os algoritmos genéticos buscam um desempenho quase ideal na procura de uma solução ótima através da justaposição dos blocos de construção (Goldberg, 1989). Essa teoria é denominada *Hipótese dos blocos de construção*.

Contudo, combinar blocos construtivos pode levar o algoritmo a convergir erroneamente para um sub-ótimo, processo conhecido como decepção (*deception*) Goldberg et al., 1989. São algoritmos conhecidos como algoritmos genéticos deceptivos (*GA-deceptive*). Isso ocorre pela combinação da definição da codificação com a função de avaliação do *fitness* dos indivíduos que tendem a conter pontos ótimos isolados, isto é, os melhores pontos tendem a ser cercados pelos piores (Goldberg, 1989).

Goldberg et al. (1989) apresentou um exemplo considerando dois esquemas com bons valores *fitness*, $0\,0 * * * * *$ (similar à *string* $0\,0 * * * 1\,1$) e $* * * * * 0\,0$ (similar à *string* $1\,1 * * * 0\,0$), mas que quando combinados geram, por exemplo, o esquema $0\,0 * * * 0\,0$ que possui um *fitness* muito menor do que o seu complementar $1\,1 * * * 1\,1$ que, por sua vez, é um bloco de construção do ponto ótimo $1\,1\,1\,1\,1\,1\,1$.

Uma das formas de contornar esse problema considera um conhecimento prévio da função a ser otimizada e, assim, codificar a solução de maneira diferente para que a operação de *crossover* tenha uma menor probabilidade de combinar blocos construtivos que apresentem bons valores *fitness* quando avaliados de maneira independente mas que,

ao serem combinados, se distanciem muito da solução ótima (Goldberg et al. (1989)). Contudo, Goldberg et al. (1989) comenta que as chances do algoritmo ser codificado com precisão são mínimas, mesmo para combinações de *bits* de ordem baixa. Além disso, a necessidade de um conhecimento prévio de informações nem sempre é viável.

Outra forma é chamada de *inversão*, em que dois pontos do cromossomo são selecionados e a ordem dos *bits* entre essas duas posições é invertida. Goldberg et al. (1989) comentam a necessidade de identificar os *bits* para que depois da inversão, seja possível recuperar a interpretação da *string* sem maiores dificuldades. Os autores apresentam o exemplo da *string* 1 1 0 0 0 1 1, identificando suas posições com uma *string* de pares ordenados ((11)(21)(30)(40)(50)(61)(71)). No exemplo, os pontos escolhidos para inversão estão entre os *bits* 2 e 3 obtendo-se ((11)(21)(71)(61)(50)(40)(30)), o que traz os elementos 1 juntos em um bloco de construção (1 1 1 1 $*$ $**$) mais próximo de uma solução ótima.

Contudo, Goldberg et al. (1985) argumenta que a inversão não tem o poder de justaposição inerente aos operadores binários, que essa operação influencia mais na diversificação do algoritmo genético do que na busca por boas estruturas. Assim, Goldberg et al. (1989) propõe os algoritmos genéticos *messy*, que se baseiam em *strings* de tamanhos variáveis e codificação e operadores *messy*.

2.2.7 Aplicação de algoritmos genéticos em problemas de otimização restritos

Uma das grandes aplicações de algoritmos genéticos é a resolução de problemas de otimização restritos, principalmente os que são difíceis de serem resolvidos computacionalmente por métodos exatos, como problemas lineares e inteiros com restrições. Para problemas de otimização restritos não basta encontrar uma solução cujo com o melhor valor para a função objetivo, é preciso que a solução esteja dentro da região factível do problema. Assim, o espaço de busca para os algoritmos genéticos precisa ou se manter dentro da região de factibilidade ou convergir para uma solução dentro dessa região. Para lidar com problemas de otimização restritos, existem diferentes formas de definir os operadores e lidar com factibilidade das soluções (Michalewicz and Schoenauer, 1996).

Para métodos com operadores que não preservam factibilidade das soluções, pode-se usar funções de penalizações para as infactibilidades e considerá-las no cálculo do valor *fitness* de cada indivíduo. Michalewicz and Schoenauer (1996) cita diversos métodos de penalização, como funções de penalização, penalizações estáticas e dinâmicas, métodos que buscam a factibilidade de soluções para cada restrição em determinada ordem, métodos que assumem que uma solução factível é sempre melhor do que uma não factível (indivíduos infactíveis não podem ser melhores do que os factíveis), um método que repara as infactibilidades dos indivíduos e métodos híbridos que combinam técnicas de computação evolutiva com métodos exatos de resolução de problemas de otimização.

Michalewicz and Schoenauer (1996) comentam que é difícil fazer uma comparação entre as diferentes maneiras de trabalhar com problemas de otimização restrita apresentadas no trabalho, uma vez que os métodos aplicados lidam com diferentes codificações dos problemas (codificação binária ou ponto flutuante) e diferentes operadores. Outra conclusão dos atores é a de que a representação em ponto flutuante dos indivíduos fornece melhores soluções e que não é claro quais características de um problema o torna difícil de ser resolvido por alguma técnica evolutiva, assim como fazer uma escolha apropriada a priori de um método evolutivo para um problema de otimização não linear.

Entre os diferentes métodos usados para lidar com problemas de otimização restritos, encontram-se os algoritmos genéticos de chaves aleatórias e os algoritmos genéticos de chaves aleatórias viciadas, que desenvolvem a geração de indivíduos e a aplicação dos operadores de mutação e *crossover* em vetores de valores gerados aleatoriamente no intervalo $[0, 1]$, ou seja, trabalham no espaço de um hipercubo unitário. Esse vetores, para serem avaliados, são transformados em soluções do problema por meio de decodificadores, que traduzem esses vetores para soluções factíveis para o problema.

2.3 Algoritmos genéticos de chaves aleatórias

Os algoritmos genéticos de chaves aleatórias (*Random Key Genetic Algorithms* - RKGA) são algoritmos implementados em problemas de otimização combinatória envolvendo sequenciamento, cujas soluções podem ser representadas por permutações (Gonçalves and Resende, 2011). Essa classe de algoritmos genéticos foi introduzida por Bean (1994), que utilizou vetores de chaves aleatórias como uma forma de evitar a geração de indivíduos infactíveis na aplicação dos operadores genéticos de mutação e *crossover* (Gonçalves and Resende, 2018). As chaves aleatórias são valores reais gerados no intervalo $[0, 1]$.

Os passos do RKGA são divididos nas partes dependente e independente do problema. A geração dos vetores de chaves aleatórias e aplicação dos operadores genéticos são independentes do problema, pois não olham para o indivíduos como solução do problema. Já o decodificador depende das características do problema para "traduzir" os vetores de chaves aleatórias em soluções e, com elas, calcular o valor *fitness* ou custo de cada indivíduo.

Uma grande vantagem dessa estrutura é que a parte independente é mais flexível, podendo ser usada em diferentes problemas ou para desenvolvimento de outras heurísticas. No caso do RKGA, por exemplo, pode-se manter a parte independente fixa e alterar apenas o decodificador e a função *fitness*, adaptando o algoritmo para diferentes problemas. Os parâmetros básicos da parte independente do problema são o número de genes no cromossomo e os tamanhos da população, da população elite e da população mutante e a probabilidade de um filho herdar o gene de um determinado pai na operação

de *crossover* (Gonçalves and Resende, 2011).

Nos algoritmos RKGA, cada cromossomo é representado por um vetor de tamanho n (parâmetro), cujos elementos são números reais gerados aleatoriamente no intervalo $[0, 1)$, chamados de chaves aleatórias (Gonçalves and Resende, 2018). Para a avaliação dos indivíduos (cromossomos) é preciso "traduzir" esses vetores em uma solução do problema, operação realizada pelo **decodificador**.

O decodificador é um algoritmo determinístico (produz sempre a mesma saída para uma mesma entrada) que mapeia vetores de chaves aleatórias (hipercubo unitário n-dimensional) para soluções no espaço solução do problema. O decodificador pode ser um algoritmo simples, como um ordenamento das chaves aleatórias, por exemplo, envolvendo um mapeamento direto entre as chaves aleatórias e a solução, ou mais complexo, envolvendo heurísticas de busca local que permite que características da busca local passem para próximas gerações (Gonçalves and Resende, 2011). Para decodificadores mais complexos como os que envolvem busca local, Gonçalves and Resende (2018) recomendam a utilização de algum tipo de ajuste no cromossomo para produzir um ótimo local, como utilizado em Resende et al. (2012).

Vamos supor um problema para decidir a ordem na qual cinco produtos devem ser processados em uma determinada máquina, no qual o vetor de chaves aleatórias tem tamanho cinco e cada posição do vetor representa um produto. Na decodificação, ordenamos os valores em ordem crescente e as posições do vetor ordenado fornecem a ordem de processamento dos produtos na máquina (Figura 13). Esse exemplo de decodificador segue a lógica de decodificação proposta por Bean (1994).

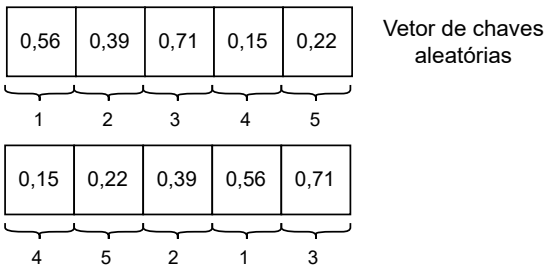


Figura 13 – Sequência de produtos em máquina.

O RKGA inicia com uma população de vetores de chaves aleatórias, com valores aleatórios gerados uniformemente no intervalo $[0, 1)$, que são decodificados em soluções do problema para terem seus valores *fitness* avaliados. A população, então, é particionada em **população elite**, formada pelos p_e indivíduos com melhores valores *fitness*, e **população não elite**, formada pelos $p - p_e$ indivíduos restantes. O tamanho da população, p, no RKGA é constante ao longo das gerações (iterações) e o tamanho da população elite é

menor do que a não elite, resultando em um tamanho menor que a metade da população total (Gonçalves and Resende, 2011):

$$p_e < p - p_e \Rightarrow p_e < \frac{1}{2}p.$$

A partir de uma geração atual, a próxima é composta pela população elite da atual, p_m indivíduos mutantes (em geral, $p_m < 0.5p$) e os $p - p_e - p_m$ restantes resultando do *crossover* uniformemente parametrizado entre dois indivíduos a e b selecionados aleatoriamente dentre todos da população, com a possibilidade de um mesmo pai poder ser escolhido para produzir diferentes herdeiros (escolha aleatória e com reposição). Como qualquer dos indivíduos a ou b podem ser da população elite ou não elite, não é garantido uma maior probabilidade do gene escolhido ser de uma classificação específica. Os indivíduos mutantes, ou imigrantes, não são gerados a partir de mutações em soluções existentes, mas da geração aleatória de p_m novos indivíduos, da mesma forma que a população inicial é gerada (Gonçalves and Resende, 2018).

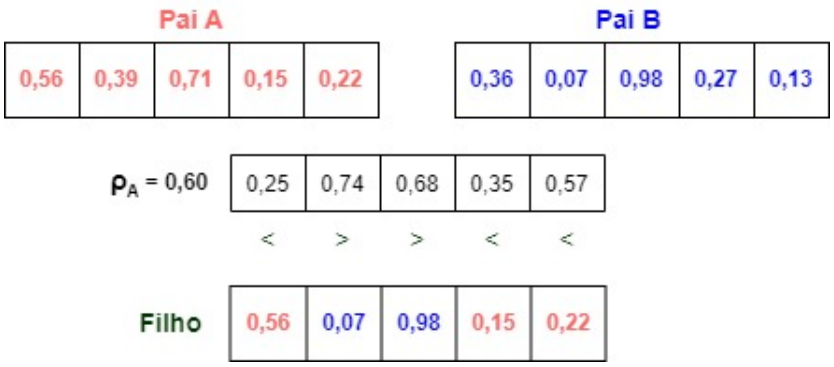


Figura 14 – Exemplo de *crossover* para o RKGA (baseado em Gonçalves and Resende (2011)).

Na Figura 14, dois pais, A e B, são escolhidos de maneira aleatória dentre a população. A probabilidade ρ_e de escolher um gene do Pai A, com um valor gerado de maneira aleatória no intervalo $[0, 1]$ para cada gene (cada posição do vetor de chaves aleatórias), simula o lançamento de uma moeda enviesada. Se o valor aleatório for menor que ρ_e, então o Filho herda o gene do Pai A, caso contrário herda o gene do Pai B.

Como qualquer vetor de chaves aleatórias pode ser decodificado em uma solução do problema, os filhos resultantes do *crossover* vão ser mapeados em soluções do problema, auxiliando na garantia de factibilidade.

Devido à estratégia de copiar a população elite de uma geração para a seguinte, o RKGA é dito um algoritmo genético elitista, mantendo o rastro das boas soluções encontradas ao longo das gerações, resultando em uma *heurística monotonicamente melhorada* (Gonçalves and Resende, 2011). Já a geração da população mutante de maneira aleatória ajuda a evitar cair em ótimos locais, sendo parte do processo de diversificação do algoritmo.

Os critérios de parada para o RKGA podem ser diferentes, como uma quantidade máxima de iterações (gerações), variação pequena entre os *fitness* de indivíduos da população elite, variação pequena entre os melhores *fitness* de indivíduos de gerações consecutivas, tempo de processamento etc.

Diversas variações do algoritmo RKGA foram propostas. Uma variação apresentada em Gonçalves and Resende (2018) considera que o algoritmo inicia com múltiplas populações de mesmo tamanho e geradas independentemente. Uma quantidade de iterações κ é definida de forma que a cada κ iterações, as populações trocam informações e as κ_m melhores soluções de cada população substituem uma determinada quantidade das piores soluções de cada população.

2.4 Algoritmos genéticos de chaves aleatórias viciadas

Os algoritmos genéticos de chaves aleatórias viciadas (*Biased Random Key Genetic Algorithms* - BRKGA) são um tipo de RKGA com diferença na forma como os pais são escolhidos no *crossover*. No RKGA os pais são escolhidos aleatoriamente dentre toda a população, já no BRKGA um dos pais é selecionado da população elite (pai elite) e o outro pai dentre os indivíduos da população não elite ou até mesmo dentre toda a população (Gonçalves and Resende, 2018). Como ocorre no RKGA, a seleção do pai é com reposição, ou seja, um mesmo pai pode gerar mais de um herdeiro.

Além disso, a probabilidade de um filho herdar um gene do pai elite (ρ_e) é maior do que de herdar do pai não elite, isto é, $\rho_e > 0,5$. Dessa forma, a probabilidade da população elite passar suas características para gerações futuras é maior.

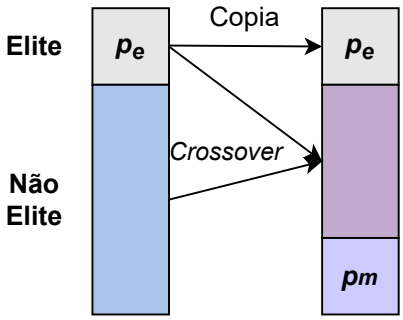


Figura 15 – Formação de uma próxima geração BRKGA, baseada em Gonçalves and Resende (2018).

Um exemplo para o *crossover* do BRKGA pode ser visto na Figura 14, considerando que o Pai A é um indivíduo escolhido dentro da população elite. No caso do exemplo, a probabilidade de o filho herdar um gene do pai não elite (Pai B) é de $1 - \rho_e = 0,4$. Assim, temos uma escolha aleatória enviesada, com vantagem para a herança do gene do pai elite. Essa diferença no elitismo dos algoritmos, gerada pela cópia da população elite

atual para a próxima e a maior probabilidade de pais elite passarem seus genes adiante pelo *crossover*, muda o equilíbrio entre diversificação e intensificação do algoritmo, com a estratégia elitista dando mais força para a intensificação das buscas.

Depois que a próxima geração é criada, os vetores de chaves aleatórias são decodificados em soluções do problemas e avaliados pela função *fitness*, iniciando o processo para produzir a geração seguinte. Assim como no RKGA, o BRKGA também pode ser particionado em partes dependente e independente do problema.

Gonçalves and Resende (2011) fornece um exemplo de BRKGA para o problema de cobertura, considerando cada posição do vetor de chaves aleatórias como uma coluna, que é considerada na solução se a sua chave aleatória for maior que $0,5$. Se o decodificador fosse definido apenas dessa forma, não seria possível garantir que as colunas selecionadas geram uma cobertura. Para completar o processo de decodificação, os autores sugerem um algoritmo guloso padrão do problema de cobertura para completar o conjunto de colunas selecionadas, selecionando colunas que cobrem o maior conjunto de linhas ainda não cobertas, e formar uma cobertura. Depois é feita uma verificação para identificar se há colunas redundantes no conjunto, eliminando-as caso existirem. O valor *fitness* de cada indivíduo é definido pela quantidade de colunas que formam essa cobertura.

Essa parte de decodificação e, depois, a avaliação das soluções decodificadas, são as partes dependentes do problema. As operações de particionamento da população em elite e não elite, a cópia da população elite para a próxima geração, o processo de *crossover* e a geração de indivíduos mutantes, são independentes do problema.

Como o BRKGA é uma variação do RKGA, os passos do algoritmo são os mesmos:

1. população inicial: geração de p vetores de chaves aleatórias;
2. decodificação dos vetores em soluções do problema e cálculo do *fitness*;
3. partição em população elite e não elite;
4. geração de $p - p_e - p_m$ indivíduos pelo *crossover*;
5. geração dos p_m indivíduos mutantes;
6. cópia dos p_e indivíduos elite;
7. com a nova população formada, retorne ao passo 2.

Esses passos são repetidos até algum critério de parada ser satisfeito. Como comentado na seção do RKGA, diferentes critérios de parada podem ser considerados, dependendo de diferentes parâmetros definidos como quantidade máxima de gerações,

quantidade máxima de gerações (iterações) sem melhora no *fitness*, limite de tempo computacional etc.

Os parâmetros do BRKGA seguem os parâmetros do RKGA, considerando que a probabilidade de um filho herdar um gene de um pai A é a probabilidade de escolher um gene do pai elite. Os valores recomendados na literatura como bons parâmetros foram definidos de maneira empírica e estão descritos na Tabela 6.

Descrição	Parâmetro	Valores recomendados
Tamanho da população	p	$p = a \times n$, $1 \leqslant a \in \mathbb{R}$ constante, n o tamanho do cromossomo
Tamanho da população elite	p_e	$0,10 \times p \leqslant p_e \leqslant 0,25 \times p$
Tamanho da população mutante	p_m	$0,10 \times p \leqslant p_m \leqslant 0,30 \times p$
Probabilidade de herdar alelo elite	ρ_e	$0,5 < \rho_e \leqslant 0,8$

Tabela 6 – Base de valores para os parâmetros (Gonçalves and Resende, 2011).

Prasetyo et al. (2015) comentam que em todos os artigos investigados, os parâmetros descritos na Tabela 6 seguem os valores sugeridos e que, para ρ_e, a maioria das pesquisas usaram o valor de $0,7$. Já para o tamanho da população, os autores comentam que o valor varia bastante de acordo com a complexidade e escopo do problema.

A estrutura de algoritmos BRKGAs permite que alguns passos sejam implementados de maneira paralelizada, como a geração da população inicial, a decodificação, a avaliação do *fitness*, a geração de mutantes e a operação de *crossover*, tanto a seleção de indivíduos (por ser com reposição) quanto o cruzamento dos genes. Gonçalves and Resende (2011) comentam que a geração da população inicial e dos indivíduos mutantes, assim como a operação de *crossover* não trazem um impacto tão grande no desempenho computacional como o processo de decodificação. Os autores comentam, também, que uma forma de paralelização é evoluir múltiplas populações de maneira independente.

Dependendo da implementação é possível, também, economizar em alguns cálculos como, por exemplo, a decodificação dos indivíduos da população elite que são transferidos para a geração seguinte como, também, o cálculo dos seus respectivos valores *fitness*.

Alguns artigos que comparam algoritmos do tipo BRKGA com outros algoritmos genéticos são mencionados por Gonçalves and Resende (2011), que comentam que os BRKGAs foram competitivos, retornando soluções tão boas quanto ou melhores, na média, do que os outros algoritmos genéticos comparados.

Assim como para o RKGA, para o BRKGA existem diversas variações, como utilizar alguma heurística para gerar indivíduos para compor a população inicial, como Gonçalves and Resende (2011) comentam, citando o trabalho de Buriol et al. (2005) como

exemplo. Nesse caso, é preciso um codificador que mapeie uma solução do problema em um vetor de chaves aleatórias, o que nem sempre é algo trivial.

Gonçalves and Resende (2018) propuseram um BRKGA com *multi-start*, no qual diversas populações são evoluídas de maneira independente. Nesse processo, alguns indivíduos elite são periodicamente selecionados de cada população para substituir os piores indivíduos de outra e, no fim, a melhor solução dentre todas as populações é retornada como saída do algoritmo. Essa estratégia de múltiplas populações independentes foi inicialmente estudada por Whitley et al. (1999) (Toso and Resende, 2015).

Além disso, os autores usam uma estratégia de *restart*, na qual a população é reiniciada após uma determinada quantidade de iterações (gerações) sem melhora no valor da melhor solução. Os autores comparam resoluções com e sem a estratégia de *restart*, para o problema de cobertura tripla de *Steiner* (Resende et al., 2012), obtendo melhores resultados com o *restart*.

Outro trabalho que utilizou as estratégias de múltiplas populações e *restart* foi Toso and Resende (2015), para acelerar a convergência do algoritmo. Os autores desenvolveram uma API, ou seja, uma interface de programação de aplicação, para implementação do BRKGA que exploram a separação em partes dependente e independente do problema, deixando por conta do usuário apenas a implementação da decodificação das chaves aleatórias em solução do problema. Além disso, a API permite o uso de paralelismo, quando possível, para acelerar o tempo computacional.

Diversas modificações em algoritmos BRKGAs foram propostas e aplicadas em diferentes tipos de problemas. Além de estratégias como *restart*, evolução de múltiplas populações de maneira independente e geração de parte da população inicial com algum tipo de busca (sem ser de maneira aleatória), Prasetyo et al. (2015) comentam outras modificações, como utilização de busca local para melhoria de performance e variação no tamanho da população ao longo das gerações, como proposto no trabalho de Arefi and Rezaei (2015), que reduz o tamanho da população de acordo com o tempo e com a taxa de elitismo.

Os autores comentam que a busca local associada é dependente do problema e citam os trabalhos de Silva et al. (2013), que propuseram um BRKGA para problemas de otimização lineares restritos com as variáveis limitadas, e Silva et al. (2014) que construíram um problema de otimização correspondente à resolução de sistema de equações não lineares com múltiplas soluções, e aplicaram um BRKGA diversas vezes para encontrar diferentes soluções para o sistema, considerando áreas de repulsa associadas a regiões de soluções já encontradas. No algoritmo BRKGA não há utilização de valores ou aproximações de derivadas mas na fase de busca local para melhoria da solução, propuseram um método que pode ser visto como uma aproximação da regra do gradiente, procurando quais soluções em uma vizinhança possuem melhora na função objetivo.

Lucena et al. (2014) propuseram algumas variações no processo de *crossover*. Uma delas é a utilização de múltiplos pais para gerar um herdeiro, combinando os genes desses pais de forma ponderada de acordo com um viés calculado para cada pai, diversificando a combinação dos genes de boas soluções. A outra modificação é a definição de gêneros, atribuindo um *bit* para cada cromossomo, cujo valor definirá qual o gênero do indivíduo. O processo de *crossover*, em si, ocorre da forma clássica e a diferença reside na escolha dos pais, que devem ser escolhidos um de cada gênero.

Prasetyo et al. (2015) comentam que, inicialmente, os algoritmos BRKGAs foram desenvolvidos para problemas de único objetivo, mas modificações feitas ampliaram para aplicações em problemas multiobjetivo. Zheng et al. (2015) propuseram um BRKGA multiobjetivo para o problema de carregamento de contêineres 3D, buscando maximizar o espaço utilizado como a quantidade de contêineres. Além de uma abordagem multiobjetivo para o BRKGA, os autores utilizaram a estratégia de múltipla população e um controlador lógico *fuzzy*.

Outros autores que utilizaram uma abordagem BRKGA multi-objetivo foram Tangpattanakul et al. (2012), para resolver o problema de seleção e escalonamento de requisições de fotografias da Terra por satélite com os objetivos de maximizar o lucro e minimizar a diferença máxima do lucro entre dois usuários.

Modificações do BRKGA ainda são limitados à procedimentos de busca local, que são específicos para cada problema (Prasetyo et al., 2015).

Como é possível observar em Gonçalves and Resende (2011), Gonçalves and Resende (2018) e Prasetyo et al. (2015), os algoritmos RKGA e BRKGA podem ser aplicados em diferentes tipos de problemas, áreas de aplicação como telecomunicação, energia, escalonamento, atribuição, roteamento de veículos, carregamento de contêineres (empacotamento) etc., e diversos problemas de otimização combinatória. Além das diversas aplicações, muitas modificações e melhorias foram propostas para melhorar o desempenho, tanto com relação ao tempo de processamento, utilizando paralelismo, quanto com melhoria nas soluções das populações, como buscas locais em torno das boas soluções de cada população.

Dentre as diversas aplicações estão aplicações do BRKGA em diferentes tipos do problema de alocação de navios em berços.

2.4.1 BRKGA aplicado ao Problema de Alocação em Berços: literatura

Um algoritmo BRKGA foi proposto por Lalla-Ruiz et al. (2014) para o PAB tático, que tem como objetivo alocar posições de atracação nos berços e atribuir perfis de guindastes aos navios, buscando minimizar os custos derivados da organização do fluxo dos contêineres entre os navios (movimentações de contêineres nos pátios) e maximizar o

valor total dos perfis de guindastes atribuídos.

Cada perfil de guindastes fornece quantos guindastes devem ser alocados para um respectivo navio em cada passo de tempo considerado e possui um valor que representa o ganho/perda do uso do respectivo perfil para um navio específico, considerando um *trade-off* entre o terminal e as exigências dos clientes (empresas responsáveis pelos navios). O PAB considerado é o discreto, dinâmico e com janelas de tempo tanto dos berços quanto dos navios.

Assim, o PAB tático considera a qualidade da alocação que depende, além de decisões operacionais como berço ao qual o navio vai ser alocado, da negociação (*trade-off*) entre o terminal e as empresas das linhas de navios, de acordo com as capacidades do porto (Giallombardo et al., 2010). Ou seja, busca integrar o BAP com a alocação de guindastes, considerando regras táticas.

As restrições do problema consideraram as janelas de tempo dos navios e berços, um berço pode atender um único navio em um passo de tempo, cada navio é atendido por um único berço, um único perfil de guindaste pode ser alocado para cada navio, o tempo de serviço de cada navio em cada berço depende da quantidade de guindastes alocados para o navio e, a cada passo de tempo, a quantidade total de guindastes alocados para os navios não ultrapassa o total disponível.

No BRKGA proposto por Lalla-Ruiz et al. (2014) para o problema tático, a população inicial é gerada integralmente de maneira aleatória. A codificação do algoritmo é determinada por cromossomos de tamanho $2 \times n$ (vetores de chaves aleatórias), sendo n a quantidade de navios, com valores gerados aleatoriamente no intervalo $[0, 1)$. As primeiras n chaves aleatórias representam a ordem de atendimento dos navios e as últimas n correspondem à atribuição dos perfis de guindastes aos navios e só depois de determinar a ordem de alocação e a atribuição de guindastes, a decisão do local de atracação (berço) é feita, através de dois algoritmos sequenciais.

A decodificação das primeiras n chaves aleatórias é feita ordenando os valores de forma crescente e adotando as posições das chaves na nova sequência como a ordem (prioridade) de atendimento dos navios. Para a determinação dos perfis de guindaste, o intervalo $[0, 1)$ é dividido pela quantidade de perfis disponíveis para cada navio, com cada parte sendo associada a um perfil. O intervalo no qual a chave aleatória estiver contida, respectivamente na ordem de atendimento definida pela decodificação anterior, determinará qual perfil vai ser atribuído ao navio.

Depois de definir a ordem de alocação e a atribuição dos perfis de guindaste, são definidos o berço e horário de atracação, levando em consideração a ordem de alocação dos navios definida anteriormente. Dois algoritmos são usados, de maneira sequencial, para definir o berço que será alocado para cada navio, satisfazendo restrições espaciais,

temporais e de disponibilidade de guindastes. O primeiro algoritmo tenta alocar o berço com o menor custo de movimentação entre berços (independente do navio), caso o berço não satisfaça alguma restrição, ordena-se os berços em ordem crescente de um custo que leva em conta o fluxo de movimentação de contêineres entre navio que está sendo considerado com navios já alocados. Os berços são atribuídos ao navio nessa ordem, até encontrar algum que satisfaça as condições espaciais e de disponibilidade de guindastes nos passos de tempo. Caso não seja possível alocar nenhum berço a algum navio, passa-se para o próximo algoritmo, que leva em conta a ordem de alocação definida e a ordem dos berços disponíveis, tentando alocar para um próximo navio, o berço seguinte ao alocado para o navio anterior. Caso esse algoritmo também não consiga alocar nenhum berço a algum navio, essa solução é descartada.

Os passos de tempo associados às alocações respeitam o passo de tempo de chegada dos navios, ao momento que o berço estará livre e possível atraso no tempo de alocação para ajustar restrição de disponibilidade de guindastes.

Lalla-Ruiz et al. (2014) compararam o BRKGA com um modelo exato proposto por Giallombardo et al. (2010), uma busca Tabu com *Branch and Price*, proposta pelos mesmos autores, com um algoritmo *Branch and Price* proposto por Vacca et al. (2011) e com a solução proposta por Vacca et al. (2013), um *Branch and Price* com inicialização do problema mestre que usa uma heurística proposta em Giallombardo et al. (2010). As conclusões dos autores descrevem que a performance do BRKGA foi, em geral, melhor do que as dos modelos comparados, seja no tempo computacional, como na qualidade nas soluções factíveis encontradas, sendo o BRKGA menos suscetível à variação na quantidade de perfis de guindastes para os navios. Além de instâncias baseadas nos trabalhos utilizados para comparação, os autores também propuseram geração de novas instâncias, com considerações mais reais, como diferentes custos de operações envolvendo transferência de contêineres entre pátios dos berços.

Para o PAB discreto e dinâmico, modelado como PRVTJ, Pomari (2014) propôs duas decodificações para aplicação do BRKGA e comparou o BRKGA isolado com o BRKGA integrado à um algoritmo de busca de *clusters* (*clustering search*, CS). Na integração CS + BRKGA, o BRKGA entra para gerar soluções para o CS. O modelo utilizado como base, é uma relaxação do PRVJT, penalizando as restrições de janela de tempo dos berços (abertura e fechamento) e dos navios (chegada e partida), proposta por Mauri et al. (2008b). Além da formulação considerada, os autores também utilizaram o processo de busca local para refinamento das soluções utilizadas por Mauri et al. (2008b), e de Oliveira et al. (2012).

Uma das decodificações propostas foi considerar as chaves aleatórias formadas por um valor inteiro no intervalo $[1, m]$, que fornece o berço de alocação, somado a um valor entre $]0, 1]$ que representa quais navios e em qual ordem são alocados aos respectivos

berços definidos pelos valores inteiros. A outra decodificação parte de chaves aleatórias geradas no intervalo $]0, 1]$ que são ordenadas para definir, a partir dos índices do vetor, a ordem dos navios a serem atendidos e a divisão do intervalo das chaves em m subintervalos, cada um representando um berço. O intervalo no qual o valor da chave aleatória cair representa o berço de alocação do navio definido pela posição no vetor.

Nos testes feitos comparando as duas decodificações, os autores concluíram que a primeira decodificação foi menos eficiente devido à limitação que considerar valores inteiros traz, pois resulta em muitos vetores de chaves aleatórias que são decodificados na mesma solução, reduzindo a diversidade no espaço de busca. Comparando o BRKGA isolado com o CS + BRKGA, os autores concluíram que a performance na qualidade da solução e tempo computacional do segundo método foi melhor do que o BRKGA isolado. Os autores comentam, também, que a combinação dos métodos mostrou-se promissora como um método robusto e eficiente na busca por soluções viáveis.

Um algoritmo BRKGA para o BAP discreto e dinâmico, que busca minimizar os tempos de espera dos navios no porto foi proposto por Martin et al. (2015). O algoritmo tem como referência dois modelos exatos, um baseado no Problema de corte bidimensional e outro na ideia de sequenciamento de máquinas, propostos por Barbosa (2014). Na decodificação, os autores definem a ordem de atendimento dos navios e, através de uma programação, definem em qual berço a alocação ocorrerá, alocando os navios, na ordem já definida, no primeiro berço disponível.

A definição do tempo de alocação foi feita de forma a evitar infactibilidade do navio ser alocado antes da sua chegada e evitar sobreposição de atendimento dos navios. Os autores utilizaram a estratégia de reinicialização da população a cada quantidade fixa de gerações e o critério de parada foi a estabilização do valor *fitness* da melhor solução da população.

A comparação do BRKGA com os modelos foi apresentada por curvas *time to target* (Aiex et al., 2007) para avaliação de meta heurísticas, para as quais os *targets* são as melhores soluções obtidas por Barbosa (2014). Os autores concluíram a competitividade do algoritmo BRKGA proposto, tanto em tempo computacional quanto em qualidade da solução, ressaltando a dificuldade do problema em ser resolvido por métodos exatos.

Rizzi et al. (2015) propuseram a combinação do algoritmo de busca de *clusters* (*Clustering search* - CS) com o BRKGA para o PAB tático, considerando a minimização do custo de transferência de cargas entre posições no pátio (para cargas que são transferidas de um navio para outro) e a otimização da utilização dos guindastes. Nessa proposta o BRKGA é utilizado para gerar as soluções que passam pelo processo de agrupamento do método CS. O modelo matemático utilizado como base de comparação é o proposto por Giallombardo et al. (2010), que considera a alocação de navios aos berços e a alocação de perfis de guindastes aos navios. Os autores também comparam os resultados com o

trabalho de Lalla-Ruiz et al. (2014), que utilizou o BRKGA isolado e não combinado com outra meta-heurística.

A codificação considera vetores de chaves aleatórias de tamanho $2 \times n$, com n sendo a quantidade de navios, e as chaves aleatórios no intervalo $[0, 1]$. Para a decodificação, as primeiras n posições são ordenadas de maneira crescente definindo a ordem de atendimento dos navios e as n últimas definem o perfil de guindaste alocado a cada navio. O intervalo $[0, 1]$ é dividido em subintervalos de acordo com a quantidade de perfis e o intervalo no qual a chave aleatória pertencer vai definir o respectivo perfil de guindaste que será atribuído.

A alocação dos berços é feita de acordo com o custo de armazenamento do berço, alocando o máximo de navios possível ao berço com menor custo. Para determinar o próximo berço que será alocado para o navio corrente, uma função que considera o custo de armazenagem e o custo do fluxo de contêineres do respectivo navio para os outros navios associados que já foram alocados.

Nos resultados, os autores comentam que o CS + BRKGA obtiveram uma pequena melhora em algumas soluções obtidas pelo BRKGA proposto por Lalla-Ruiz et al. (2014), obtendo resultados similares para outras instâncias. Os autores também concluíram que o método proposto se mostrou robusto e um bom método para a resolução do TBAP.

Correcher and Alvarez-Valdes (2017) propuseram o método BRKGA com múltiplas populações desenvolvidas independentemente para a resolução do PAB contínuo e dinâmico com alocação de guindastes invariante no tempo, isto é, a alocação não muda durante o processamento do navio. Os autores comentam que, na literatura, a versão invariante no tempo recebe menos atenção que a versão na qual a alocação de guindastes pode variar ao longo do atendimento do navio. Para o problema de guindastes, os autores consideraram as variações que alocam a quantidade de guindastes (BACAP), considerando limites mínimo e máximo de guindastes permitidos para cada navio, e a variação que também especifica quais guindastes devem ser alocados (BACASP). O tempo de atendimento dos navios está associado à quantidade de guindastes alocados para o navio e não à sua posição de alocação e é considerado um parâmetro de entrada.

O custo considera o tempo de espera dos navios até serem atendidos, o atraso na liberação dos navios (de acordo com o horário máximo de permanência dos navios no porto e a extrapolação do horizonte de planejamento) e um custo pelo desvio da alocação da posição desejada no cais. A posição desejada de alocação no cais é de acordo com a posição no pátio na qual a carga será armazenada e o desvio é a diferença entre a posição desejada e a posição alocada para o navio. Os autores admitem soluções nas quais os navios podem ser atendidos para além do horizonte de planejamento, penalizando o tempo adicional.

O BRKGA desenvolvido para o BACAP considera cada cromossomo formado por duas listas, uma de chaves aleatórias entre 0 e 1 cujos índices da ordenação não decrescente fornecem os navios, e a outra lista com valores que representam a quantidade de guindastes alocados para os navios, geradas no intervalo entre as quantidades mínima e máxima definidas. Um algoritmo construtivo gera uma solução factível, alocando um navio por vez de acordo com a ordem já estabelecida, definindo o local de atracação, o horário e a quantidade (factível) de guindastes.

Com o resultado do algoritmo construtivo, um esquema memético é aplicado olhando para os guindastes ociosos e alocando eles aos navios de acordo com a ordem de de partida desejada. Com isso, os tempos de atendimento são reduzidos e os próximos navios a serem atendidos podem iniciar o atendimento antes, podendo reduzir o atraso no horário de partida. A nova lista de alocação de guindastes é alimentada no algoritmo construtivo, que é aplicado novamente e, se uma solução melhor é encontrada, o cromossomo é atualizado com a nova quantidade de guindastes alocados.

As populações iniciais não são geradas totalmente aleatórias, elas são geradas de acordo com regras de prioridade definidas pelas características do problema, como horário de chegada dos navios, tamanho, horário máximo preferível de partida, tempos máximo e mínimo de atendimento, maior e menor área (tempos máximo e mínimo multiplicados pelo respectivo tamanho do navio) e à folga de tempo do horário que chega, somando o tempo de atendimento, até o horário máximo desejado. Os navios são ordenados de maneira crescente ou decrescente de acordo com essas regras, fornecendo soluções iniciais boas e diversificadas. A quantidade de regras não é suficiente para gerar toda a população, assim o resto dos indivíduos são gerados de forma aleatória.

As operações genéticas utilizadas pelos autores foram o *crossover*, imigração (ou mutação) e migração. O *crossover* é o uniforme (clássico), com viés na escolha do gene, dando preferência para o gene do pai elite, e é aplicado de maneira independente na primeira e na segunda metade do cromossomo, ou seja, na parte que define a ordem de atendimento dos navios e na alocação da quantidade de guindastes. A imigração é o equivalente à população mutante, gerada de maneira aleatória, e a migração é feita a cada quantidade fixa de gerações e consiste em copiar o melhor indivíduo dentre todas as populações para outras que não o contenham e excluindo o pior indivíduo delas, para manter o tamanho das populações fixo.

O critério de parada é um limite no tempo de execução. Depois, uma busca local é feita com o intuito de refinar as melhores soluções do BRKGA. Para a busca local, foram propostos três procedimentos diferentes, todos baseados na movimentação dos navios dentro do planejamento e são aplicados em todos os indivíduos de cada população resultante do BRKGA. Das três, duas são heurísticas e a terceira é uma *mateurística* (procedimento híbrido de método exato com heurístico). O critério de parada das buscas

locais é tempo limite de execução.

A extensão do método de solução do BACAP para lidar com o BACASP parte do princípio de que é possível, sob certa condição, obter uma solução ótima para o segundo problema a partir de uma ótima do primeiro problema correspondente em tempo polinomial. Türkoğulları et al. (2014) definem sequências de navios ordenados de maneira não decrescente com relação às suas posições no cais como completas quando, tomando os navios dois a dois nessa sequência, os pares se sobrepõem em pelo menos um período de tempo na alocação. Uma sequência completa é dita própria quando a soma dos guindastes das alocações de toda a sequência é menor ou igual ao total de guindastes disponíveis. Türkoğulları et al. (2014) provaram que, dada uma solução ótima para o BACAP, χ^*, pode-se se obter em tempo polinomial uma solução ótima para o BACASP se toda sequência completa extraída de χ^* for própria e propuseram um algoritmo polinomial para encontrar uma solução ótima do BACASP quando a condição determinada é satisfeita. Para os casos em que não é satisfeita, propuseram um algoritmo de plano de corte baseado no modelo BACAP proposto por eles.

O ajuste no BRKGA para encontrar soluções para o BACASP é feito no algoritmo construtivo, que passa a aceitar apenas soluções que não formem sequências impróprias e, assim, as buscas locais baseadas no algoritmo construtivo não precisam de alteração. Contudo, a busca local baseada na *mateurística* passa a aplicar o algoritmo de plano de corte ao modelo para o BACAP proposto por Correcher et al. (2017) e, depois, aplica o algoritmo proposto por Türkoğulları et al. (2014) para encontrar uma solução para o BACASP.

As instâncias geradas foram baseadas nas descritas em Meisel and Bierwirth (2009) e Park and Kim (2003) e chegam a considerar até 100 navios, rodando os experimentos para o BACAP e para o respectivo BACASP. O BRKGA proposto foi comparado com um modelo linear inteiro proposto por Correcher et al. (2017) e os autores puderam concluir o bom desempenho do algoritmo genético proposto tanto com relação a tempo computacional quanto a qualidade da solução, ressaltando que é possível indicar que o BRKGA performou uma boa exploração do espaço de busca.

Para o caso do BACAP, um tempo limite menor de busca já foi suficiente para o algoritmo convergir bem, indicando que um aumento no tempo pudesse não influenciar tanto na qualidade da solução. Já para o BACASP, um aumento no tempo computacional permitiu uma melhora considerável nas soluções, alcançando novas soluções ótimas para algumas instâncias.

Além de comparar com um modelo linear inteiro misto, os autores compararam com um algoritmo de evolução discreta diferencial (*Discrete Differential Evolution Algorithm* - DDE), que os autores comentam apresentar um bom desempenho para problemas de escalonamento. O esquema seguido foi o mesmo proposto por Pan et al. (2007) com a

diferença de não aplicarem uma busca local interna e que a população inicial é gerada da mesma forma que no BRKGA. Os autores concluíram que o BRKGA performou melhor, em geral do que o algoritmo DDE, exceto por algumas instâncias em que encontrou soluções um pouco melhores que o BRKGA, tanto para o BACAP quanto para o BACASP.

3 Clusterização

A clusterização (ou análise exploratória de dados, Xu and Wunsch (2005)) é uma técnica de aprendizagem de máquina (*machine learning*) que busca particionar um conjunto de dados em diferentes grupos, ou *clusters*, de maneira que dados que pertençam ao mesmo grupo sejam similares entre si e distintos dos outros grupos. Ou seja, busca otimizar maximizando a similaridade entre indivíduos do mesmo grupo e a dissimilaridade entre diferentes grupos. A clusterização é uma técnica de aprendizagem não supervisionado, pois não há um conjunto de entradas e saídas conhecidas para que o modelo possa aprender, ou seja, classifica resultados a partir de dados não rotulados.

Um exemplo de aplicação de clusterização é no *marketing*, onde a partir de um conjunto de clientes dos quais se tem acesso a diferentes informações (como renda, região da residência etc.) e se quer classificar quais clientes são mais receptivos a determinados produtos, para guiar quais anúncios serão apresentados para essas pessoas (James et al., 2013). Outras aplicações podem ser para estimar valores faltantes em um conjunto de dados ou identificar *outliers* (Nerurkar et al., 2018). Além de aplicações em diversos outros campos, como segmentação de imagens, genética, buscas em internet, sendo uma importante ferramenta para classificar, visualizar e entender melhor os dados (Xu and Wunsch, 2005, Oyelade et al., 2019, Nerurkar et al., 2018).

Dependendo dos dados que se tem disponível e de quais características se quer agrupar os dados, é preciso definir quais devem ser consideradas e como elas devem ser tratadas no processo de clusterização. A escolha de quais é importante para poder descrever os dados com foco no que se quer medir, sem considerar características que possam guiar o processo de agrupamento para uma direção diferente. Outra decisão é como será medida a similaridade entre indivíduos, utilizando as características deles.

A similaridade pode ser através do conceito de distância (Euclidiana, por exemplo), densidade ou distribuição de probabilidade, por exemplo (Nerurkar et al., 2018). Através da escolha do conceito de similaridade, de acordo com o conjunto de dados com o qual se está trabalhando, espera-se que os indivíduos agrupados em um mesmo *cluster* tenham uma maior similaridade (ou menor variabilidade) entre si e uma maior dissimilaridade (ou maior variabilidade) com os indivíduos de outros *clusters*. Além das formas de se definir a similaridade, existem diferentes tipos de clusterização mas, como comenta Nerurkar et al. (2018), não há um consenso em qual é melhor tipo ou algoritmo, uma vez que cada um tem suas suposições e vieses.

Oyelade et al. (2019) cita diferentes tipos de algoritmos de clusterização, dentre eles a clusterização hierárquica (*hierarquical clustering*), a particionada (*partitional*

clustering), clusterização baseada em grade (*grid based clustering*), baseada em densidade (*density based clustering*) e a *soft clustering*, no qual os dados podem pertencer a diferentes *cluster* com um certo grau de pertencimento.

Os algoritmos de clusterização particionada parte do princípio em que cada objeto (ou indivíduo, ou dado) pertence a um único *cluster*, cada *cluster* deve possui pelo menos um objeto e o número de *clusters*, K, é predefinido pelo usuário (*K-partitioning algorithms*) (Nerurkar et al., 2018). São algoritmos baseados no centro geométrico de cada *cluster*, ou seja, o centro geométrico do conjunto de dados pertencentes ao *cluster*, atualizando iterativamente o valor desses centros de acordo com a atualização dos dados designados para cada *clusters* através da distância dos dados até o centro de cada *cluster* (Nerurkar et al., 2018, Xu and Tian, 2015). São algoritmos que partem de uma solução inicial e, a partir dela, calculando os centros dos *clusters*, vão realocando os dados de acordo com os centros mais próximos, convergindo, assim, para ótimos locais, usando como definição de similaridade a distância entre pontos (dados) e centros geométricos.

Por serem algoritmos que partem de soluções iniciais, eles são sensíveis a elas, ou seja, diferentes soluções iniciais guiam para diferentes resultados. Uma estratégia pode ser executar os algoritmos algumas vezes, com soluções iniciais diferentes e selecionar a melhor solução (James et al., 2013). Por serem algoritmos baseados no centro do *cluster* tendem a gerar *clusters* esféricos, sendo uma técnica não aconselhada para dados não convexos, além de ser sensível a dados *outliers* por obrigatoriamente inseri-los em *clusters* com o centroide mais próximo, mesmo sendo diferentes (Xu and Tian, 2015). Como a quantidade de *clusters* deve ser pré definida, existem diversas estratégias para estimar a quantidade ideal, como métodos empíricos e o método do cotovelo que leva em conta a distância média dos centroides.

Um dos algoritmos mais conhecidos desse tipo de clusterização é o *K-means*, baseado na distância Euclidiana entre os pontos e que busca minimizar o quanto cada indivíduo varia dos outros que estão no mesmo *cluster*, resultando em um problema de otimização (minimização). O método *K-medóides* (*K-medoids*) é um desenvolvimento do *K-means* que para lidar com dados discretos (Xu and Tian, 2015).

A clusterização hierárquica busca construir uma relação hierárquica entre os dados baseada na similaridade entre eles, agrupando dados aos mais próximos, construindo *subclusters* e formando uma estrutura de árvore (chamada de dendrograma), que pode ser construída no sentido *bottom-up* ou *top-down* (Xu and Wunsch, 2005). A forma de construção dessa árvore permite que a decisão da quantidade de *clusters* seja com base no quanto se deseja limitar a dissimilaridade dos dados dentro de cada *cluster*, podendo cortar a árvore na altura que forneça a melhor construção de *clusters*, não sendo necessário decidir a quantidade *a priori*.

Os algoritmos hierárquicos podem ser classificados como aglomerativos (*bottom*

up), que inicia com cada dado formando um *cluster* (folhas da árvore) próprio e vai conectando (construindo nós) pares de objetos até conectar todos os dados (raiz da árvore), ou divisivos (*top down*) que desenvolve de maneira inversa, iniciando com todos os dados em um único *cluster* (raiz) e segue dividindo em *clusters* até que cada dado se torne um único (folhas) (Xu and Wunsch, 2005). Os algoritmos divisivos não são muito utilizados por serem caros computacionalmente, devido a quantidade de possibilidades de divisão que se pode fazer (Xu and Wunsch, 2005).

Os algoritmos hierárquicos não precisam iniciar com a quantidade de *clusters* pré determinado, a escolha dessa quantidade pode ser feita através do nível de similaridade máximo que se considera aceitável dentro de cada *cluster*, que pode ser percebido na altura do dendrograma.

A ideia da clusterização baseada em *grid* é transformar o espaço de dados em uma estrutura de grade (*grid*) e agrupa os dados em células da grade, em que cada célula representa uma região do espaço. Ou seja, divide o espaço de dados em regiões menores e que podem ser processadas de forma independente (favorável para paralelizar) (Oyelade et al., 2019, Xu and Tian, 2015). Os algoritmos mais comuns desse tipo de clusterização são o *STING* e *OPTGRID* (Oyelade et al., 2019).

Nos métodos de clusterização considera os *cluster* como regiões densas de objetos no espaço dos dados, sendo divididos por regiões de baixa densidade e possibilitando formas arbitrárias (diferentemente do *partitional clustering* que gera formatos esféricos), podendo formar as regiões com base em vizinhança ou funções de densidade (Xu and Tian, 2015, Nerurkar et al., 2018, Oyelade et al., 2019). O algoritmo baseado em densidade mais conhecido é o *DBSCAN*, que consegue identificar *clusters* que não são linearmente separáveis, utiliza a distância Euclidiana para calcular a distância entre os pontos no espaço e se baseia no conceito de vizinhança, e não necessita da quantidade de *clusters* predefinida. Tem um bom desempenho com conjuntos de dados grandes mas não para os que possuem alta dimensionalidade (Xu and Tian, 2015, Nerurkar et al., 2018, Oyelade et al., 2019).

No presente trabalho, o método selecionado para ser inserido no processo do BRKGA foi o método hierárquico aglomerativo, implementado utilizando a biblioteca *Sklearn* na linguagem *Python*. Com essa ferramenta, foi possível definir para esse método de clusterização, uma matriz de distância própria, além de não ser necessário definir previamente a quantidade de *clusters* e poder defini-los por um grau de similaridade entre os indivíduos considerados.

3.1 Algoritmo de clusterização Aglomerativo

O método de clusterização hierárquico aglomerativo constrói uma árvore, denominada dendrograma, baseado na dissimilaridade ou distância entre os indivíduos. A raiz do dendrograma representa todo o conjunto de dados e cada folha um dado, os nós intermediários indicam que, no respectível nível da árvore, os *clusters* ligados por esse nó representam menor dissimilaridade (menor distância) corrente e a altura normalmente representa a distância entre cada par de *clusters* ligados pelo nó (que pode estar ligando dois dados, um dado com um *cluster* ou dois *clusters*) (Xu and Wunsch, 2005).

A definição de distância entre os dados garante a primeira parte do algoritmo aglomerativo, quando cada *cluster* é formado por um único indivíduo. Contudo, é preciso definir também como medir a distância entre grupos de indivíduos, ou seja, entre *clusters* com mais de um indivíduo, denominada *linkage* (ou ligação). Os tipos mais comuns de *linkage* são: ligação completa *complete*, simples (*single*), média (*average*) e centróide (*centroid*) (James et al., 2013).

- Ligação completa: calcula a distância entre todos os pares de indivíduos dos dois *clusters* e define a ligação como o valor máximo dessas distâncias (máxima dissimilaridade entre os *clusters*).
- Ligação simples: cálculo equivalente ao da completa mas ao invés de adotar o valor máximo, adota o mínimo (mínima dissimilaridade).
- Ligação média: definida pelo valor médio das distâncias entre todos os pares de indivíduos dos *clusters*.
- Ligação dos centróides: distância entre os centróides de cada um dos *clusters*, ou seja, a distância entre os pontos médios de cada *cluster*, onde o ponto médio de cada *cluster* é a média das distâncias entre seus indivíduos.

Assim, com a distância e o tipo de ligação definidos, é possível construir o dendrograma. Na Figura 16 temos o dendrograma gerado para cada um dos tipos de ligação completa, média e simples, utilizando uma população aleatória gerada pelo BRKGA para o PAB, considerando como medida de dissimilaridade a quantidade de alocações diferentes que têm entre uma solução e outra (como descrito na Seção 4.4). Ou seja, quanto maior o valor, mais diferentes são uma alocação da outra.

Como os dendrogramas foram gerados a partir da clusterização utilizando a biblioteca *Sklearn* e esta não possui o tipo de ligação média implementada, ela não está sendo considerada na figura. Essa biblioteca possui outro método de ligação disponível, que é a ligação *ward*, que só pode ser implementada com a distância Euclidiana e como a

matriz de distância utilizada foi pré definida de acordo com os cálculos da Seção 4.4, esse tipo de ligação também não está descrita na Figura 16.

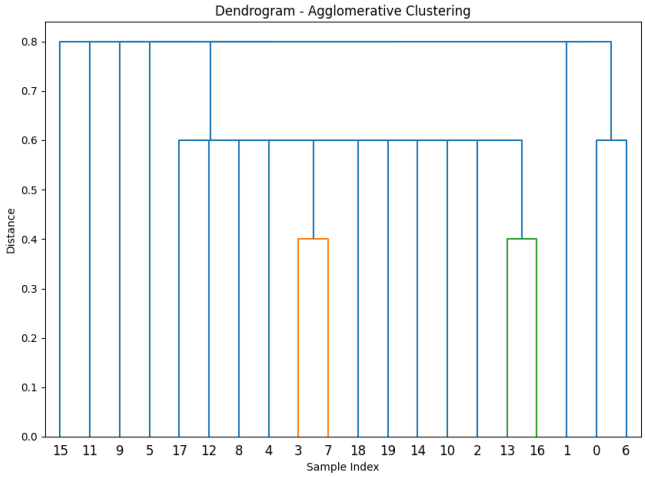
Ao escolhermos um valor de dissimilaridade para fazer o corte e determinar a quantidade de *clusters*, significa que queremos adotar os *clusters* cujos indivíduos possuem no máximo a dissimilaridade escolhida. Nos dendrogramas, é possível visualizar traçando uma reta na altura do corte selecionado e verificar os *clusters* formados pelos nós abaixo do corte.

Analisando a Figura 16, podemos perceber que cada tipo de ligação gera diferentes dendrogramas e, escolhendo um nível de similaridade para fazer o corte para definir os *clusters* como, por exemplo, $0,5$, podemos ver que, para a ligação simples, teríamos 18 *clusters*, dois *clusters* contendo dois indivíduos (cores laranja e verde) e os outros formados por um único dado. Se fizéssemos o corte em $0,7$, teríamos 7 *clusters* com 13 indivíduos, outro com 2 e o resto com um único indivíduo cada.

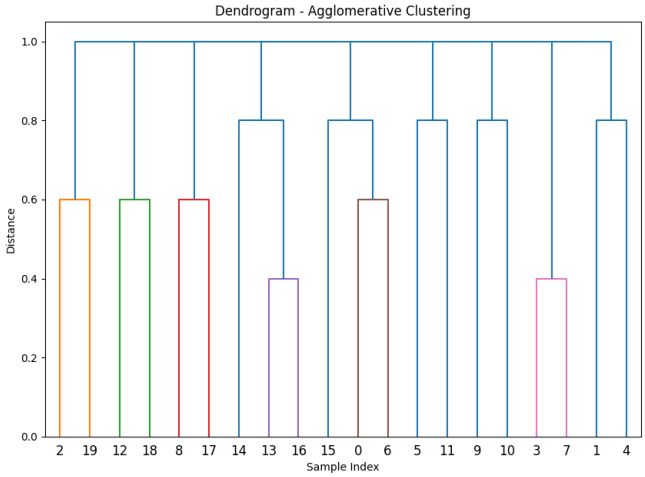
Já para a ligação completa, o corte de $0,5$ também gera 18 *clusters*, dois contendo dois indivíduos e o resto com um único cada. Para o corte em $0,7$, isto é, definindo que os indivíduos de um mesmo *cluster* podem ter uma dissimilaridade de no máximo $0,7$ (no máximo 70% de alocações diferentes), teríamos 14 *clusters*, seis com dois indivíduos e o resto com um único indivíduo cada.

Para a ligação média, definindo o corte em $0,5$, também gera 18 *clusters*, dois com dois indivíduos cada e o resto com um único e, para o corte em $0,7$, foram gerados 14 *clusters*, sete com dois indivíduos cada e o resto com um único. Se definirmos o corte em $0,85$, aí conseguiríamos reduzir para seis *clusters*, quatro com três indivíduos e dois com 4.

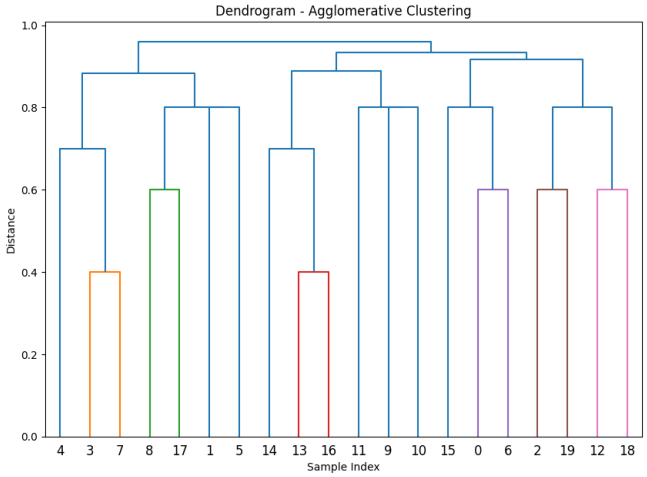
Para esse exemplo, podemos notar, então, que para reduzir a quantidade de *clusters* precisamos aumentar o grau de dissimilaridade escolhido, ou seja, precisamos aceitar indivíduos mais diferentes entre si.



(a) Ligação simples.



(b) Ligação completa.



(c) Ligação média.

Figura 16 – Exemplos de dendrogramas para uma população gerada pelo BRKGA para o PAB.

4 Modelos para o PAB implementados

Neste capítulo apresentamos dois modelos clássicos para resolver o PAB discreto e dinâmico de forma exata, o *Dinamic Berth Allocation Problem*, DBAP, proposto por Imai et al. (2001), e um modelo baseado no Problema de Roteamento de Veículos (PRVJT), proposto por Cordeau et al. (2005). Apresentamos, também, a proposta de variação do modelo baseado no PRVJT para considerar, também, a minimização do tempo de ociosidade dos berços, além de um método BRKGA, meta-heurístico, para resolver tanto o modelo baseado no PRVJT quanto a variação proposta.

Todos os modelos consideram que:

- os navios podem chegar após a abertura dos berços (problema dinâmico);
- os berços são representados por um conjunto discreto (problema discreto);
- cada berço pode atender apenas um navio por vez e pode ter momentos ociosos (sem atender nenhum navio);
- os tempos que cada navio leva para ser atendido em cada berço são fixos e fornecidos, assim como os horários de chegada dos navios e abertura dos berços;
- o atendimento de cada navio deve ser feito completamente por um único berço, isto é, um navio não pode ser atendido parcialmente por diferentes berços;
- o objetivo dos modelos é minimizar o tempo total de permanência dos navios no porto, considerando tempo de espera e tempo de atendimento.

Os modelos implementados, apesar de diferentes, possuem parâmetros em comum:

- N: conjunto de navios, $n = |N|$;
- M: conjunto de berços, $m = |M|$;
- h_{ik}: duração do atendimento do navio $i \in N$ no berço $k \in M$;
- a_i: horário de chegada do navio $i \in N$;
- s_k: horário de abertura do berço $kinM$.

4.1 Modelo *Dinamic Berth Allocation Problem*, DBAP

Imai et al. (2001) propôs o modelo *DBAP* (*Dinamic Berth Allocation Problem*) baseado no problema estático, proposto por Imai et al. (1997). No DBAP, as variáveis são responsáveis por definir onde e em qual ordem os navios são alocados e o tempo ocioso dos berços. O modelo considera a ordem de alocação dos navios mas não inclui o instante de tempo em que serão alocados como variável. Além disso, os navios podem ser atendidos em qualquer berço. A função objetivo desse modelo busca minimizar o tempo total de permanência dos navios no porto, isto é, tempo de espera acrescido do tempo de atendimento.

Além dos parâmetros comuns para os modelos, apresentados no início deste capítulo, temos o conjunto de ordens de serviço e o subconjunto de navios que chegam antes do berço estar disponível:

- P: conjunto de ordens de serviço (ou posições de serviço), $|P| = n$;
- $P(p)$: subconjunto ordens (ou posições) de serviço antes de p, isto é,

$$P(p) = \{q \in P \,|\, q < p\} \subseteq P, p \in P;$$

- $N(k)$: subconjunto de navios $j \in N$ tais que $a_i \geqslant s_k$. Ou seja,

$$N(k) = \{i \in N \,|\, a_i \geqslant s_k\} \subseteq N, k \in M.$$

As variáveis do modelo são:

- y_{ipk}: fornece o tempo ocioso do berço k entre a partida do $(p-1)$-ésimo navio e a chegada do p-ésimo navio no berço, se o navio i é o p-ésimo a ser servido.
- $x_{ipk} \in \{0, 1\}$, $\forall i \in N$, $\forall p \in P$, $\forall k \in M$,

$$x_{ipk} = \begin{cases} 1 & \text{se navio } i \text{ é o } p\text{-ésimo navio atendido pelo berço } k, \\ 0 & \text{caso contrário.} \end{cases}$$

minimizar

$$Z_{DBAP} = \sum_{k\in M}\sum_{i\in N}\sum_{p\in P}\left[(n-p+1)h_{ik} + s_k - a_i\right]x_{ipk} + \sum_{k\in M}\sum_{i\in N(k)}\sum_{p\in P}(n-p+1)y_{ipk} \tag{4.1}$$

sujeito a

$$\sum_{k\in M}\sum_{p\in P} x_{ipk} = 1, \qquad \forall\, i \in N \tag{4.2}$$

$$\sum_{i\in N} x_{ipk} \leqslant 1, \qquad \forall\, k \in M,\, p \in P \tag{4.3}$$

$$\sum_{l\in N}\sum_{q\in P(p)} (h_{lk}\, x_{lqk} + y_{lqk}) + y_{ipk} \geqslant (a_i - s_k)x_{ipk}, \qquad \forall\, k \in M,\, i \in N(k),\, p \in P \tag{4.4}$$

$$x_{ipk} \in \{0,1\}, \qquad \forall\, k \in M,\, i \in N,\, p \in P \tag{4.5}$$

$$y_{ipk} \geqslant 0, \qquad \forall\, k \in M,\, i \in N,\, p \in P \tag{4.6}$$

A função objetivo (4.1) busca minimizar o tempo total de espera e atendimento de cada navio no porto, assim como os tempos de ociosidade dos berços. Como o tempo de atendimento de um navio em um berço afeta o tempo de espera dos navios atendidos posteriormente, o termo $(n-p+1)$ multiplica o valor de tempo de atendimento do navio no berço para as próximas posições de atendimento do berço, ou seja, já adiciona seu tempo de atendimento como tempo de espera para os próximos navios a serem atendidos nesse berço.

O conjunto de restrições (4.2) garante que cada navio é atendido por um único berço. As restrições (4.3) garantem que para uma mesma posição em um mesmo berço pode ter, no máximo, um navio alocado.

O conjunto de restrições (4.4) é responsável por garantir que os navios sejam atendidos depois dos seus respectivos horários de chegada, considerando um tempo ocioso do berço, y_{ipk}. Se um navio i é o p-ésimo atendido num berço k e chega um tempo depois que o $(p-1)$-ésimo navio atendido deixou o berço k, então haverá um tempo ocioso do berço, isto é, $y_{ipk} > 0$.

O lado esquerdo de (4.4) é a soma dos tempos de atendimento dos navios atendidos anteriormente ao navio i, com o tempo ocioso do berço antes da chegada desse navio, ou seja, o tempo de duração entre s_k e o início do serviço do navio i. Se o navio já está no berço no momento em que os atendimentos anteriores terminam, então a função objetivo força o tempo ocioso do berço entre o $(p-1)$-ésimo e o p-ésimo navios a serem atendidos a ser zero, caso contrário, o valor de ociosidade do berço será a_i menos o tempo total de atendimento dos navios atendidos anteriormente (considerando os respectivos tempos de ociosidade).

Imai et al. (2001) supõem que os parâmetros h_{ik}, s_k e a_i são inteiros, pois a integralidade da função objetivo reduz o esforço computacional quando a solução ótima é encontrada pelo método do sub gradiente. Devido à função objetivo e ao conjunto de restrições (4.5), as variáveis y_{ipk} também serão inteiras, por mais que sejam definidas como contínuas no modelo.

Cordeau et al. (2005) adicionou um conjunto de restrições para considerar a janela de tempo dos berços:

$$\sum_{i \in N} \sum_{p \in P} (h_{ik} x_{ipk} + y_{ipk}) \leqslant e_k - s_k, \ \forall k \in M. \tag{4.7}$$

A função objetivo busca minimizar o tempo total de permanência dos navios no porto e o tempo de ociosidade dos berços. Mas a função, como está definida nessa formulação, não está correta. Este fato fica mais claro com um exemplo.

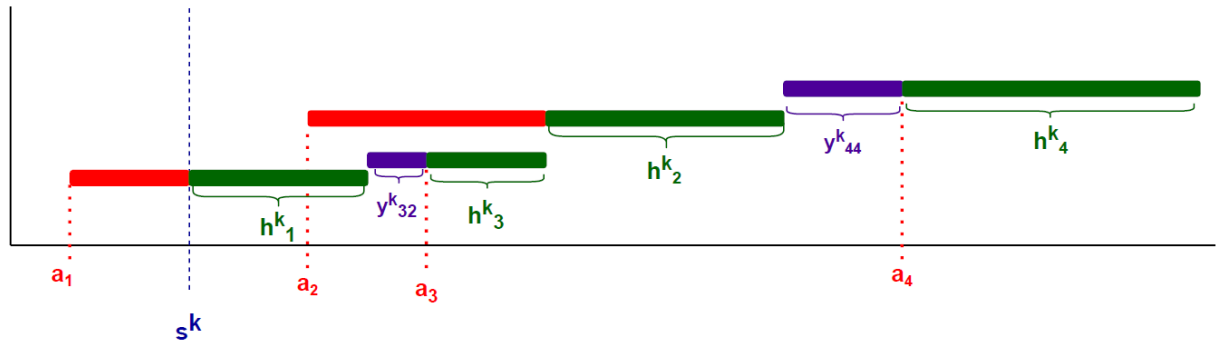


Figura 17 – Exemplo de alocação em um berço k.

Na Figura 17, a cor vermelha representa o tempo de espera do navio, do momento que chega até ser atendido; a cor verde representa o tempo de atendimento no berço; a cor roxa representa o período de ociosidade do berço até que um navio inicie seu atendimento.

De acordo com a Figura 17, suponhamos uma alocação de quatro navios, do total de 5 navios ($n = 5$), em um berço k, tal que:

- $a_1 = 1$, $a_2 = 5$, $a_3 = 7$ e $a_4 = 15$;
- $h_{1k} = 3$, $h_{2k} = 4$, $h_{3k} = 2$ e $h_{4k} = 5$;
- $s_k = 3$;
- $x_{11k} = x_{23k} = x_{32k} = x_{44k} = 1$;
- $y_{32k} = 1$, $y_{44k} = 2$.

Para essa alocação, considerando $n = 5$, temos que a primeira parte da função objetivo fica:

$$\sum_{k\in M}\sum_{i\in N}\sum_{p\in P}\left[(5-p+1)h_{ik}+s_k-a_i\right]x_{ipk} = \sum_{k\in M}\sum_{i\in N}\sum_{p\in P}\left[(4-p)h_{ik}+s_k-a_i\right]x_{ipk}$$

Considerando, agora para o exemplo de alocação:

- navio 1 (primeiro alocado): $(4-1)*3+3-1=11$;
- navio 2 (terceiro alocado): $(4-3)*4+3-5=2$;
- navio 3 (segundo alocado): $(4-2)*2+3-7=0$;
- navio 4 (quarto alocado): $(4-4)*5+3-15=-12$;
- tempos de ociosidade: $y_{32k}=1$, $y_{44k}=2$;
- total: $11+2+0-12+1+2=4$.

Calculando, agora, os tempos de permanência dos navios no porto de acordo com os horários de chegada dos navios e abertura dos berços, temos:

- navio 1: chega no tempo 1 e é atendido quando o porto abre, no tempo 3. Assim, há um tempo de espera de
$$\underbrace{3}_{s_k} - \underbrace{1}_{a_1} = 2$$
unidades de tempo, mais o tempo de atendimento no berço, que é 3. Logo, o tempo total de permanência do primeiro navio no porto é 5. O tempo que o navio deixa o porto é
$$\underbrace{1}_{a_1} + \underbrace{2}_{\text{espera}} + \underbrace{3}_{\text{atendimento}} = 6.$$
- Navio 3: chega no tempo 7 e é atendido no momento que chega no porto, mas o berço termina o atendimento do navio anterior no tempo 6. Assim, temos um tempo de ociosidade do berço de 1 unidade de tempo. Como o navio é atendido no momento que chega no porto, ele permanece no porto somente o tempo de atendimento dele no berço, que é de 2 unidades de tempo.
- Navio 2: chega no tempo 5 mas só foi alocado em
$$\underbrace{6}_{\text{saída navio 1}} + \underbrace{1}_{\text{ociosidade berço}} + \underbrace{2}_{\text{atendimento navio 3}} = 9,$$
que é o horário que o berço estará disponível. Então, há um tempo de espera de $9-5=4$. Somando o tempo de atendimento no berço, o Navio 2 permanece no

porto por $4+4=8$ unidades de tempo. O atendimento desse navio no berço termina no tempo $9+4=13$.

- Navio 4: chega no tempo 15 e o navio atendido anteriormente deixa o berço no tempo 13, logo há um tempo de ociosidade do berço de $15-13=2$ unidades de tempo. Como esse navio é atendido no momento que chega no porto, o tempo de permanência dele é de 5 unidades de tempo.

Resultando em um tempo total de permanência dos navios no porto, considerando a ociosidade do berço:

$$\underbrace{2+3}_{\text{Navio 1}} + \underbrace{1}_{\text{ociosidade}} + \underbrace{2}_{Navio3} + \underbrace{4+4}_{Navio2} + \underbrace{2}_{\text{ociosidade}} + \underbrace{5}_{\text{Navio 4}} = 23.$$

Esse valor total difere do valor resultando da função objetivo para essa alocação.

Com relação aos tempos de ociosidade dos berços, como a função objetivo minimiza os tempos de ociosidade, garante-se o menor tempo possível de ociosidade entre atendimentos dos navios. Olhando, por exemplo, para o conjunto de restrições (4.4), para o terceiro navio no berço k, por exemplo:

$$(3*1+0)+y_{32k} \geqslant 7-3=4, \text{ isto é, } y_{32k} \geqslant 1.$$

Podemos perceber, assim, como o conjunto de restrições (4.4) define as variáveis de ociosidade, juntamente com a minimização da função objetivo. Considerando que a função é de minimização, isso garante que $y_{32k}=1$.

O termo $n-p+1$ na função objetivo do modelo parte da última posição possível de atendimento e subtrai a ordem de atendimento em que o navio é alocado, considerando as posições de atendimento após a posição que o navio foi atendido.

Hansen and Oğuz (2003) sugeriram uma forma de correção no modelo para lidar com o erro da função objetivo, que é considerar a ordem reversa nos índices de posição de atendimento. Assim, eles consideraram que a variável x_{ipk} representa o navio i sendo o $p-$ésimo último navio atendido no berço k, com mudanças na função objetivo.

Os autores também comentam, na formulação do modelo estático, que o erro na função objetivo acaba não cobrindo a restrição de que, para ter um navio alocado na posição p, é preciso que outro navio tenha sido alocado na posição $p-1$. Para resolver essa questão, mantendo a formulação como a desenvolvida por Imai et al. (2001), os autores sugeriram o seguinte conjunto de restrições:

$$\sum_{i\in N} x_{ipk} \leqslant \sum_{i\in N} x_{i\,(p-1)\,k}, \ \forall k \in M, p \in P\backslash\{1\}. \tag{4.8}$$

De fato, como a função objetivo soma de forma acumulada o tempo de atendimento de cada navio para as próximas posições de atendimento como tempo de espera dos navios atendidos, quanto maior a posição de atendimento, menos unidades do tempo de atendimento do navio são consideradas. Considerando, por exemplo, um total de 5 navios para serem alocados e um navio i alocado em um berço k na posição de atendimento 2. Dessa forma, o tempo de atendimento dele será acumulado para as posições de atendimento $2, 3, 4$ e 5 mas, se o navio for alocado na posição de atendimento 4, ele só vai acumular o seu tempo de atendimento para as posições 4 e 5.

Na implementação feita, testando instâncias simples, foi possível observar a necessidade desse conjunto de restrições também para o modelo dinâmico.

Os autores de Imai et al. (2001) lançaram o *corrigendum* Imai et al. (2005a), ajustando a interpretação das variáveis do modelo:

- y_{ipk}: fornece o tempo ocioso do berço k entre a partida do $(n-p+2)$-ésimo último navio e a chegada do $(n-p+1)$-ésimo último navio no berço, se o navio i é o $(n-p+1)$-ésimo último navio a ser servido.
- $x_{ipk} \in \{0,1\}$, $\forall i \in N$, $\forall p \in P \, \forall k \in M$,

$$x_{ipk} = \begin{cases} 1 & \text{se navio } i \text{ é o (n-p+1)-ésimo último navio atendido pelo berço } k, \\ 0 & \text{caso contrário.} \end{cases}$$

4.2 Modelo para o PAB baseado no PRVJT

O BAP modelado como um Problema de Roteamento de Veículos com Janela de Tempo e múltiplos depósitos (Cordeau et al., 2005), cada depósito possuindo um único veículo, considera que os depósitos como os berços e os navios como os clientes a serem atendidos. Contudo, Buhrkal et al. (2011) comenta que o modelo também pode ser interpretado como um Problema de Roteamento de Veículos Heterogêneo com Janela de Tempo, considerando um único depósito com múltiplos veículos diferentes, que pode ser definido através de um grafo, em vez de multi grafo sem afetar, contudo, a complexidade do algoritmo.

As considerações para esse modelo, além das apresentadas anteriormente, incluem que todo navio pode ser atendido em qualquer berço do porto.

4.2.1 Modelo proposto na literatura

O problema de roteamento pode ser representado por um multigrafo $G = (V, A)$, com $V = \{v_0, v_1, \ldots, v_n\}$ o conjunto de vértices e $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ o conjunto de arcos. O vértice v_0 representa o depósito e os outros vértices representam os clientes a

serem atendidos. A cada cliente estão associados uma carga (demanda), duração de serviço e uma janela de tempo e a cada arco em A está associado um custo de tempo de viagem.

Além dos parâmetros mencionados no início deste capítulo, esse modelo possui mais alguns parâmetros:

- e_k: horário de fechamento do berço $k \in M$;
- b_i: horário de término da janela de tempo para navio $i \in N$;
- ν_i: valor (custo) do tempo de serviço do navio $i \in N$, usado também para definir prioridade de atendimento;
- $M_{ijk} = \max\{b_i + h_{ik} - a_j, 0\}$, $\forall k \in M, \forall (i,j) \in A^k$.

A demonstração de que a fórmula para M_{ijk} garante que seu valor é grande o suficiente para não causar infactibilidades erradas pode ser vista em Peteam (2016).

Variáveis:

- $x_{ijk} \in \{0,1\}$, $\forall k \in M$, $\forall (i,j) \in A^k$,

$$x_{ijk} = \begin{cases} 1 & \text{se navio } j \text{ é atendido após navio } i \text{ pelo berço } k, \\ 0 & \text{caso contrário;} \end{cases}$$

- T_{ik}, $\forall k \in M$, $\forall i \in N$: horário que o navio i atracou no berço k;
- $T_{o(k)\,k}$, $\forall k \in M$: horário que o primeiro navio atracou no berço k;
- $T_{d(k)\,k}$, $\forall k \in M$: horário que o último navio saiu do berço k.

Os índices $o(k)$ e $d(k)$ indicam o índice do primeiro navio a chegar no berço k e o índice do último navio a sair desse berço (vértices de origem e destino), respectivamente.

O modelo matemático que descreve o PAB por um PRVJT é dado por:

minimizar

$$Z = \sum_{i \in N} \sum_{k \in M} \nu_i \left[T_{ik} - a_i + h_{ik} \sum_{j \in N \cup \{d(k)\}} x_{ijk} \right] \tag{4.9}$$

sujeito a

$$\sum_{k \in M} \sum_{j \in N \cup \{d(k)\}} x_{ijk} = 1, \qquad \forall\, i \in N \tag{4.10}$$

$$\sum_{j \in N \cup \{d(k)\}} x_{o(k)\, j\, k} = 1, \qquad \forall\, k \in M \tag{4.11}$$

$$\sum_{i \in N \cup \{o(k)\}} x_{i\, d(k)\, k} = 1, \qquad \forall\, k \in M \tag{4.12}$$

$$\sum_{j \in N \cup \{d(k)\}} x_{ijk} - \sum_{j \in N \cup \{o(k)\}} x_{jik} = 0, \qquad \forall\, k \in M,\ \forall\, i \in N \tag{4.13}$$

$$T_{ik} + h_{ik} - T_{jk} \leqslant (1 - x_{ijk}) M_{ijk}, \qquad \forall\, k \in M,\ \forall\, (i,j) \in A^k \tag{4.14}$$

$$T_{ik} \geqslant a_i, \qquad \forall\, k \in M,\ \forall\, i \in N \tag{4.15}$$

$$T_{ik} + h_{ik} \sum_{j \in N \cup \{d(k)\}} x_{ijk} \leqslant b_i, \qquad \forall\, k \in M,\ \forall\, i \in N \tag{4.16}$$

$$T_{o(k),\, k} \geqslant s_k, \qquad \forall\, k \in M \tag{4.17}$$

$$T_{d(k),\, k} \leqslant e_k, \qquad \forall\, k \in M \tag{4.18}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall\, k \in M,\ \forall\, (i,j) \in A^k \tag{4.19}$$

$$T_{ik} \in \mathbb{R}^+, \qquad \forall\, k \in M,\ \forall\, i \in N. \tag{4.20}$$

Na função objetivo descrita em (4.9), o termo $T_i^k - a_i$ representa o tempo de espera do navio antes de ser atendido, e o termo que se segue (com somatório) representa o tempo de atendimento do navio no berço ao qual ele foi alocado, resultando no tempo total de permanência dos navios no porto. O elemento ν_i multiplicando o tempo total do navio i no porto representa o peso de acordo com a prioridade de atendimento do navio.

As restrições (4.10) garantem que cada navio é atendido por um único berço, (4.11) e (4.12) garantem que cada berço tem um único primeiro e um único último navios, respectivamente. O conjunto de restrições (4.13) garante o fluxo de navios em cada berço, um navio i atendido em um berço k é precedido por outro navio e sucedido por outro navio, atendidos também no berço k.

As restrições (4.14)–(4.18) estão associadas ao horário de atendimento dos navios. O conjunto (4.14) garante a não sobreposição de navios nos berços (um navio só pode começar a ser atendido em um berço depois que seu predecessor no berço terminou de ser atendido), (4.15) garante que os navios são alocados só depois do horário de chegada determinado, (4.16) garante que os navios devem ser liberados até o horário máximo de permanência permitido para eles no porto e as restrições (4.17) e (4.18) garantem que

o primeiro e últimos navios, respectivamente, devem ser atendidos dentro da janela de tempo dos berços.

4.2.2 Modelo implementado

No modelo de roteamento de veículos, o primeiro e o último vértices para cada veículo representam o depósito ao qual o veículo pertence. Na modelagem apresentada acima, esse papel é representado pelo conjunto $\{o(k), d(k)\}$, indicando os primeiros e últimos n navios alocados ao berço k, $k = 1, \cdots, m$. Para a implementação desses índices, foram criados m navios fictícios representando os primeiros e m representando os últimos navios atendidos em cada berço, de forma que $o(k) = k$ e $d(k) = n + m + k$.

Assim, os primeiros índices de navios $\{1, \ldots, m\}$ e os últimos $\{m + n + 1, \ldots, 2 \cdot m + n\}$ representam os navios fictícios que marcam os primeiros e últimos navios de cada berço, respectivamente. Os índices $\{m + 1, \ldots, m + n\}$ são os índices que representam os navios reais do problema.

Os tempos de atendimento dos navios fictícios foram definidos como zero, assim como o horário de chegada dos navios fictícios. A permanência máxima dos primeiros navios fictícios foi definida com o horário de abertura dos respectivos berços e, para os últimos, foi definida com o horário de fechamento.

Além das adequações associadas às considerações dos navios fictícios, as restrições referentes às equações 4.11 e 4.12 foram alteradas para considerar as variáveis de um berço serem nulas caso o berço não atenda nenhum navio.

O modelo implementado apresenta a seguinte forma:

$$Z = \min \sum_{i \in N} \sum_{k \in M} \nu_i \left[T_{i,k} - a_i + h_{ik} \sum_{j \in \cup d(k)} x_{ijk} \right] \tag{4.21}$$

sujeito a:

$$\sum_{k \in M} \sum_{j \in N \cup d(k), j \neq i} x_{ijk} = 1, \qquad \forall\, i \in N, \tag{4.22}$$

$$\sum_{j \in N \cup d(k)} x_{o(k)\, j\, k} \leqslant 1, \qquad \forall\, k \in M, \tag{4.23}$$

$$\sum_{i \in N \cup o(k)} x_{i\, d(k)\, k} \leqslant 1, \qquad \forall\, k \in M, \tag{4.24}$$

$$\sum_{j \in N \cup d(k), j \neq i} x_{ijk} - \sum_{j \in N \cup o(k), j \neq i} x_{jik} = 0, \qquad \forall i \in N, \forall\, k \in M, \tag{4.25}$$

$$T_{ik} + h_{ik} - T_{jk} \leqslant (1 - x_{ijk}) M_{ijk}, \quad \forall\, k \in M, \forall\, i \in N \cup o(k), \forall\, j \in N \cup d(k),\ j \neq i, \tag{4.26}$$

$$T_{ik} \geqslant a_i, \qquad \forall\, i \in N, \forall\, k \in M \tag{4.27}$$

$$T_{ik} + h_{ik} \sum_{j \in N \cup d(k)} x_{ijk} \leqslant b_i, \qquad \forall\, k \in M, \forall\, i \in N \tag{4.28}$$

$$T_{o(k)\, k} = s_k, \qquad \forall\, k \in M, \tag{4.29}$$

$$T_{d(k)\, k} \leqslant e_k, \qquad \forall\, k \in M \tag{4.30}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall\, k \in M, \forall\, i \in N \cup o(k), \forall j \in N \cup d(k). \tag{4.31}$$

$$T_{ik} \in \mathbb{R}^+, \qquad \forall\, k \in M, \forall\, i \in N. \tag{4.32}$$

As desigualdades 4.23 e 4.24 permitem os navios fictícios assumirem o valor nulo caso o berço não atenda nenhum navio. As restrições 4.22 garantem que todo navio real é atendido em algum berço, 4.25 são as restrições de fluxo. 4.26 garantem a não sobreposição nos horários de atendimento dos navios e os valores definidos para $M_{i,j,k}$ permanecem os mesmos propostos por Cordeau et al. (2005). O conjunto de restrições 4.27 garantem que todo navio real será atendido depois da sua chegada no porto, enquanto 4.29 forçam o horário de atendimento dos primeiros navios fictícios serem iguais à abertura dos respectivos berços. As restrições 4.28 garantem que os navios reais são atendidos até o horário máximo de permanência no porto e 4.30 garante que o último navio fictício é atendido antes do fechamento do respectivo berço, com esses dois conjuntos de restrições, garantimos que os navios reais também são atendidos antes do fechamento dos berços.

4.2.3 Instância teste

Para validação, algumas instâncias foram montadas para testar o comportamento do modelo diante de algumas características na distribuição dos dados.

Um exemplo de instância testada foi a seguinte:

- $n = 3$ e $m = 2$;
- $a = [1, 5, 13]$ e $b = [17, 16, 22]$;
- Tempo de atendimento dos navios nos berços:

$$H = \begin{bmatrix} 9 & 10 \\ 7 & 7 \\ 4 & 4 \end{bmatrix};$$

- $s = [5, 0]$ e $e = [24, 24]$.

Como a implementação foi feita na linguagem *Python* e os índices partem do 0, temos que os navios fictícios de índices 0 e 5 representam o primeiro e último do berço 0 e os navios 1 e 6 representam o primeiro e último navios do berço 1. Os navios reais do problema são representados pelos índices $2, 3$ e 4. Assim, a instância que o modelo resolve fica da forma:

- $a = [0, 0, 1, 5, 13, 0, 0]$ e $b = [5, 5, 17, 16, 22, 24, 24]$;
- Tempo de atendimento dos navios nos berços:

$$H = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 9 & 10 \\ 7 & 7 \\ 4 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

A alocação resultante do modelo foi:

- Para o berço 0: $x_{0,3,0} = x_{3,5,0} = 1$.
- Tempos de atendimento no berço 0: $T_{0,0} = 0$ e $T_{3,0} = 5$.
- Para o berço 1: $x_{1,2,1} = x_{2,4,1} = x_{4,6,1} = 1$.
- Tempos de atendimento no berço 1: $T_{2,1} = 1$, $T_{4,1} = 13$.
- Valor objetivo: 21

Segue na Figura 18 a representação dessa solução:

Analisando a Figura 18, percebemos que uma solução alternativa seria a alocação do navio 4 no berço 0 no mesmo passo de tempo 13. Como o tempo de atendimento desse

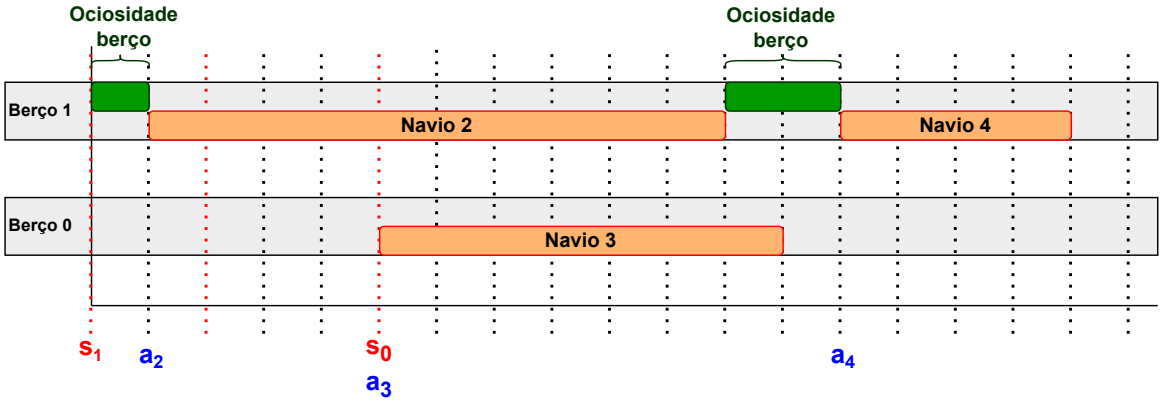


Figura 18 – Solução do modelo para a instância exemplo.

navios nos dois berços é o mesmo, assim como o horário de alocação, o valor objetivo final dessa solução alternativa seria o mesmo que da solução acima. A diferença estaria no tempo de ociosidade total pois, como o navio 3 termina seu tempo de atendimento no berço um passo de tempo a mais que o navio 2, resultaria em um tempo de ociosidade menor e, assim, o berço 1 poderia ter suas funções finalizadas antes, como pode ser observado na Figura 19.

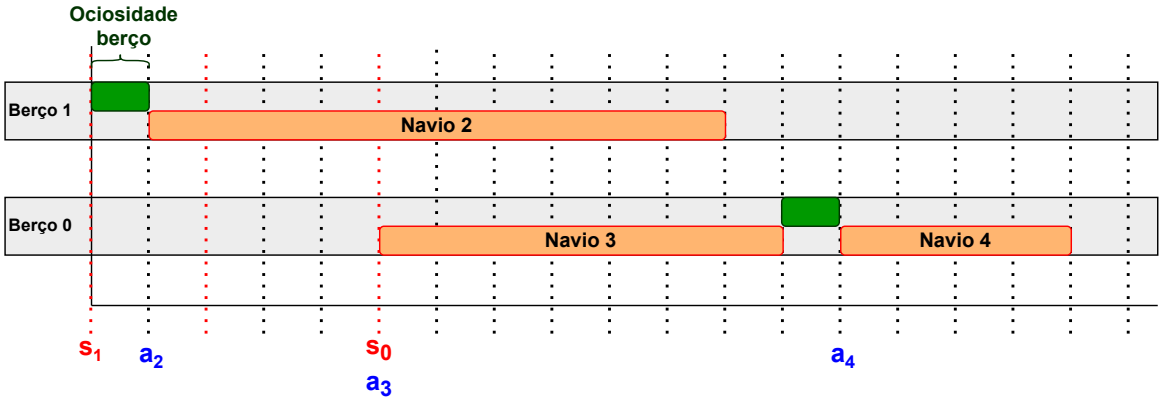


Figura 19 – Solução alternativa.

A função objetivo considerando a minimização dos tempos de permanência dos navios no porto prioriza a visão dos clientes, tornando os navios disponíveis mais rapidamente. Considerar, por exemplo, a ociosidade dos berços no modelo traz uma visão de aproveitamento dos tempos do berço, buscando um interesse do porto, liberando os berços mais rapidamente.

Assim, uma extensão do modelo que leva em consideração o tempo de ociosidade dos berços foi proposta, como descrito na Seção 4.2.4.

4.2.4 Extensão do modelo PRVJT: modelo PRVJT-O

Para considerar o tempo de ociosidade no modelo, precisamos inserir uma nova variável ($y_{i,j,k}$) inteira e não negativa, que representa o tempo de ociosidade entre as alocações dos navios i e j no berço k.

Uma restrição para a definição dessa variável como a diferença entre o tempo de alocação de um navio com a soma do tempo de alocação e de atendimento do navio anterior precisa ser inserida. Quando não houver diferença nesses tempos, é preciso garantir que não haverá infactibilidade entre os valores, incorrendo na necessidade de definir um valor *big-M*.

$$\overline{M}_{i,j,k} = \max\{b_j - (a_i + h_{i,k}), 0\}. \tag{4.33}$$

O conjunto de restrições a ser inserido, além do de não negatividade, é dado por:

$$T_{j,k} - (T_{i,k} + h_{i,k}) \leqslant (1 - x_{i,j,k})\overline{M}_{i,j,k} + y_{i,j,k} \tag{4.34}$$

O valor definido em 4.33 é grande o suficiente para garantir a factibilidade da restrição. De fato:

1. $x_{i,j,k} = 1$: da restrição 4.26 temos que $T_{j,k} - (T_{i,k} + h_{i,k}) \geqslant 0$ e, assim, $y_{i,j,k} \geqslant T_{j,k} - (T_{i,k} + h_{i,k}) \geqslant 0$. Ou seja, definimos um limite inferior para a variável de tempo de ociosidade igual à diferença entre o passe de tempo em que o navio anterior deixou o berço e quando o navio corrente é atendido, que corresponde ao intervalo de ociosidade do berço entre o atendimento desses dois navios.

2. $x_{i,j,k} = 0$: navios i e j não são atendidos consecutivamente no berço k e o valor de $y_{i,j,k}$ deve ser definido como 0. Para isso, devemos ter na restrição que $y_{i,j,k} \geqslant 0$ ou que $y_{i,j,k} \geqslant -L$, $L \geqslant 0$, ou seja, maior ou igual ao um valor negativo. Dessa forma, junto com a restrição de não negatividade e com o objetivo de minimização, o valor definido para a variável de ociosidade será 0.

   a) os navios i e j não são atendidos no berço k e, assim, $T_{i,k} = a_i$ e $T_{j,k} = a_j$:

      i.

$$\overline{M}_{i,j,k} = 0 \Rightarrow b_j - (a_i + h_{i,k}) \leqslant 0 \tag{4.35}$$

      De 4.34 temos

$$\begin{aligned} y_{i,j,k} \geqslant T_{j,k} - (T_{i,k} + h_{i,k}) - \overline{M}_{i,j,k} &= T_{j,k} - (T_{i,k} + h_{i,k}) \\ &= a_j - (a_i + h_{i,k}). \end{aligned}$$

E, como $a_j - (a_i + h_{i,k}) \underbrace{\leqslant}_{a_j \leqslant b_j} b_j - (a_i + h_{i,k}) \underbrace{\leqslant}_{Eq.\,4.35} 0$, temos que $y_{i,j,k} \geqslant -L$, $L \geqslant 0$, onde $L = |a_j - (a_i + h_{i,k})|$.

ii.

$$\overline{M}_{i,j,k} = b_j - (a_i + h_{i,k}) \geqslant 0 \tag{4.36}$$

De 4.34 temos

$$\begin{aligned} y_{i,j,k} \geqslant T_{j,k} - (T_{i,k} + h_{i,k}) - \overline{M}_{i,j,k} &= T_{j,k} - (T_{i,k} + h_{i,k}) - b_j + (a_i + h_{i,k}) \\ &= a_j - b_j \underbrace{\leqslant}_{a_j \leqslant b_j} 0. \end{aligned}$$

Assim, sendo $L = -(a_j - b_j)$, temos que $y_{i,j,k} \geqslant -L$, $L \geqslant 0$.

b) navio i é atendido em k mas j não. Assim, $T_{i,k} \geqslant a_i$ e $T_{j,k} = a_j$:

i.

$$\overline{M}_{i,j,k} = 0 \Rightarrow b_j - (a_i + h_{i,k}) \leqslant 0 \tag{4.37}$$

De 4.34 temos

$$\begin{aligned} y_{i,j,k} \geqslant T_{j,k} - (T_{i,k} + h_{i,k}) - 0 &= a_j - (T_{i,k} + h_{i,k}) \\ \underbrace{\leqslant}_{a_j \leqslant b_j} b_j - (T_{i,k} + h_{i,k}) &\underbrace{\leqslant}_{-T_{i,k} \leqslant -a_i} b_j - a_i - h_{i,k} \\ &\underbrace{\leqslant}_{Eq.4.37} 0. \end{aligned} \tag{4.38}$$

Temos, então, que $y_{i,j,k} \geqslant -L$, $L \geqslant 0$, para $L = -(b_j - (a_i + h_{i,k}))$.

ii.

$$\overline{M}_{i,j,k} = b_j - (a_i + h_{i,k}) \geqslant 0 \tag{4.39}$$

De 4.34 temos

$$\begin{aligned} y_{i,j,k} \geqslant T_{j,k} - (T_{i,k} + h_{i,k}) - \overline{M}_{i,j,k} &= a_j - (T_{i,k} + h_{i,k}) - b_j + a_i + h_{i,k} \\ &= \underbrace{a_j - b_j}_{\leqslant 0} + \underbrace{(a_i - T_{i,k})}_{\leqslant 0} + \underbrace{h_{i,k} - h_{i,k}}_{=0} \leqslant 0. \end{aligned}$$

Assim, $y_{i,j,k} \geqslant -L$, $L \geqslant 0$, onde $L = -(a_j - b_j + a_i - T_{i,k})$.

c) navio i não é atendido em k mas j é. Assim, $T_{i,k} = a_i$ e $T_{j,k} \geqslant a_j$:

i.

$$\overline{M}_{i,j,k} = 0 \Rightarrow b_j - (a_i + h_{i,k}) \leqslant 0 \tag{4.40}$$

De 4.34 temos

$$\begin{aligned} y_{i,j,k} \geqslant T_{j,k} - (T_{i,k} + h_{i,k}) - 0 &= T_{j,k} - (a_i + h_{i,k}) \\ &\underbrace{\leqslant}_{T_{j,k} \leqslant b_j} b_j - (a_i + h_{i,k}) \underbrace{\leqslant}_{Eq.4.40} 0. \end{aligned}$$

Para $L = -(b_j - (a_i + h_{i,k})) \geqslant 0$, temos $y_{i,j,k} \geqslant -L$.

ii.

$$\overline{M}_{i,j,k} = b_j - (a_i + h_{i,k}) \geqslant 0 \tag{4.41}$$

De 4.34 temos

$$\begin{aligned} y_{i,j,k} &\geqslant T_{j,k} - (T_{i,k} + h_{i,k}) - (b_j - (a_i + h_{i,k})) \\ &\underbrace{=}_{T_{i,k} = a_i} T_{j,k} - (a_i + h_{i,k}) - b_j + (a_i + h_{i,k}) \\ &= T_{j,k} - b_j \underbrace{\leqslant}_{T_{j,k} \leqslant b_j} 0. \end{aligned}$$

Assim, temos que $y_{i,j,k} \geqslant -L$, onde $L = -(T_{j,k} - b_j) \geqslant 0$.

d) navios i e j são atendidos em k mas não consecutivamente. Assim, $T_{i,k} \geqslant a_i$ e $T_{j,k} \geqslant a_j$:

i.

$$\overline{M}_{i,j,k} = 0 \Rightarrow b_j - (a_i + h_{i,k}) \leqslant 0 \tag{4.42}$$

De 4.34 temos

$$\begin{aligned} y_{i,j,k} \geqslant T_{j,k} - (T_{i,k} + h_{i,k}) - 0 &\underbrace{\geqslant}_{T_{j,k} = b_j} b_j - (T_{i,k} + h_{i,k}) \\ &\underbrace{\geqslant}_{T_{i,k} \geqslant a_i} b_j - (a_i + h_{i,k}) \underbrace{\leqslant}_{Eq.4.42} 0. \end{aligned}$$

Logo, para $L = -(b_j - (a_i + h_{i,k})) \geqslant 0$, temos $y_{i,j,k} \geqslant -L$.

ii.

$$\overline{M}_{i,j,k} = b_j - (a_i + h_{i,k}) \geqslant 0 \tag{4.43}$$

De 4.34 temos

$$\begin{aligned} y_{i,j,k} &\geqslant T_{j,k} - (T_{i,k} + h_{i,k}) - (b_j - (a_i + h_{i,k})) \\ &= T_{j,k} - T_{i,k} - b_j + a_i - h_{i,k} + h_{i,k} \\ &= \underbrace{T_{j,k} - b_j}_{\leqslant 0} + \underbrace{a_i - T_{i,k}}_{\leqslant 0} \leqslant 0. \end{aligned}$$

Assim, temos que $y_{i,j,k} \geqslant -L$, $L \geqslant 0$, onde $L = -(T_{j,k} - b_j)$.

O modelo PRVJT-O é definido, então, pelas mesmas restrições do modelo PRVJT, acrescido do conjunto dado por 4.34. De maneira equivalente para a função objetivo, com a soma do termo:

$$\sum_{k \in M} \sum_{i \in N \cup o(k)} \sum_{j \in N} y_{i,j,k} \tag{4.44}$$

O termo 4.44 na função objetivo pode ter um impacto menor, pois seu valor pode ser consideravelmente menor do que a soma dos tempos dos navios no porto. Assim, para buscar uma maior representatividade da ociosidade do berço, um peso pode ser considerado para esse termo na função objetivo do modelo ou poderíamos, também, normalizar os parâmetros da função objetivo.

4.3 BRKGA proposto para resolução do PAB sem e com ociosidade de berços

O BRKGA apresentado nesta seção foi proposto para a resolução do PAB, considerando uma decodificação que define qual a ordem e em qual berço os navios serão alocados e, posteriormente, o horário em que essa alocação acontece, de acordo com os cálculos propostos por Mauri et al. (2008a).

4.3.1 Decodificação

A decodificação proposta define tanto a ordem em que os navios serão atendidos, quanto em quais berços, de maneira independente, para buscar uma melhor variabilidade na geração de soluções. A definição da ordem de alocação está associada a uma parte do vetor de chaves aleatórias enquanto a definição de qual berço é feita a partir de outra parte do vetor. Depois de definidas a ordem e o local de atendimento dos navios, os horários de alocação são definidos, de maneira a manter factíveis as restrições de horário de chegada dos navios, abertura dos berços e não sobreposição no atendimento.

Os indivíduos da população são formados por cromossomos (vetores de chaves aleatórias) de tamanho $2 \times n$, ou seja, o dobro da quantidade de navios, com os n primeiros elementos (primeira metade) definindo a ordem de alocação e os n últimos (segunda metade) definindo o berço. Gonçalves and Resende (2011) comentam um exemplo em que geram indivíduos com o dobro do tamanho de alguma dimensão, com a primeira metade do vetor definindo a ordem e, a segunda metade, o estado das variáveis do problema.

As posições iniciais da primeira metade do vetor estão associadas ao respectivo índice de cada navio, ou seja, o primeiro elemento representa o primeiro navio, etc. Cada índice (navio) dessa metade do vetor está associado a um índice da segunda, definindo uma relação de navio com berço. Ou seja, o índice (navio) i da primeira metade do vetor vai estar associado ao índice $n + i$ da segunda metade.

Os valores das chaves da primeira metade do vetor são ordenados em ordem não decrescente e a nova posição dos índices indica a ordem em que os respectivos navios serão atendidos. A segunda metade do vetor segue essa reordenação e, assim, os valores que definem o berço de cada navio acompanham a nova posição daquele navio no vetor. Essa operação é ilustrada na Figura 20, para um exemplo com 3 navios e 2 berços, onde as cores identificam o par (navio, berço).

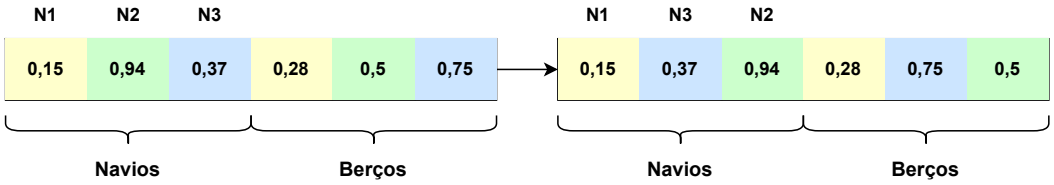


Figura 20 – Decodificação 1 - ordem de atracação.

A definição do berço é feita pegando o teto da multiplicação dos valores das chaves aleatórias da segunda metade do vetor pela quantidade de berços, sendo o valor resultante o berço no qual o respectivo navio será alocado. Seguindo o exemplo apresentado na Figura 20, a Figura 21 ilustra essa operação de definição dos berços.

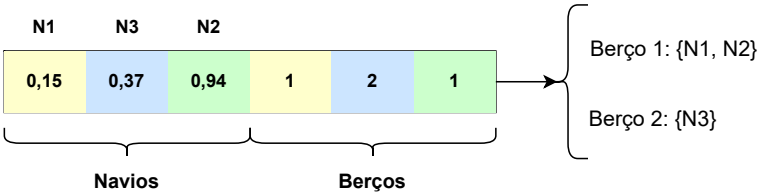


Figura 21 – Decodificação 1 - local de atracação.

Essa forma de decodificação que inclui a definição do berço através das chaves aleatórias e independentemente das que determinam a ordem de alocação dos navios

pode melhorar a variabilidade na geração de soluções, importante para a convergência, principalmente considerando que o BRKGA é um algoritmo genético elitista.

Com essa decodificação é garantido que todos os navios são alocados e que cada navio é atendido por um único berço. Além da ordem e local de alocação, é preciso definir os passos de tempos dessas alocações.

O cálculo do tempo de alocação é definido de acordo com a proposta de Mauri et al. (2008a), garantindo que o primeiro navio de cada berço só é alocado depois da sua chegada ao porto e quando o berço está livre para atendê-lo, garantindo o cumprimento das restrições de chegada do navio e abertura do berço e de não sobreposição no atendimento.

O tempo de alocação do navio i no berço k é definido levando em consideração o horário de chegada do navio (a_i), o horário de abertura do berço (s_k) e o término de atendimento do navio anterior ao i no berço k ($T_{jk} + t_{jk}$), dado por:

$$T_{ik} = \begin{cases} \max\{a_i, s_k\} & \text{se navio } i \text{ é o primeiro atendido pelo berço } k, \\ \max\{a_i, T_{jk} + t_{jk}\} & \text{se navio } j \text{ é atendido antes de } i; \end{cases}$$

Essa definição do horário de atendimento não garante, contudo, as restrições de fechamento dos berços e tempo máximo de permanência dos navio no porto. Essas restrições são tratadas, então, através de penalizações na função *fitness*.

Outra opção de decodificação pode ser não considerar as posições da primeira metade do vetor de chaves aleatórias associadas às posições da segunda metade, atrelando a ordenação da primeira com uma ordenação da segunda. A Figura 22 exemplifica essa situação.

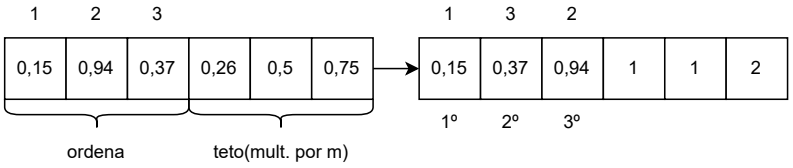


Figura 22 – Decodificação 2.

Aqui a troca na ordem dos elementos da primeira metade não carrega uma troca da ordem dos elementos da segunda metade. Dessa forma, no exemplo apresentado na Figura 22, teremos no berço 1 os navios 1 e 3 (nessa ordem) e, no berço 2, o navio 2.

4.3.2 Mutação

A mutação apresentada é a clássica do BRKGA, ou seja, a geração dos indivíduos de maneira aleatória, da mesma forma que a população inicial é gerada.

4.3.3 Cruzamento

O cruzamento no BRKGA prioriza as qualidades dos indivíduos da população elite para a geração dos herdeiros. Como os indivíduos definem tanto a ordem quanto o berço da alocação, essas duas características devem ser passadas juntas para os herdeiros. Assim, o cruzamento é feito considerando o navio e o respectivo berço, sendo a seleção principal focada no navio. A seleção do pai elite e do pai não elite é feita de maneira aleatória e o cruzamento utilizado é o uniformemente parametrizado (Michalewicz, 1996, Gonçalves and Resende, 2018).

A partir de um vetor de tamanho n (quantidade de navios) gerado aleatoriamente e com a probabilidade de herdar um gene do pai elite, determina-se de qual pai o índice da primeira metade do vetor e o respectivo valor de berço da segunda metade do vetor também segue, junto, para montar o filho. Como uma solução elite associa não só a ordem de alocação mas também o berço no qual esse navio é alocado, essas duas informações são herdadas em pares.

4.3.4 Função *fitness*

A função *fitness* busca, assim como as funções objetivo dos modelos PRVJT e PRVJT-O, minimizar a soma dos tempos de permanência dos navios no porto sendo definida e, além disso, foram adicionadas penalizações das restrições do tempo de permanência máximo dos navios no porto e fechamento dos berços, não garantidas na decodificação. Para o BRKGA do modelo PRVJT-O, é acrescentado na *fitness*, o custo por ociosidade dos berços.

Como o modelo PRVJT é para um problema de minimização e, em um algoritmo genético, queremos os indivíduos com maior valor *fitness* (mais adaptados ao meio), a função objetivo do modelo entra com sinal trocado. De maneira equivalente para o PRVJT-O.

As restrições referentes ao horário máximo de permanência dos navios no porto são dadas pela equação 4.28:

$$T_{i,k} + h_{i,k} \sum_{j \in N \cup d(k)} x_{i,j,k} \leqslant b_i.$$

Ajustando o conjunto de equações para obter uma restrição de não negatividade, temos:

$$b_i - T_{i,k} + h_{i,k} \sum_{j \in N \cup d(k)} x_{i,j,k} \geqslant 0. \tag{4.45}$$

Com isso, para que as restrições sejam satisfeitas, o lado esquerdo da equação deve ser positivo, o que favorece maior valor *fitness*. Além disso, quanto maior seu valor, mais "distante" da não factibilidade está a solução e mais adiantado na partida o navio estará.

Encontra-se, na literatura, a forma de lidar com essa infactibilidade em métodos (meta)heurísticos, a penalização apenas quando ocorre infactibilidade (Mauri et al., 2008b,

Mauri et al., 2008a, Mauri et al., 2016, de Oliveira et al., 2012, Lopes et al., 2011), ou seja, seria como penalizar na função *fitness* o termo:

$$\min \left\{ b_i - T_{i,k} + h_{i,k} \sum_{j \in N \cup d(k)} x_{i,j,k},\ 0 \right\}. \tag{4.46}$$

Contudo, considerar todo o caso e não só a infactibilidade, traz na *fitness* a possibilidade de considerar que adiantar a saída de navios pode ser benéfico. Considerando o mesmo peso para os atrasos e adiantamentos, o *fitness* pode beneficiar o adiantamento de vários navio em detrimento do atraso de outros. Por exemplo, pode compensar adiantar em 4 unidades de tempo a antecipação da saída de um navio mesmo atrasando a saída de outro em 1 unidade de tempo $(4 - 1 = 3)$, do que adiantar a saída do primeiro navio em apenas uma unidade de tempo e garantir o segundo sair do porto no seu horário máximo permitido $(1 + 0 = 1)$.

Pode-se considerar, também, diferentes pesos para quando o termo é positivo e quando é negativo, podendo penalizar mais ou menos o atraso em relação ao benefício de adiantar a saída de navios, inclusive combinar essas características dependendo do navio.

A decisão da melhor forma a se considerar pode depender das decisões de negócio do porto, se há algum tipo de ganho em buscar adiantar a saída de alguns navios mesmo que em detrimento do atraso de outros ou se o ideal é apenas evitar atrasos na saída dos navios.

Ideia equivalente pode ser aplicada na consideração da relaxação das restrições 4.30, podendo não apenas penalizar a infactibilidade mas também considerar o benefício em adiantar o fechamento do berço. Na função *fitness* foi considerada a penalização da factibilidade e o benefício da antecipação com um coeficiente p_2 também positiva.

Por fim, considerando o custo de adiantamento e atraso nas janelas de tempo e a minimização dos tempos dos navios no porto, a função *fitness* para o modelo PRVJT pode ser descrita por:

$$\begin{aligned} -\sum_{i \in N} \sum_{k \in M} \nu_i \left(T_{i,k} - a_i + h_{i,k} \sum_{j \in N \cup d(k)} x_{i,j,k} \right) \\ + p_1 \left(b_i - T_{i,k} + h_{i,k} \sum_{j \in N \cup d(k)} x_{i,j,k} \right) \\ + p_2 \left(e_k - T_{i,k} \right). \end{aligned} \tag{4.47}$$

Acrescentando o termo dos tempos de ociosidade dos berços na função objetivo, para o modelo PRVJT-O a função *fitness* pode ser descrita por:

$$
\begin{aligned}
&- \sum_{i \in N} \sum_{k \in M} \nu_i \left(T_{i,k} - a_i + h_{i,k} \sum_{j \in N \cup d(k)} x_{i,j,k} \right) \\
&\qquad - \sum_{k \in M} \sum_{i \in N \cup o(k)} \sum_{j \in N} y_{i,j,k} \\
&\quad + p_1 \left(b_i - T_{i,k} + h_{i,k} \sum_{j \in N \cup d(k)} x_{i,j,k} \right) \\
&\qquad + p_2 \left(e_k - T_{i,k} \right). \qquad (4.48)
\end{aligned}
$$

4.3.5 Implementação

Em um algoritmo BRKGA, o cálculo de decodificação e avaliação do *fitness* de cada indivíduo pode ser custoso computacionalmente, influenciando na qualidade e performance da busca. Algumas estratégias são usadas para lidar com essa situação, melhorando o desempenho e acelerando a convergência, como evoluir múltiplas populações independentemente, paralelizar operações, reiniciar população e utilizar buscas locais no processo de evolução (Gonçalves and Resende, 2011, Toso and Resende, 2015, Gonçalves and Resende, 2018). APIs generalizadas de BRKGA deixam por conta do usuário a implementação da decodificação, uma vez que é particular de cada problema, utilizando estratégias que possam melhorar o desempenho do algoritmo independentemente da decodificação utilizada e, em geral, buscam em linguagens mais complexas, como C++, a possibilidade de *multithread* e paralelização.

Utilizando o *Profiler* [1] do *Python* para medir desempenho do algoritmo, que identifica quais são os passos que mais consumem tempo computacional, foi possível identificar que a parte de cálculo dos horários de alocação dos navios e liberação dos berços é a mais custosa em tempo computacional. Essa parte da decodificação é complexa computacionalmente, apesar de não ser exponencial com o tamanho da instância. Uma forma de melhorar o desempenho e a convergência do algoritmo foi utilizar a biblioteca *numpy* do *Python* para vetorizar as operações dos cálculos dos horários de atendimento e espera dos navios, das restrições relaxadas e da função *fitness* e, para o modelo PRVJT-O, dos tempos de ociosidade dos berços.

A ideia é calcular de maneira matricial e acumulativa os valores de tempo de ocupação dos berços e, para isso, foi preciso redimensionar os vetores associados às janelas de tempo e ao tempo de atendimento dos navios nos berços, estendendo uma dimensão para cada indivíduo da população. As variáveis de horário de alocação dos navios também

[1] *Python Profilers*

foram criadas considerando, além de $n \times m$, a dimensão do tamanho da população, sendo calculado de maneira matricial e cumulativa.

Assim, para cada berço e cada navio, são calculados os tempos de todos os indivíduos da população. Os valores relacionados ao horário em que o berço estará livre para atender o próximo navio e o tempo de espera dos navios são calculados separadamente, assim como a parte relacionada à função objetivo do modelo e as relaxações, para podermos identificar se a solução gerada satisfaz a restrição de fim de janela de tempo. A geração das populações inicial e mutante e a ordenação das chaves aleatórias também é feita matricialmente. Essa forma matricial de cálculo não permite o cômputo da decodificação de cada indivíduo da população separadamente.

4.4 Clusterização

Para definir a similaridade ou a distância entre dois elementos poderíamos considerar, por exemplo, o valor da função objetivo das soluções ou a distância Euclidiana entre os vetores de chaves aleatórias (abordagem clássica). Mas, como queremos identificar o quanto uma alocação é parecida ou não com outra, as distâncias entre duas soluções foram definidas nos vetores decodificados por 1 menos a razão entre a quantidade de alocações iguais pelo total de alocações. Essa razão representa a porcentagem de alocações diferentes e, ao subtrair essa razão de 1, identificamos a porcentagem de alocações iguais. Assim, as soluções iguais possuem distância 0 e as completamente diferentes, 1.

Consideremos um exemplo com $N = 3$ navios e $M = 2$ berços, gerando dois vetores de chaves aleatórias e decodificando-os, supomos obter as seguintes soluções descritas na Figura 23, onde é possível observar que há apenas o navio alocado de maneira igual na duas soluções dentre as três alocações para os navios. Assim, o valor da distância é definida por:

$$dist_{sol_1,\, sol_2} = 1 - \frac{1}{3} = \frac{2}{3}.$$

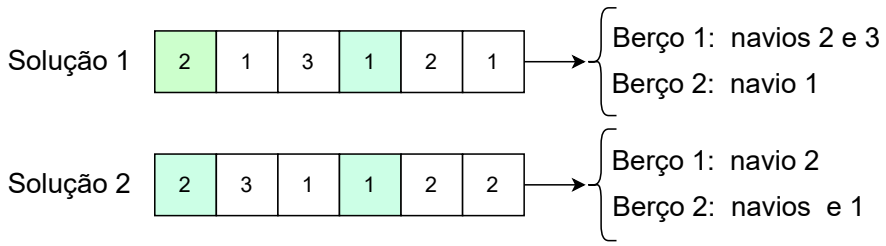


Figura 23 – Exemplo para cálculo de distâncias da clusterização.

Ou seja, a distância representa a dissimilaridade entre os indivíduos. Por exemplo, se a distância entre duas soluções for de $0,4$, temos que 40% dos navios foram

alocados em posições e berços diferentes e 60% possuem a mesma alocação em ordem e local.

O modelo de *clusterização* selecionado para os testes foi a clusterização aglomerativa, que não exige que a quantidade de *clusters* seja pré determinada e possibilita escolher o nível de dissimilaridade aceita para que dois indivíduos estejam em um mesmo *cluster*. Por exemplo, se selecionarmos o valor de $0,6$, significa que as distâncias ou a ligação entre dois indivíduos (ou clusters) não é maior que $0,6$.

Além do nível de similaridade, outro parâmetro a ser escolhido é o *linkage*, ou seja, a medida de distância entre dois *clusters*. Como foi utilizado o método *AgglomerativeClustering* da biblioteca *sklearn* do *Python*, temos disponíveis para a escolha, considerando que trabalhamos com uma matriz de distância pré calculada e não a Euclidiana, as ligações simples, completa ou média. [2]

[2] AgglomerativeClustering

5 Experimentos computacionais

Neste capítulo apresentamos os testes realizados com o modelo *Dinamic Berth Allocation Problem* (DBAP) proposto por Imai et al. (2001), com instâncias definidas manualmente a fim de validar o modelo proposto.

Além disso, apresentamos os testes computacionais envolvendo o modelo baseado no Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) e a adaptação proposta que inclui a ociosidade dos berços. Para esses modelos fizemos alguns testes iniciais com instâncias geradas e exploramos mais com outras da literatura, testando com o BRKGA com diferentes estratégias, incluindo a proposta de clusterização, e com o CPLEX.

As implementações e testes computacionais foram feitos em linguagem *Python*, versão 3.8.8. O CPLEX utilizado foi da versão 20.1, com o DOCPLEX versão 2.20.204, para a implementação em *Python*. Para o teste com o modelo DBAP e os testes iniciais com as instâncias geradas para o modelo PRVJT sem ociosidade de berços pelo BRKGA e CPLEX, foram feitos numa máquina Intel(R) Core(TM) i5-9300HF, CPU 2.40 GHz, RAM instalada: 8GB e Armazenamento: 256GB SSD.

Esses testes iniciais para o modelo PRVJT sem e com ociosidade de berços com as instâncias geradas foram feitos mais para validar os modelos. Todos os outros testes envolvendo as instâncias da literatura, foram desenvolvidos em uma máquina Intel(R) Xeon(R), com 40 CPUs, CPU E5-2650 2.30GHz, com memória de 125GB.

Os testes com instâncias da literatura consideraram um subconjunto das geradas e disponibilizadas por Lalla-Ruiz et al. (2012), que podem ser encontradas neste endereço. O subconjunto de instâncias testadas, com suas respectivas nomenclaturas, está descrito na Tabela 7.

Os testes computacionais dos problemas sem e com ociosidade de berços implementados como um Problema de Roteamento de Veículos com Janela de Tempo foram realizados para diferentes abordagens com o BRKGA, ilustrados na Figura 24, e para diferentes abordagens com o CPLEX, que envolvem limite de tempo de processamento e utilização do BRKGA como solução inicial, ilustrados na Figura 25.

5.1 Modelo *Dinamic Berth Allocation Problem* para o PAB

A implementação do modelo DBAP sugerido por Imai et al. (2001), para o PAB discreto e dinâmico, foi testado para pequenas instâncias para avaliar a qualidade da solução fornecida, devido ao fato do modelo não avaliar corretamente o valor da função

Instância	Navios	Berços
f30x3-1	30	3
f30x3-2	30	3
f30x3-3	30	3
f30x3-4	30	3
f30x3-5	30	3
f40x5-1	40	5
f40x5-2	40	5
f40x5-3	40	5
f40x5-4	40	5
f40x5-5	40	5
f55x5-1	55	5
f55x5-2	55	5
f55x5-3	55	5
f55x5-4	55	5
f55x5-5	55	5
f60x5-1	60	5
f60x5-2	60	5
f60x5-3	60	5
f60x5-4	60	5
f60x5-5	60	5

Tabela 7 – Subconjunto de instâncias da literatura geradas por Lalla-Ruiz et al. (2012).

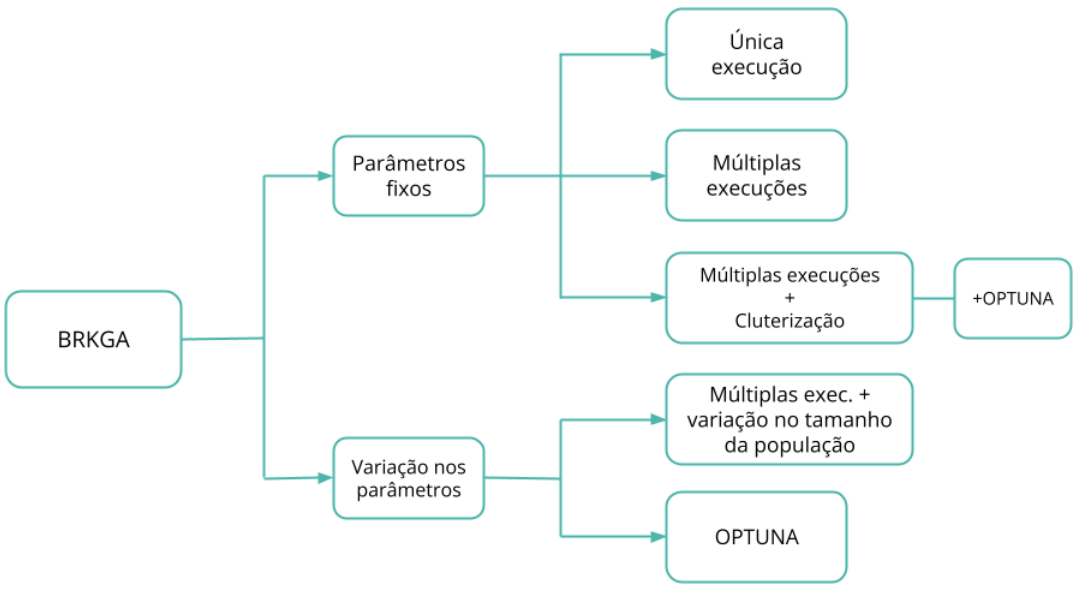


Figura 24 – Organização dos testes computacionais com o BRKGA.

objetivo.

Uma instância testada foi a seguinte:

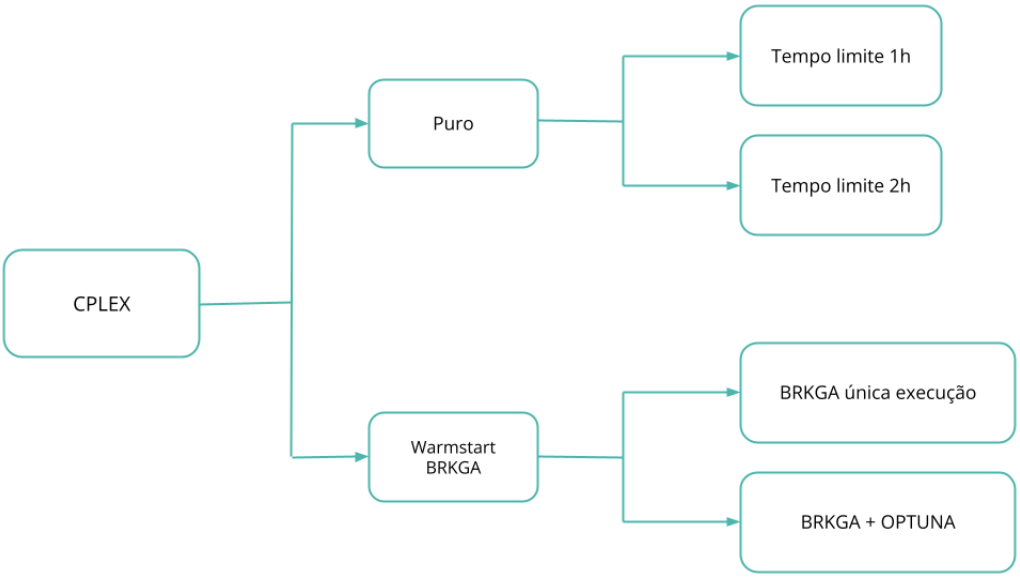


Figura 25 – Organização dos testes computacionais com o CPLEX.

- Berços: $m = 2$
- Navios: $n = 3$
- Abertura berços: $S = [0, 2]$
- Chegada dos navios: $A = [1, 0, 3]$
- Tempo de atendimento navio $\times$ berço (h_{ik}):

$$H = \begin{bmatrix} 2 & 4 \\ 3 & 5 \\ 13 & 8 \end{bmatrix}$$

O resultado obtido pela implementação é ilustrado na Figura 26, considerando a interpretação da solução de que x_{ipk} está associado ao navio i alocado na posição p no berço k.

A soma dos tempos de permanência dos navios no porto, com os tempos de ociosidade dos berços resulta em 22 unidades de tempo:

- Atendimento: $3 + 2 + 13 = 18$
- Espera navios: $2 + 2 = 4$
- Soma dos tempos de permanência dos navios no porto: 22

Os valores não nulos das variáveis e os respectivos valores na função objetivo do modelo são:

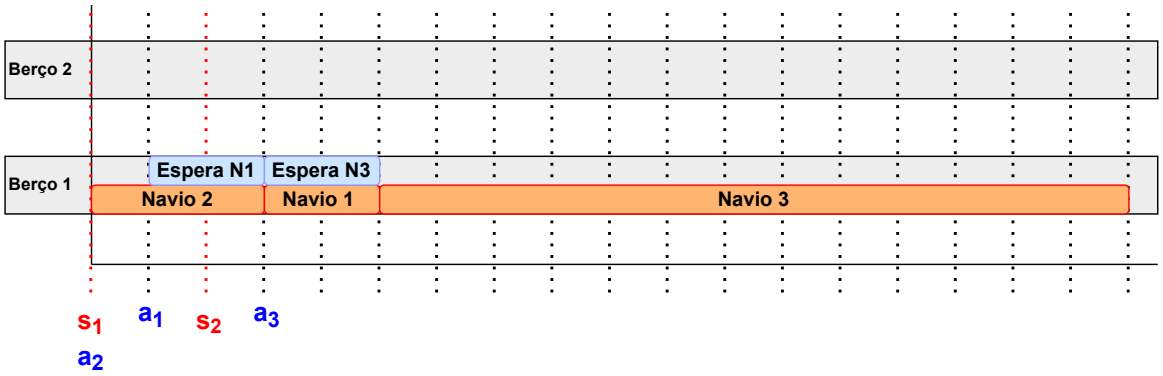


Figura 26 – Solução de uma instância do DBAP implementado.

- $x_{2,1,1} = 1$: $(3 - 1 + 1) * 3 + 0 - 0 = 9$
- $x_{1,2,1} = 1$: $(3 - 2 + 1) * 2 + 0 - 1 = 3$
- $x_{3,3,1} = 1$: $(3 - 3 + 1) * 13 + 0 - 3 = 10$
- Total da função objetivo: $9 + 3 + 10 = 22$

Contudo, se fizermos uma pequena troca na alocação e alocar o terceiro navio no segundo berço, teremos a solução factível ilustrada na Figura 27:

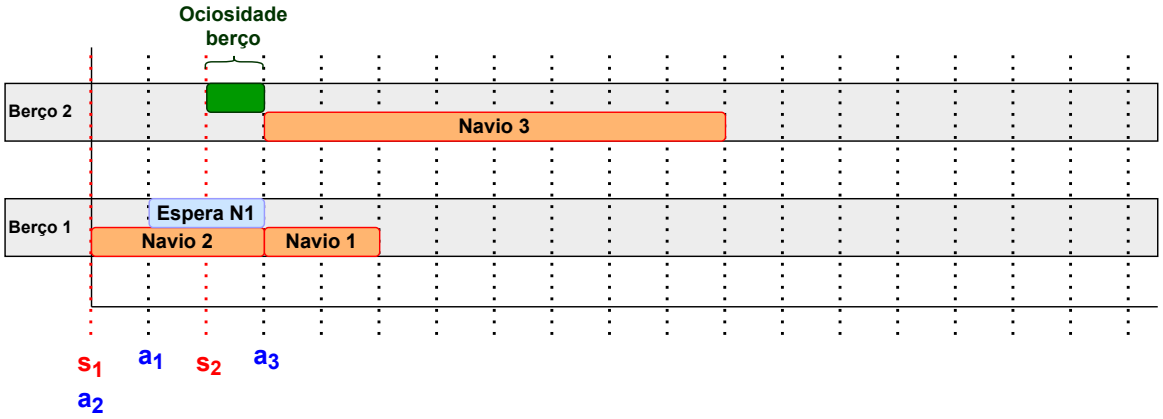


Figura 27 – Solução factível da instância testada.

A soma dos tempos de permanência dos navios no porto, com os tempos de ociosidade dos berços resulta em 16 unidades de tempo:

- Atendimento: $3 + 2 + 8 = 13$
- Espera navios: 2
- Ociosidade berços: 1

- Soma dos tempos de permanência dos navios no porto e ociosidade dos berços: 16

Podemos notar que essa solução (factível) apresenta um menor tempo total de permanência dos navios no porto. Vamos analisar essa solução no modelo:

- $x_{2,1,1} = 1$: $(3 - 1 + 1) * 3 + 0 - 0 = 9$
- $x_{1,2,1} = 1$: $(3 - 2 + 1) * 2 + 0 - 1 = 3$
- $x_{3,1,2} = 1$: $(3 - 1 + 1) * 8 + 2 - 3 = 23$
- $y_{3,1,2} = 1$: $(3 - 1 + 1) * 1 = 3$
- Total da função objetivo: $9 + 3 + 23 + 3 = 35$

Percebe-se, nesse caso, que o valor da função objetivo dessa solução factível, que resulta em um menor tempo total de permanência dos navios no porto, é maior que o valor objetivo da solução indicada como solução do problema.

Como a função objetivo traz a soma acumulada do tempo de atendimento de um navio para a própria posição e as posições seguintes, quanto mais avançada a posição de atendimento do navio, menor o valor acumulado. Assim, a alocação de todos os navios em um único berço resulta em um valor de função objetivo menor.

Trazendo a interpretação da solução apresentada no *corrigendum* Imai et al. (2005a), teríamos a mesma ordem de atendimento apresentada na Figura 26 mantendo, assim, uma solução que é factível mas não a melhor possível para o problema, como uma solução ótima.

Considerando a mesma instância testada anteriormente, acrescentando o conjunto de restrições associado à janela de tempo dos berços, 4.8, a solução apresentada para a alocação dos navios foi a mesma, com exceção de uma variável de ociosidade do berço 2, mas sem alteração na função objetivo.

O tempo de ociosidade do berço 2 associado à alocação do primeiro navio foi definida com um valor positivo, $y_{122} = 22$, apesar de não ter ocorrido nenhuma alocação nesse berço. Esse valor de ociosidade para o berço não altera o valor da função objetivo sendo, assim, uma solução alternativa. Como o termo da função objetivo que minimiza os tempos de ociosidade dos berços está definida para os navios que chegam depois que o berço abre e o navio 1 chega antes, essa variável pode assumir valor positivo sem alterar a função objetivo.

Para ajustar essa situação, pode ser considerado um conjunto de restrições do tipo M-grande que permita que o valor das variáveis de ociosidade do berço sejam positivas somente se houver pelo menos um navio alocado no respectivo berço.

Mantendo o modelo proposto em Imai et al. (2001), uma alteração na função objetivo teria que considerar o tempo de atendimento dos navios atendidos nas posições anteriores e não posteriores. Mas uma alteração dessa forma tornaria a função objetivo não linear. Como já mencionado, Hansen and Oğuz (2003) propuseram uma correção no modelo, considerando a ordem reversa de alocação nas variáveis.

5.2 PAB modelado como PRVJT sem e com ociosidade de berços: resolução pelo CPLEX

Os experimentos computacionais desta seção foram feitos resolvendo o modelo PRVJT sem e com ociosidade de berços pelo CPLEX, com configurações padrão, para analisar soluções e tempo computacional, com as instâncias da literatura descritas em 7. Devido à dificuldade de resolver o modelo por conta do tempo computacional, foi necessário limitar o tempo do CPLEX. Assim, os experimentos foram feitos para os tempos limites de uma e duas horas, sem considerar o tempo de pré-processamento e analisamos as diferenças nas soluções encontradas.

As Tabelas 8 e 10 fornecem as diferenças de valor objetivo para cada limite de tempo de processamento para o PRVJT sem e com ociosidade de berços, respectivamente, e a Tabela 9 fornece os *status* das soluções obtidas para o modelo com ociosidade de berços, na qual podemos observar que o modelo não conseguiu nem definir o *status* para algumas soluções dentro do tempo limite definido. Para o modelo sem ociosidade de berços, todas as soluções encontradas foram factíveis.

O *gap* da diferença dos valores objetivos é definido da seguinte forma:

$$gap_{obj} = \frac{(V.Obj._{1h} - V.Obj._{2h}) \times 100}{V.Obj._{1h}}, \tag{5.1}$$

onde V.Obj significa valor objetivo.

Assim, esse *gap* representa o quanto, em porcentagem, o resultado para o tempo limite de 2 horas foi menor (melhor) que o valor objetivo para o limite de 1 hora. Caso o valor resultante seja negativo, então o resultado para o tempo limite de duas horas foi maior (pior) que o de uma hora.

Dos resultados apresentados na Tabela 8 podemos observar que, para o modelo sem ociosidade de berços, o valor objetivo para duas horas tem um máximo de melhora em torno de 5% e, em alguns poucos casos, resulta em valor objetivo maior que a execução com uma hora, e com o dobro do tempo de processamento.

Para o modelo com ociosidade de berços, na Tabela 9, o *status* UNKNOWN do CPLEX significa que não foi processado tempo suficiente para poder provar qualquer

Instância	$V.Obj._{1h}$	$V.Obj._{2h}$	gap_{obj}
f30x3-01	1771	1767	0,226
f30x3-02	2097	2116	-0,906
f30x3-03	2213	2201	0,542
f30x3-04	1572	1556	1,018
f30x3-05	2143	2139	0,187
f40x5-01	2377	2343	1,430
f40x5-02	2998	2864	4,470
f40x5-03	2921	2947	-0,890
f40x5-04	2240	2159	3,616
f40x5-05	3025	2893	4,364
f55x5-01	5027	4896	2,606
f55x5-02	6221	6003	3,504
f55x5-03	6019	5826	3,207
f55x5-04	4759	4768	-0,189
f55x5-05	5936	5790	2,460
f60x5-01	6379	6212	2,618
f60x5-02	7872	7508	4,624
f60x5-03	7462	7241	2,962
f60x5-04	5564	5382	3,271
f60x5-05	7562	7543	0,251

Tabela 8 – Valores objetivo pelo CPLEX para diferentes limites de tempo, modelo sem ociosidade de berços.

coisa sobre o modelo e que um motivo comum pode ser um tempo limite insuficiente. Já o *status* FEASIBLE_SOLUTION significa que o CPLEX encontrou uma solução factível, mas não chegou na solução ótima ou não conseguiu provar a otimalidade por falta de tempo de processamento.

Para a instância f60x5-02, é possível observar que, para o tempo limite de uma hora, o CPLEX não conseguiu provar nada sobre o modelo mas encontrou uma solução factível para duas horas. Para as instâncias f55x5-05 e f60x5-05 nenhum dos dois tempos limite definidos foi suficiente para o CPLEX conseguir encontrar uma solução factível.

Para o modelo com ociosidade de berços, o aumento no tempo limite de uma para duas horas resultou em um maior impacto no valor objetivo encontrado do que para o modelo sem ociosidade, chegando a melhorar em até, aproximadamente, 21%. A diferença se apresentou mais relevante para as instâncias maiores, com 55 e 60 navios e 5 berços, incluindo a obtenção de uma solução factível para o tempo de duas horas em uma instância que, com uma hora, o CPLEX não pôde concluir nada sobre o modelo.

Resolvendo o mesmo conjunto de instâncias pelo BRKGA proposto, podemos comparar o desempenho com os modelos resolvidos pelo CPLEX. Como, para o modelo sem ociosidade, a melhora no tempo limite de duas horas é pequena diante do aumento no tempo computacional, a comparação será com o modelo resolvido com uma hora de

Instância	*Status* CPLEX 1h	*Status* CPLEX 2h
f30x3-01	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f30x3-02	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f30x3-03	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f30x3-04	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f30x3-05	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f40x5-01	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f40x5-02	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f40x5-03	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f40x5-04	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f40x5-05	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f55x5-01	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f55x5-02	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f55x5-03	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f55x5-04	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f55x5-05	UNKNOWN	UNKNOWN
f60x5-01	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f60x5-02	UNKNOWN	FEASIBLE_SOLUTION
f60x5-03	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f60x5-04	FEASIBLE_SOLUTION	FEASIBLE_SOLUTION
f60x5-05	UNKNOWN	UNKNOWN

Tabela 9 – *Status* do CPLEX para o modelo PRVJT com ociosidade de berços.

tempo limite e os resultados podem ser vistos na Tabela 14. Já o modelo com ociosidade foi comparado com o tempo limite de duas horas e os resultados podem ser vistos na Tabela 15.

5.3 PAB modelado como PRVJT sem e com ociosidade de berços: comparando resolução pelo CPLEX com resolução pelo BRKGA

5.3.1 Instâncias geradas

As instâncias foram geradas de maneira aleatória buscando garantir factibilidade. Para $m = 5$, $m = 8$ foram geradas 10 instâncias com $n = 10$, $n = 20$, $n = 40$, $n = 60$ navios, para $m = 13$ berços, 10 instâncias para $n = 20$, $n = 40$, $n = 60$, $n = 80$ e $n = 90$ navios e, para $m = 20$ berços, foram geradas 10 instâncias com $n = 90$ navios. Para cada combinação (m, n) foram geradas 10 instâncias.

Essas instâncias possuem os horários de chegada dos navios mais distribuídos ao longo do horizonte de planejamento, resultando em um "menor congestionamento" no porto, o que torna o problema mais fácil de ser resolvido pelo CPLEX. Assim, é possível comparar os dois métodos para instâncias nas quais o CPLEX tem melhor desempenho. Os testes com as instâncias geradas foram feitos apenas com o modelo sem ociosidade de

Instância	$V.Obj._{1h}$	$V.Obj._{2h}$	GAP_{obj}
f30x3-01	1783	1801	-1,010
f30x3-02	2133	2117	0,750
f30x3-03	2192	2188	0,182
f30x3-04	1551	1561	-0,645
f30x3-05	2174	2134	1,840
f40x5-01	2425	2403	0,907
f40x5-02	2955	2893	2,098
f40x5-03	3056	2960	3,141
f40x5-04	2241	2225	0,714
f40x5-05	3054	2966	2,881
f55x5-01	5401	4976	7,869
f55x5-02	7098	6320	10,961
f55x5-03	6487	5958	8,155
f55x5-04	5520	5099	7,627
f55x5-05	–	–	–
f60x5-01	8633	6834	20,839
f60x5-02	–	7880	–
f60x5-03	8326	7913	4,960
f60x5-04	6770	6224	8,065
f60x5-05	–	–	–

Tabela 10 – Valores objetivo pelo CPLEX para diferentes limites de tempo, modelo com ociosidade de berços.

berços.

A partir dos testes com as instâncias geradas, comparando uma implementação mais simples do BRKGA com a vetorizada em relação ao CPLEX. A implementação mais simples não resultou em melhores tempos computacionais ou resultados objetivos iguais ao CPLEX, uma vez que este alcançou a solução ótima para essas instâncias. Estes testes serviram para validar o BRKGA proposto, assim como sua implementação, com relação à resolução pelo CPLEX. Os resultados apresentados nesta seção são para a implementação vetorizada.

5.3.1.0.1 Parâmetros BRKGA

Os parâmetros do BRKGA para o BAP foram escolhidos de maneira empírica dentro dos valores sugeridos pela literatura, encontrados na Tabela 6, e definidos por:

1. tamanho da população: $p = 10 \times n$;
2. tamanho da população elite: $p_e = 0,2 \times p$;

3. tamanho da população mutante: $p_m = 0,3 \times p$;
4. probabilidade de herdar gene do pai elite: $\rho_e = 0,6$.

Com relação aos critérios de parada, foram considerados quantidade máxima de gerações e quantidade de gerações consecutivas com variação do melhor *fitness* menor que uma tolerância definida. Os valores dos parâmetros associados aos critérios de parada são:

1. máximo de gerações: 1000;
2. quantidade máxima de gerações sem variação significativa, de acordo com uma precisão, no *fitness* do melhor indivíduo: 15 gerações;
3. precisão: 10^{-04}.

5.3.1.0.2 Resultados computacionais

Nas Tabelas 11 e 12 podemos observar as médias de gap da função objetivo e do tempo computacional entre as 10 instâncias testadas para cada dimensão.

O *gap* da função objetivo foi definida da seguinte forma:

$$Gap_{obj} = \frac{|Obj_{BRKGA} - Obj_{CPLEX}|}{Obj_{CPLEX}} \tag{5.2}$$

Ou seja, o *gap* dos valores objetivos fornece o quão distante, em porcentagem, o valor objetivo do BRKGA está do PRVJT. Quanto mais próximo de zero o *gap*, mais próximo o BRKGA chegou do valor ótimo da solução resolvida de forma exata pelo modelo PRVJT.

O *gap* do tempo computacional foi definido como a porcentagem que o tempo de execução do BRKGA corresponde do tempo de execução para a resolução do PRVJT. Ou seja, para porcentagens menores que 100% do *gap*, significa que o BRKGA foi mais rápido.

$$Gap_{tempo} = \frac{Tempo_{BRKGA}}{Tempo_{CPLEX}} \tag{5.3}$$

Pela Tabela 11 podemos reparar que a maior diferença, em média, entre o valor objetivo encontrado pelo BRKGA com o PRVJT é de 30%. O maior desvio padrão de 22% pode ser explicado por alguns *outliers* nos resultados, que podem ser vistos nos gráficos *boxplot* abaixo. Com relação ao tempo computacional, o BRKGA apresentou, em média,

Berços	Navios	Média *gap* objetivo	Desvio padrão *gap* objetivo
5	10	0,02	0,03
5	20	0,09	0,07
5	40	0,28	0,09
5	60	0,48	0,12
8	10	0,09	0,11
8	20	0,16	0,08
8	40	0,22	0,10
8	60	0,35	0,12
13	20	0,13	0,07
13	40	0,23	0,09
13	60	0,24	0,06
13	80	0,38	0,22
13	90	0,34	0,05
20	90	0,32	0,07

Tabela 11 – Comparação dos valores objetivos encontrados entre PRVJT e BRKGA.

Berços	Navios	Média *gap* tempo execução	Desvio padrão *gap* tempo execução
5	10	0,20	0,05
5	20	0,36	0,05
5	40	0,63	0,08
5	60	0,66	0,39
8	10	0,15	0,06
8	20	0,26	0,06
8	40	0,51	0,11
8	60	0,77	0,15
13	20	0,17	0,02
13	40	0,40	0,05
13	60	0,58	0,07
13	80	0,72	0,15
13	90	0,77	0,13
20	90	0,66	0,19

Tabela 12 – Comparação tempo de execução entre PRVJT e BRKGA.

tempo menor que o PRVJT em todas as instância chegando, no máximo, em torno de 77% do tempo computacional do PRVJT.

As Figuras 28 e 29 mostram gráficos *boxplot* da distribuição dos *gaps*. Nos gráficos é possível perceber melhor a distribuição dos gaps, incluindo os *outliers*.

Podemos notar, a partir das tabelas e dos gráficos, que o BRKGA teve um desempenho computacional melhor que a resolução pelo modelo exato mas, como é de se esperar, o BRKGA não traz soluções ótimas para o problema. Outra não garantia do BRKGA proposto é a factibilidade com relação ao tempo de permanência máximo dos navios no porto. Mas, considerando instâncias infactíveis pode ser mais vantajoso, para o negócio, que o navio saia atrasado do que não seja atendido.

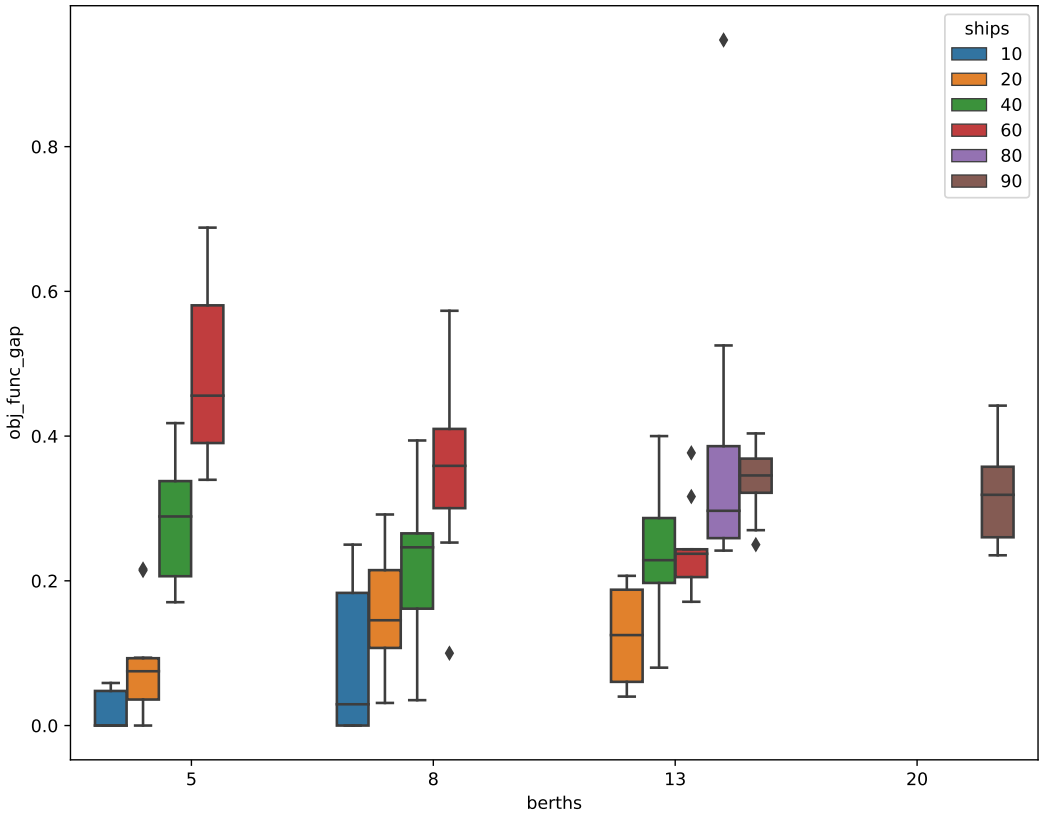


Figura 28 – Distribuição dos gaps dos valores objetivos do BRKGA com relação ao PRVJT.

Para uma quantidade fixa de berços, aumentar a quantidade de navios piora o gap entre os valores objetivos e os tempos computacionais. Contudo, aumentar a quantidade de berços, para uma quantidade fixa de navios, não muda muito os valores do gap, o que pode significar que a variação na quantidade de navios é mais relevante do que a variação na quantidade de berços.

5.3.2 Instâncias da literatura

Os testes computacionais feitos nesta seção consideram executar o BRKGA uma única vez, com parâmetros fixos para cada instância e executar o CPLEX para os tempos limites de uma e duas horas, de forma separada. Os resultados computacionais estão apresentados na Tabela 14 e os parâmetros utilizados para o BRKGA estão descritos na Tabela 13.

Para o BRKGA, os critérios de parada considerados são os mesmo que nos testes para as instâncias geradas, ou seja, quantidade máxima de gerações (1000), máximo de gerações consecutivas com variação do melhor *fitness* menor que uma tolerância, definida

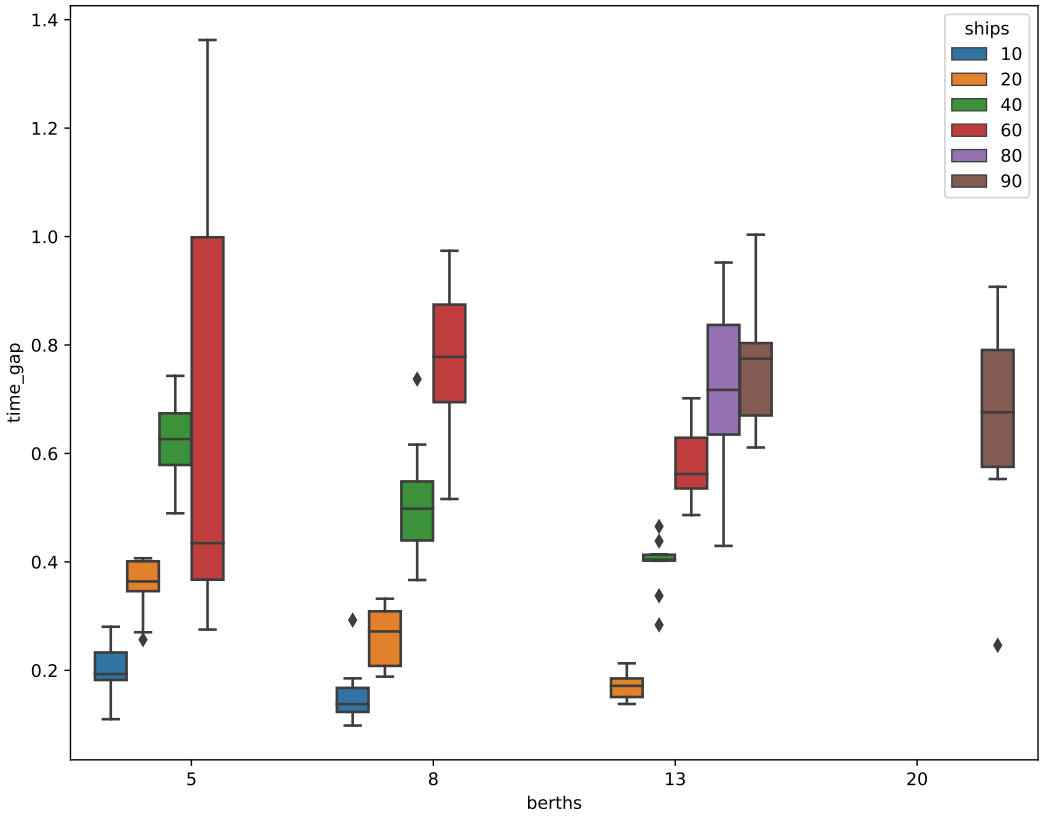


Figura 29 – Distribuição dos *gaps* dos tempos computacionais do BRKGA com relação ao PRVJT.

como 15 gerações e com tolerância de 10^{-04}.

Tamanho da população:	$p = 30 \times n$	n: quantidade de navios
Tamanho população elite:	$p_e = 0,2 \times p$	p: tamanho da população
Tamanho da população mutante:	$p_m = 0,3 \times p$	p: tamanho da população
Probabilidade de herdar gene elite	$\rho_e = 0,7$	

Tabela 13 – Parâmetros BRKGA

Os valores de GAP_{obj} e $GAP_{t.p}$ foram calculados de acordo com as equações 5.4 e 5.5, respectivamente.

$$GAP_{obj} = \frac{(Obj_{CPLEX} - Obj_{BRKGA}) \times 100}{Obj_{CPLEX}}, \tag{5.4}$$

$$GAP_{T.p} = \frac{T.p._{BRKGA} \times 100}{T.p._{CPLEX}}, \tag{5.5}$$

Instância	Obj_{BRKGA}	$Tp._{BRKGA}$(s)	Obj_{CPLEX}	$Tp._{CPLEX}$(s)	GAP_{obj}	$GAP_{Tp.}$
f30x3-01	1779	1,616	1771	3626,319	-0,452	0,045
f30x3-02	2132	1,949	2097	3619,708	-1,669	0,054
f30x3-03	2222	2,007	2213	3613,717	-0,407	0,056
f30x3-04	1615	0,864	1572	3619,617	-2,735	0,024
f30x3-05	2124	1,245	2143	3633,504	0,887	0,034
f40x5-01	2384	9,923	2377	3644,070	-0,294	0,272
f40x5-02	2949	2,769	2998	3646,303	1,634	0,076
f40x5-03	2935	8,292	2921	3607,489	-0,479	0,230
f40x5-04	2074	9,333	2240	3667,567	7,411	0,254
f40x5-05	2938	10,919	3025	3642,009	2,876	0,300
f55x5-01	4963	8,288	5027	3681,740	1,273	0,225
f55x5-02	5654	19,815	6221	3652,489	9,114	0,542
f55x5-03	5627	11,613	6019	3655,445	6,513	0,318
f55x5-04	4502	10,262	4759	3625,742	5,400	0,283
f55x5-05	5536	17,231	5936	3621,517	6,739	0,476
f60x5-01	5919	16,108	6379	3648,075	7,211	0,442
f60x5-02	7002	23,820	7872	3657,354	11,052	0,651
f60x5-03	7004	9,200	7462	3663,004	6,138	0,251
f60x5-04	5492	5,477	5564	3626,034	1,294	0,151
f60x5-05	7023	11,150	7562	3624,344	7,128	0,308

Tabela 14 – BRKGA *vs* CPLEX com limite de uma hora, modelo sem ociosidade.

Instância	Obj_{BRKGA}	$Tp._{BRKGA}$(s)	Obj_{CPLEX}	$Tp._{CPLEX}$(s)	GAP_{obj}	$GAP_{Tp.}$
f30x3-01	1770	1,496	1801	7260,640	1,721	0,021
f30x3-02	2105	1,697	2117	7335,913	0,567	0,023
f30x3-03	2209	1,746	2188	7263,071	-0,960	0,024
f30x3-04	1578	1,523	1561	7286,666	-1,089	0,021
f30x3-05	2140	2,129	2134	7240,938	-0,281	0,029
f40x5-01	2412	5,502	2403	7289,672	-0,375	0,075
f40x5-02	2871	4,970	2893	7238,033	0,760	0,069
f40x5-03	2939	4,617	2960	7250,256	0,709	0,064
f40x5-04	2151	9,073	2225	7309,773	3,326	0,124
f40x5-05	2928	7,373	2966	7337,379	1,281	0,100
f55x5-01	4809	11,207	4976	7329,748	3,356	0,153
f55x5-02	5679	17,123	6320	7313,038	10,142	0,234
f55x5-03	5704	6,182	5958	7318,133	4,263	0,084
f55x5-04	4565	7,470	5099	7323,171	10,473	0,102
f55x5-05	5536	19,601	–	7330,794	–	0,267
f60x5-01	5952	11,298	6834	7263,878	12,906	0,156
f60x5-02	7086	15,285	7880	7353,438	10,076	0,208
f60x5-03	7082	6,886	7913	7319,473	10,502	0,094
f60x5-04	5299	11,759	6224	7221,468	14,862	0,163
f60x5-05	7213	10,958	–	7236,295	–	0,151

Tabela 15 – BRKGA *vs* CPLEX com limite de duas horas, modelo com ociosidade.

onde, Obj_{CPLEX} e Obj_{BRKGA} são os valores objetivos encontrados pelo CPLEX e pelo BRKGA proposto, respectivamente, e $T.p._{CPLEX}$ e $T.p._{BRKGA}$ são os tempos de processamento de cada método, respectivamente.

Da Tabela 14, para o modelo sem ociosidade de berços, podemos notar que os valores objetivo encontrados pelo BRKGA são melhores para as instâncias maiores, de 40, 55 e 60 navios, com uma melhora máxima de aproximadamente 11%. Para as instâncias menores, o CPLEX resultou em geral em melhores soluções, com uma melhora de, no máximo, $2,8\%$. Além de boas soluções com relação ao CPLEX, o BRKGA levou menos de 1% do tempo que o CPLEX levou para obter esses valores e todas as soluções encontradas pelo BRKGA foram factíveis, mesmo a decodificação não garantindo todas as restrições e as de permanência máxima dos navios no porto e fechamento dos berços sendo penalizadas na função *fitness*.

Para o modelo com ociosidade de berços, podemos notar pela Tabela 15, que o BRKGA teve um desempenho ainda melhor que o CPLEX do que para o modelo sem ociosidade de berços. O que demonstra que o aumento da dimensão do problema é mais impactante na resolução pelo CPLEX do que pelo BRKGA.

5.4 Modelos PRVJT sem e com ociosidade de berços: BRKGA

Depois de uma comparação inicial do desempenho entre o BRKGA implementado de forma matricial com o CPLEX, validando o BRKGA proposto, alguns testes foram realizados para estudar o comportamento do BRKGA considerando os modelos baseados em PRVJT sem e com ociosidade de berços. As análises envolvem variação nos parâmetros do modelo, múltiplas execuções do método para cada instância e avaliando as melhores e piores soluções desse conjunto de execuções e utilização da ferramenta OPTUNA para definição dos parâmetros do BRKGA, incluindo os que definem os critérios de parada.

5.4.1 Variação no tamanho da população

Foram feitos testes variando o tamanho da população para um subconjunto das instâncias apresentadas em 7, descrito na Tabela 16.

Para cada instância, os tamanhos de população testados foram: $10 \times n$, $20 \times n$, $30 \times n$, $40 \times n$, $50 \times n$, com n sendo a quantidade de navios, ou seja, os multiplicadores que definem o tamanho da população testados foram 10, 20, 30, 40, 50 e, para cada par (instância, tamanho da população), o BRKGA foi rodado 10 vezes.

Os parâmetros para o BRKGA usados nestes testes foram os mesmos definidos na Tabela 13. Os critérios de parada considerados foram:

1. máximo de gerações: 1000;

Instância	Navios	Berços
f30x3-1	30	3
f30x3-2	30	3
f30x3-3	30	3
f30x3-4	30	3
f30x3-5	30	3
f40x5-1	40	5
f40x5-2	40	5
f40x5-3	40	5
f40x5-4	40	5
f40x5-5	40	5

Tabela 16 – Instâncias da literatura geradas por Lalla-Ruiz et al. (2012).

2. quantidade máxima de gerações sem variação significativa, de acordo com uma precisão, no *fitness* do melhor indivíduo: 15 gerações;

3. precisão: 10^{-04}.

5.4.1.0.1 Modelo sem ociosidade de berços

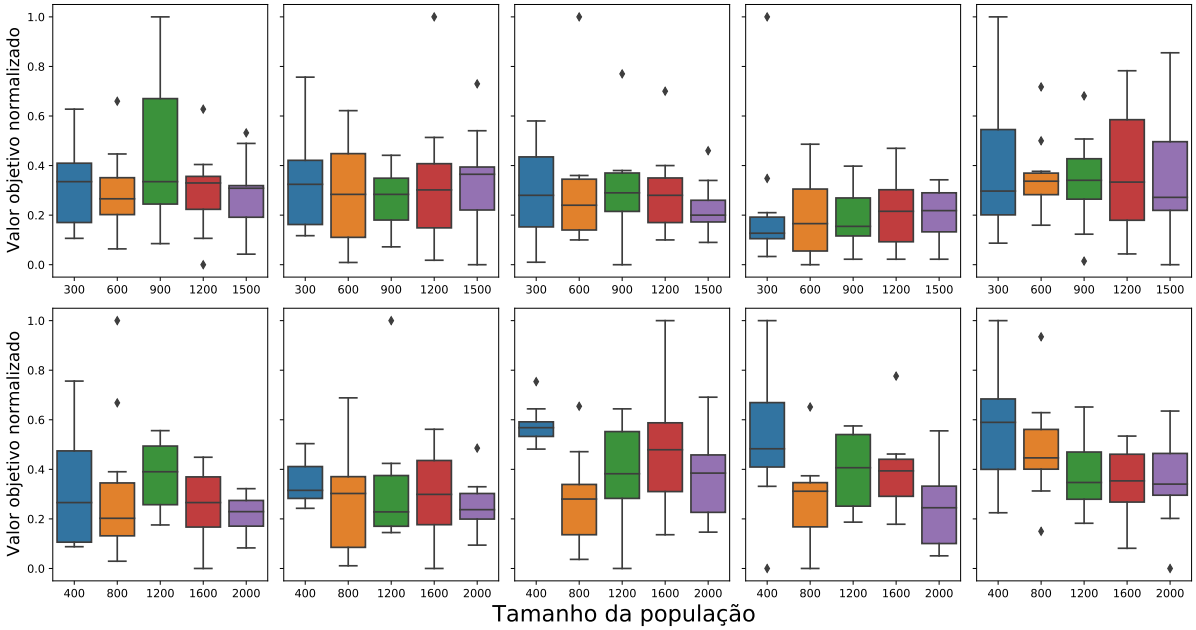


Figura 30 – Resultados da variação no tamanho da população, valores objetivo normalizados, modelo sem ociosidade de berços.

Na Figura 30, a primeira linha apresenta os resultados para as instâncias com 30 navios e 3 berços, a segunda linha para 40 navios e 5 berços. Em cada linha, cada gráfico

representa uma instância do respectivo tamanho, o primeiro gráfico da linha representa a instância com sufixo -1, o segundo com sufixo -2, etc. Ou seja, o primeiro gráfico da primeira linha representa os resultados para a instância f30x3-1, o primeiro gráfico da segunda linha representa os resultados para a instância f40x5-1, e assim por diante.

Os valores da função objetivo encontradas pelo BRKGA foram normalizados para uma melhor visualização da variação do método ao executá-lo 10 vezes para cada configuração de parâmetros. Podemos observar que a média do valor objetivo encontrado para a mesma instância com diferentes tamanhos da população não varia muito para as instâncias de 30 navios e 3 berços, apresentando uma variação um pouco maior para as instâncias de 40 navios e 5 berços. Além disso, não é possível perceber um padrão como, por exemplo, quanto maior o tamanho da população melhores os valores objetivos encontrados, ou para um determinado multiplicador observamos para todas as instâncias que retorna os melhores valores objetivos.

Como o BRKGA não é um algoritmo determinístico, cada vez que executamos obtemos resultados diferentes e, em alguns casos, podemos observar uma variação maior dos valores, como para a instância f30x3-01 com o tamanho da população 900 que variou de valores objetivo normalizados de menos que $0,2$ até $1,0$. Apesar de uma variação maior, a média continua próxima dos resultados para os outros tamanhos de população. Também não é possível observar algum padrão de um determinado multiplicador para o tamanho da população resultar em resultados com menor variação entre as 10 execuções para todas as instâncias testadas.

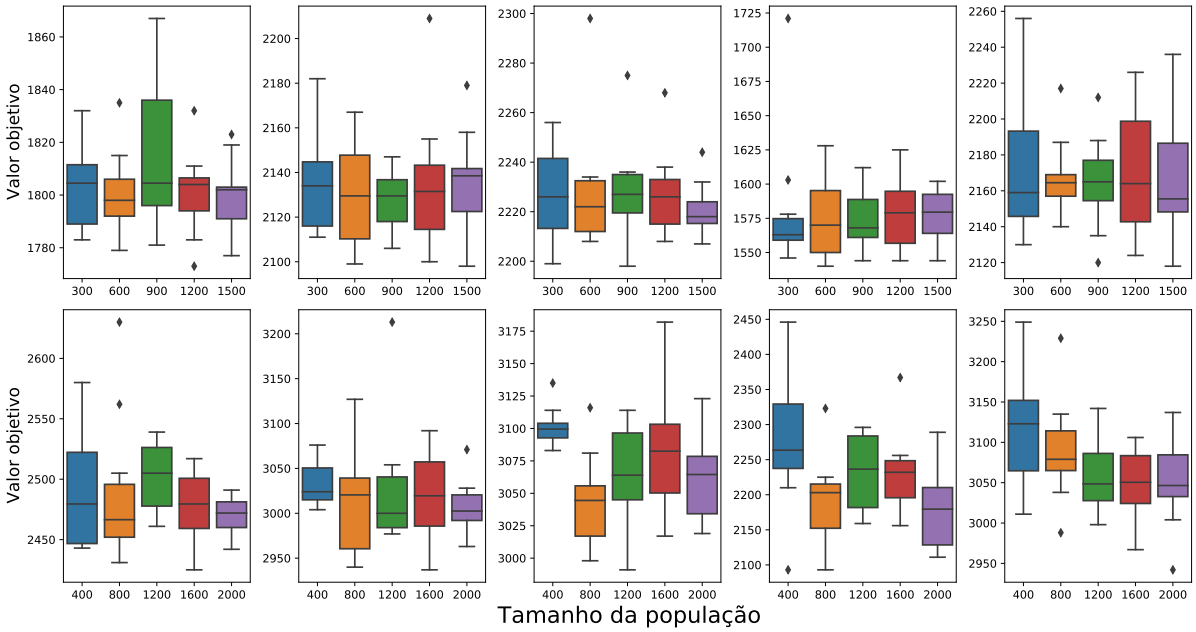


Figura 31 – Resultados da variação no tamanho da população, modelo sem ociosidade de berços.

A estrutura da Figura 31 é semelhante à Figura 30, com o eixo y representando os valores objetivos da solução, sem normalização. Com essa figura, é possível analisar que aumentar o tamanho da população não necessariamente resulta em uma melhoria no valor objetivo, o que pode estar associado à variabilidade da população gerada. Na Figura 31 podemos observar o caso da variação no valor objetivo da instância f30x3-01 com tamanho de 900, variando próximo a 100 unidades, com um mínimo em torno de 1780, uma variação de em torno de $5,6\%$ do menor valor.

5.4.1.0.2 Modelo com ociosidade de berços

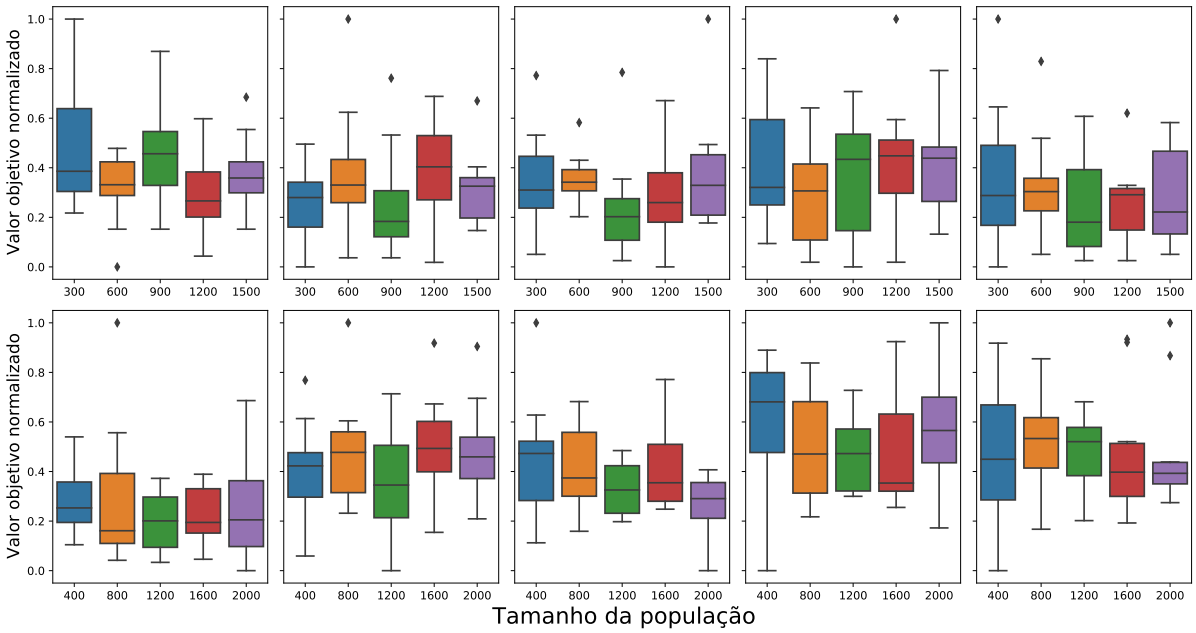


Figura 32 – Resultados da variação no tamanho da população, valores objetivo normalizados, modelo com ociosidade de berços.

As Figuras 32 e 33 descrevem as distribuições dos resultados dos testes para o modelo com ociosidade de berços, com estrutura equivalente às Figuras 30 e 31, respectivamente.

Da mesma forma que para o modelo sem ociosidade de berços com relação a variabilidade nos resultados para cada iteração dentro das 10 execuções. Com relação aos valores objetivos, é possível observar uma variação um pouco maior das médias do que para o modelo sem ociosidade de berços, mas também sem a identificação de um padrão de multiplicador que possa fornecer valores objetivos ou variações nos valores menores para todas as instâncias.

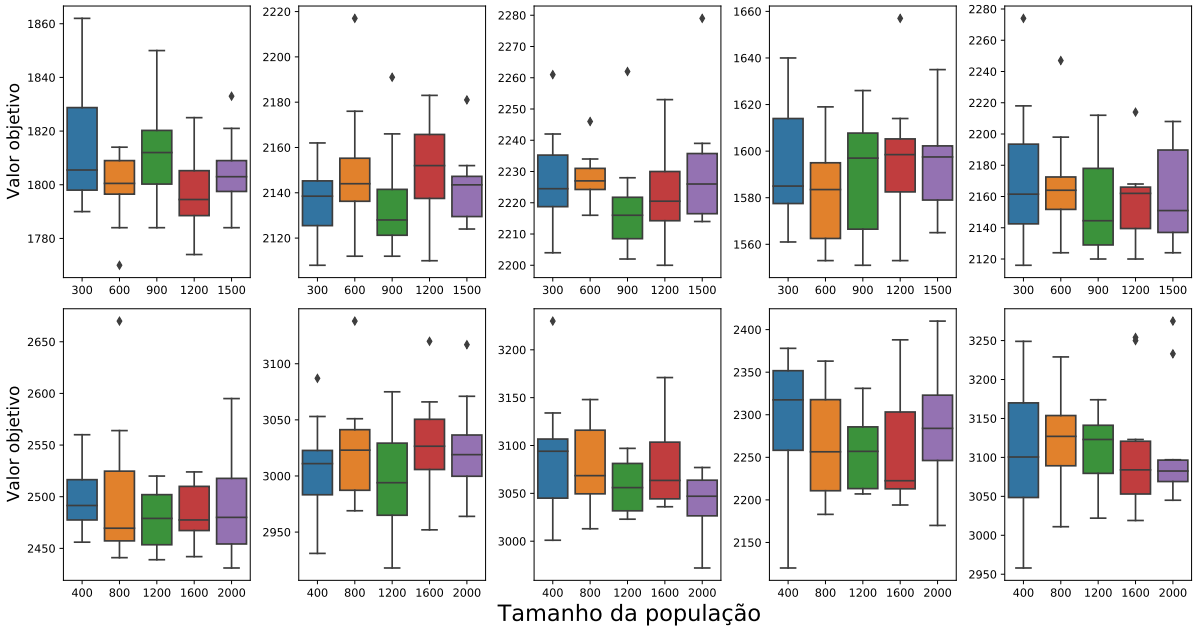


Figura 33 – Resultados da variação no tamanho da população, modelo com ociosidade de berços.

5.4.2 Múltiplas rodadas com parâmetros fixos

Como o BRKGA não é determinístico e pode obter resultados diferentes com os mesmos parâmetros, testamos executar o BRKGA 50 vezes para cada instância, de forma paralela e independente, com os parâmetros fixos e definidos empiricamente, para obter o melhor valor objetivo obtido dentre as rodadas, uma estratégia já indicada na literatura (Whitley et al., 1999, Prasetyo et al., 2015, Toso and Resende, 2015).

Os parâmetros utilizados para os testes para os modelos sem e com ociosidade de berços foram os mesmo e são descritos por:

- Fração população elite: $0,2$.
- Fração população mutante: $0,2$.
- Probabilidade de herdar gene do pai elite: $0,7$.
- Multiplicador que gera o tamanho da população: 30.
- Tolerância para diferença entre melhor *fitness* entre gerações: 10^{-04}.
- Máximo de gerações com variação de *fitness* menor que tolerância: 20.

5.4.2.0.1 Modelo sem ociosidade de berços

Na Tabela 17 podemos conferir os resultados para o modelo sem ociosidade de berços, rodando 50 vezes com os parâmetros do BRKGA fixos. As informações são dos menores valores objetivo encontrado entre todas as rodadas (*Obj. min*), o maior (*Obj. max*), a diferença relativa em porcentagem entre o menor e o maior valor objetivo com relação ao menor, assim como a média e o desvio padrão dos valores objetivo encontrados e o tempo total em minutos.

Na Figura 34 podemos ver histogramas das distribuições dos valores objetivos encontrados para cada instância, também para o modelo sem ociosidade de berços.

Instância	Obj. min	Obj. max	Dif. (%)	Média	Desv. Pad.	Tempo (s)
f30x3-01	1767	1813	2,6	1786,1	13,0	3,21
f30x3-02	2094	2187	4,4	2123,1	18,5	3,28
f30x3-03	2197	2264	3,0	2212,4	13,7	3,36
f30x3-04	1550	1635	5,5	1582,4	18,8	2,97
f30x3-05	2120	2194	3,5	2148,8	17,1	3,07
f40x5-01	2335	2534	8,5	2375,4	31,1	11,62
f40x5-02	2853	2990	4,8	2887,4	27,6	11,40
f40x5-03	2894	3039	5,0	2934,0	28,3	11,84
f40x5-04	2024	2189	8,2	2087,0	35,9	11,63
f40x5-05	2846	3084	8,4	2941,0	57,2	10,66
f55x5-01	4705	4865	3,4	4758,2	29,9	28,99
f55x5-02	5498	5740	4,4	5559,3	42,8	27,12
f55x5-03	5529	5756	4,1	5588,5	54,2	28,13
f55x5-04	4225	4400	4,1	4276,4	36,7	27,32
f55x5-05	5498	5706	3,8	5566,0	56,7	28,75
f60x5-01	5780	6129	6,0	5841,8	69,8	34,49
f60x5-02	6898	7058	2,3	6982,2	38,3	37,81
f60x5-03	6814	7179	5,4	6916,1	97,5	39,26
f60x5-04	5146	5393	4,8	5221,8	57,4	39,05
f60x5-05	6737	6917	2,7	6791,0	40,7	38,60

Tabela 17 – Múltiplas rodadas BRKGA com parâmetros fixos para modelo sem ociosidade de berços

Da Tabela 17 podemos observar que o percentual de diferença entre o maior e menor valores objetivo encontrados pelo BRKGA rodando 50 vezes não passou de 9%, uma diferença que, em termos monetários, pode ser bastante considerável para a economia e logística de um porto. Analisando a coluna que fornece o tempo total das múltiplas execuções em paralelo, em segundos. Podemos notar que apesar de os tempos totais das 50 execuções em paralelo aumentarem com a dimensão da instância e serem maiores do que os tempos de uma única execução, também não chega a 1 minuto.

Da Figura 34 podemos notar que em algumas instâncias a maior parte das soluções estão em torno dos menores valores objetivo, como é o caso das instâncias f30x3-01, f30x3-03, f40x5-02, f55x5-05 e f60x5-03. Contudo, a maioria dos casos isso não acontece,

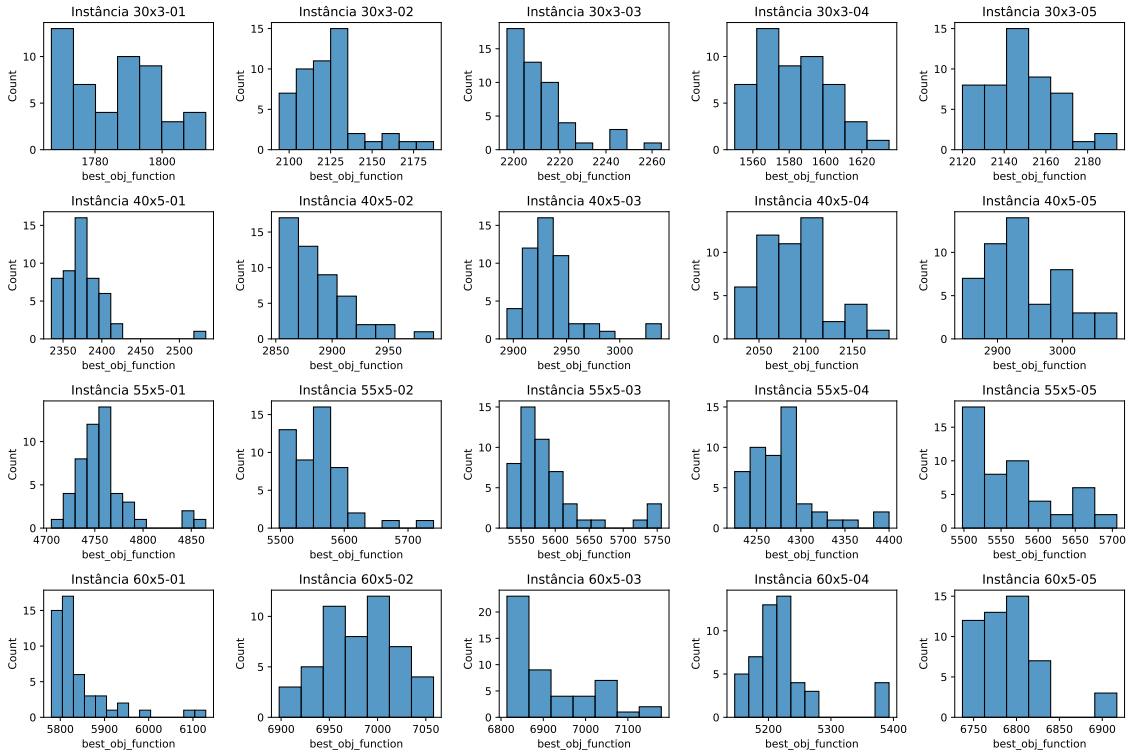


Figura 34 – Histograma valores objetivo de múltiplas rodadas BRKGA com parâmetros fixos, modelo sem ociosidade.

podendo ter uma baixa concentração de soluções em torno do valor objetivo mais baixo encontrado durante as múltiplas execuções. Ou seja, a probabilidade de obter as melhores soluções não é garantida. Assim, podemos concluir que vale a pena executar múltiplos BRKGAs para adotar o melhor valor objetivo encontrado, pois podemos garantir um melhor valor objetivo com um tempo computacional ainda reduzido.

5.4.2.0.2 Modelo com ociosidade de berços

Para o modelo com ociosidade dos berços, as informações dos 50 testes para cada instância são apresentadas na Tabela 18, com as informações do menor e maior valor objetivo encontrado, a diferença em porcentagem entre esses valores, relativa ao menor valor encontrado, assim como a média e desvio padrão dos valores objetivos encontrados, para o modelo com ociosidade de berços. Na Figura 35 temos os histogramas dos valores objetivos encontrados para cada instância, que possibilita analisar a distribuição dos dados.

Da mesma maneira que para o caso do modelo sem ociosidade de berços, observamos que não há um padrão de comportamento de que a maioria das soluções dentre as 50 execuções sejam mais próximas do menor valor objetivo encontrado dentre

Instância	Obj. min	Obj. max	Dif. (%)	Média	Desv. Pad.	Tempo (s)
f30x3-01	1766	1833	3,8	1787,9	15,4	3,04
f30x3-02	2101	2172	3,4	2125,4	14,4	2,98
f30x3-03	2194	2238	2,0	2211,2	11,8	3,01
f30x3-04	1553	1630	5,0	1591,8	17,9	3,10
f30x3-05	2124	2244	5,6	2154,8	24,8	3,30
f40x5-01	2328	2469	6,1	2377,1	23,5	11,00
f40x5-02	2862	2947	3,0	2896,3	20,6	10,96
f40x5-03	2904	3097	6,6	2943,5	27,6	11,75
f40x5-04	2076	2223	7,1	2126,6	34,9	10,35
f40x5-05	2870	3108	8,3	2930,2	48,7	10,15
f55x5-01	4717	4917	4,2	4767,0	40,0	28,02
f55x5-02	5507	5622	2,1	5564,0	30,0	27,06
f55x5-03	5519	5871	6,4	5606,6	65,3	28,57
f55x5-04	4229	4595	8,7	4304,6	66,2	28,70
f55x5-05	5502	6022	9,5	5581,3	92,5	28,38
f60x5-01	5786	5971	3,2	5832,7	43,9	37,90
f60x5-02	6913	7221	4,5	7008,6	68,8	35,56
f60x5-03	6808	7297	7,2	6901,5	86,6	32,86
f60x5-04	5147	5490	6,7	5236,4	62,5	36,24
f60x5-05	6741	6963	3,3	6794,0	53,5	35,79

Tabela 18 – Múltplas rodadas BRKGA com parâmetros fixos para modelo com ociosidade de berços.

as rodadas. Pelos histogramas da Figura 35, podemos observar que para o modelo com ociosidade de berços a ocorrência de *outliers* foi maior que para o modelo sem ociosidade dos berços, o que também pode ser observado na coluna de "Desv. Pad." da Tabela 18. Assim como nos testes de variação do tamanho da população, foi observado uma maior variação nos resultados dos testes para o modelo com ociosidade do que sem.

Com relação aos tempos totais das 50 execuções, também observamos o impacto do tamanho da dimensão e os tempos são maiores do que uma única execução, mas também não ultrapassam 40 segundos. Os tempos para o modelo com ociosidade dos berços são um pouco maiores do que para o modelo sem ociosidade, o que indica que o aumento na complexidade do modelo não afeta significativamente a dificuldade de resolução pelo BRKGA, seja com execução única ou com múltiplas execuções.

5.4.3 Definição dos parâmetros com a ferramenta OPTUNA

A ferramenta OPTUNA [1] é um *framework* para automatização da busca de hiper parâmetros com foco em modelos de aprendizagem de máquina, mas que pode ser usado para heurísticas e meta-heurísticas [2].

[1] Repositório OPTUNA.
[2] OPTUNA.

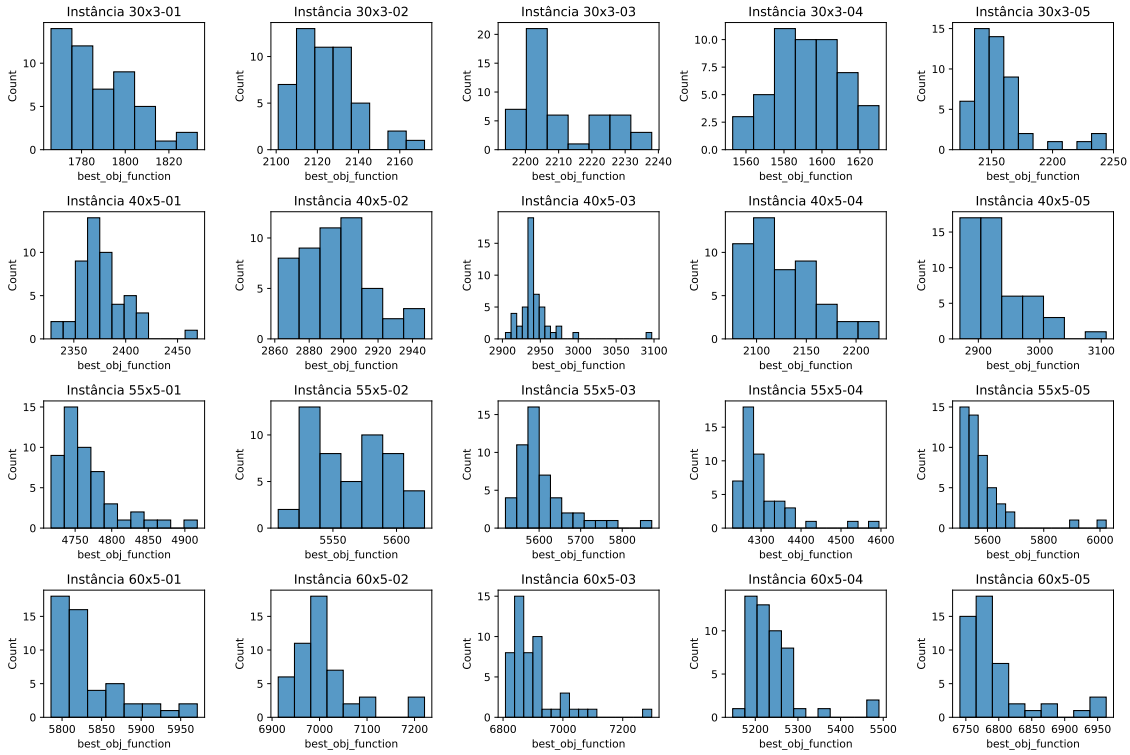


Figura 35 – Histograma valores objetivo encontrados pelo BRKGA em múltiplas rodadas com parâmetros fixos, modelo com ociosidade.

A ferramenta tem como base dois conceitos, o estudo (*study*) e o experimento (*trial*). O estudo é uma otimização baseada em uma função objetivo (métrica) e que busca a melhor combinação dos hiper parâmetros através de experimentos. O experimento é uma única execução da função objetivo, com valores de hiper parâmetros definidos. Com isso, a ferramenta realiza uma busca de diferentes combinações de valores para os hiper parâmetros avaliando a qualidade de cada combinação pelo valor objetivo definido.

O OPTUNA disponibiliza diferentes algoritmos que geram amostras de hiper parâmetros e podam combinações pouco promissoras, restringindo continuamente o espaço de busca [3]. O algoritmo de busca padrão é o *TPESampler* (*Tree-structured Parzen Estimator*), que utiliza amostragem independente, isto é, determina o valor de um parâmetro sem considerar relações entre parâmetros e, para cada experimento e cada parâmetro, o *TPE* ajusta um Modelo de Mistura Gaussiana (*Gaussian Mixture Model - GMM*).

Na otimização do estudo, os critérios de parada para a busca podem ser pela quantidade de experimentos ou pelo tempo, em segundos, que a busca irá durar. Além disso, é possível definir a quantidade experimentos que correrão em paralelo.

Juntamente com o OPTUNA, utilizamos a ferramenta *MLflow* [4], para visuali-

[3] OPTUNA: algoritmos que geram amostras.
[4] Documentação Mlflow

zação e análise dos dados. Com o *Mlflow* integrado com o OPTUNA [5] pudemos visualizar gráficos com os resultados de todos os experimentos, selecionando diferentes parâmetros em uma mesma visualização, buscando padrões de comportamentos na variação dos parâmetros resultantes dos experimentos, com os valores objetivo resultantes, através de registros dos experimentos. O *Mlflow* é comumente usado para gerenciar todo o ciclo de projetos de aprendizagem de máquina e, uma vez que o OPTUNA também é mais usado para modelos de aprendizagem de máquina, o OPTUNA fornece uma integração com o *Mlflow*[6], simplificando o gerenciamento dos experimentos.

Os testes realizados com o OPTUNA incluíram a busca dentro do intervalo de valores do parâmetros definidos na literatura, descritos na Tabela 6 (Prasetyo et al. (2015)), e também relaxando os intervalos para incluir mais valores e analisar o comportamento do BRKGA. Nos testes realizados, o tamanho da população é definido por uma constante multiplicada pela quantidade de navios e não pelo tamanho do cromossomo, ou seja, difere do proposto na literatura. Contudo, o tamanho do cromossomo está relacionado com a quantidade de navios, de forma que o cromossomo é um múltiplo da quantidade de navios.

Para os testes com os intervalos dentro do proposto na literatura temos as seguintes definições:

- Multiplicador que define o tamanho da população: $[10; 50]$.
- Tamanho da população elite: fração da população total que defina o tamanho da população elite. O intervalo considerado para busca foi: $[0,1; 0,25]$.
- Tamanho da população mutante: fração da população total que defina o tamanho da população mutante. O intervalo considerado para busca foi: $[0,1; 0,3]$.
- Probabilidade de herdar gene do pai elite: intervalo de busca considerado foi $[0,5; 0,8]$.
- Quantidade máxima de gerações: $[100; 5000]$.
- Quantidade máxima de gerações com mudança no valor objetivo da melhor solução abaixo do nível de tolerância: $[10; 30]$. Caso a quantidade de gerações consecutivas com diferença entre o objetivo do melhor indivíduo for menor que a tolerância definida, o BRKGA encerra sua busca.
- Tolerância para a diferença entre melhores valores objetivos para gerações consecutivas: $[10^{-06}; 10^{-04}]$.

A busca utilizada foi a padrão, isto é, a *TPESampler*, com o critério de parada sendo o tempo de busca, limitado a 30 minutos, considerando que o tempo máximo

[5] Mlflow integrado com OPTUNA
[6] *API OPTUNA Integration.*

permitido para o CPLEX resolver o BAP sem ociosidade foi de uma hora. A quantidade de experimentos em paralelo foi definido com o máximo de CPUs disponíveis na máquina, podendo utilizar todos ou não. As instâncias utilizadas nos testes foram as descritas na Tabela 7 e, além dos parâmetros clássicos descritos na Tabela 13, os parâmetros responsáveis pelos critérios de parada também foram configurados para serem definidos na otimização pela ferramenta.

5.4.3.1 PAB sem ociosidade de berços

Na Tabela 19 podemos observar os melhores valores objetivos encontrados pelo OPTUNA, os tempos totais de processamento e os tempos até encontrar a melhor solução, assim como a porcentagem da razão entre esses tempos (coluna "Razão"), que indica a porcentagem do tempo total que o OPTUNA levou para alcançar a melhor solução. Desta tabela, é possível perceber que não há um padrão para o tempo que a ferramenta leva para encontrar a melhor solução dentro da busca.

Já na Tabela 20 vemos a comparação entre as soluções obtidas pelo OPTUNA e as soluções obtidas das execuções independentes do BRKGA com parâmetros fixos, apresentadas na Seção 5.3 (única execução), Tabela 14 (Obj_{BRKGA1}), e na Seção 5.4.2 (múltiplas execuções), Tabela 17 (Obj_{BRKGA2}), assim como os *gaps* referentes às diferenças do objetivo encontrado pelo OPTUNA com os outros dois valores objetivos, calculados a partir da fórmula:

$$gap = \frac{(Obj_{BRKGA} - Obj_{OPT}) \times 100}{Obj_{BRKGA}}. \tag{5.6}$$

O $gap1$ é calculado a partir do Obj_{BRKGA1} e o gap a partir do Obj_{BRKGA2}.

Da Tabela 20, podemos observar que o OPTUNA obteve melhores valores objetivo em quase todos os casos, com exceção de um único, principalmente para as instâncias maiores com 55 e 60 navios e 5 berços, com a porcentagem de melhora com relação a uma única execução é maior do que com as múltiplas execuções. A instância em que o OPTUNA não obteve melhor valor objetivo foi a f55x5-01, mas com um gap de $0,1\%$ e, para os outros casos, a melhora não superou o gap de 1% do Obj_{BRKGA2}.

Na Tabela 21 estão os tempos de processamento em minutos rodando o BRKGA pelo OPTUNA (T_{OPT}) e os tempos processando o BRKGA uma única vez (T_{BRKGA1}) e múltiplas vezes (T_{BRKGA2}) cujos valores objetivo foram apresentados na Tabela 20.

Da Tabela 21, o tempo computacional utilizado para obter as soluções através de múltiplas execuções em paralelo foi muito menor que pelo OPTUNA. Mesmo considerando a soma dos tempos de cada execução ao invés de considerar em paralelo, os tempos foram menores com, no máximo, $18,613$ minutos para a instância f60x5-05.

Instância	Obj_{OPT}	Tempo total (min)	Tempo melhor sol. (min)	Razão (%)
f30x3-01	1763	30,462	29,119	95,591
f30x3-02	2091	30,372	23,494	77,354
f30x3-03	2188	30,711	29,538	96,181
f30x3-04	1538	31,154	31,146	99,974
f30x3-05	2114	31,156	12,187	39,116
f40x5-01	2321	30,626	7,474	24,404
f40x5-02	2836	31,014	2,464	7,945
f40x5-03	2888	30,537	11,363	37,211
f40x5-04	2013	33,275	22,611	67,952
f40x5-05	2832	30,987	13,899	44,854
f55x5-01	4711	33,071	28,103	84,978
f55x5-02	5492	33,198	19,010	57,262
f55x5-03	5511	36,242	28,802	79,471
f55x5-04	4203	32,960	32,935	99,924
f55x5-05	5488	39,107	39,097	99,974
f60x5-01	5771	34,450	34,446	99,988
f60x5-02	6896	34,667	28,372	81,842
f60x5-03	6798	35,574	21,228	59,673
f60x5-04	5130	36,942	36,937	99,986
f60x5-05	6731	34,164	34,147	99,950
Média				67,682

Tabela 19 – Resultados OPTUNA com intervalo de valores da literatura para PAB sem ociosidade de berços.

A melhora do valor objetivo do OPTUNA com relação a uma única execução foi maior do que para múltiplas execuções, ambas com parâmetros fixos, sendo mais significativa considerando as instâncias maiores de 55 e 60 navios. Comparando com as múltiplas execuções, as porcentagens de melhora dos valores objetivos obtidos pelo OPTUNA não superaram 1%, o que pode indicar uma certa robustez do BRKGA com relação aos valores objetivos e variação nos parâmetros.

Para uma aplicação em problema real, é preciso entender se compensa o *trade-off* entre a melhora na função objetivo e o aumento no tempo de processamento, seria necessário levar em consideração se o tempo entre 30 e 40 minutos é viável no planejamento do porto e se a porcentagem de melhora resulta em uma economia financeira e de planejamento relevante. Além disso, vale analisar o quanto essa melhora na solução fornecida pelo OPTUNA afeta o CPLEX com *warmstart*, ou seja, com uma solução inicial.

Na busca feita pelo OPTUNA, diferentes configurações testadas alcançaram o menor (mesmo) valor objetivo. Na Tabela 22 estão descritas uma das diferentes combinações de parâmetros que alcançou o menor valor objetivo na busca para cada instância testada. Nas Tabelas 53 e 54 estão descritas todas as configurações de parâmetros que alcançaram os menores valores da função objetivo para cada instância, nas quais é possível observar

Instância	Obj_{BRKGA1}	Obj_{BRKGA2}	Obj_{OPT}	$gap1$	$gap2$
f30x3-01	1779	1767	1763	0,899	0,226
f30x3-02	2132	2094	2091	1,923	0,143
f30x3-03	2222	2197	2188	1,530	0,410
f30x3-04	1615	1550	1538	4,768	0,774
f30x3-05	2124	2120	2114	0,471	0,283
f40x5-01	2384	2335	2321	2,643	0,600
f40x5-02	2949	2853	2836	3,832	0,596
f40x5-03	2935	2894	2888	1,601	0,207
f40x5-04	2074	2024	2013	2,941	0,543
f40x5-05	2938	2846	2832	3,608	0,492
f55x5-01	4963	4705	4711	5,078	-0,128
f55x5-02	5654	5498	5492	2,865	0,109
f55x5-03	5627	5529	5511	2,061	0,326
f55x5-04	4502	4225	4203	6,642	0,521
f55x5-05	5536	5498	5488	0,867	0,182
f60x5-01	5919	5780	5771	2,500	0,156
f60x5-02	7002	6898	6896	1,514	0,029
f60x5-03	7004	6814	6798	2,941	0,235
f60x5-04	5492	5146	5130	6,591	0,311
f60x5-05	7023	6737	6731	4,158	0,089
Média				2,972	0,305

Tabela 20 – Comparação valores objetivos OPTUNA com BRKGA para parâmetros fixos, PAB sem ociosidade dos berços.

que a variação nas combinações de parâmetros que resultaram na mesma e melhor solução é maior nas instâncias menores.

Nas Tabelas 53 e 54 estão descritas todas as configurações de parâmetro, para cada instância, que alcançaram o valor objetivo mínimo, para o modelo sem ociosidade de berços. A Tabela 53 fornece os resultados para as instâncias de 30 navios e 3 berços, e a Tabela 54 para as outras instâncias.

Dessas tabelas, é possível observar que as instâncias com 30 navios e 3 berços, principalmente a instância f30x3-04, possuem muitas configurações diferentes que alcançaram o menor valor objetivo. Além disso, podemos observar uma alta variação nos valores de cada parâmetro. Para a instância f30x3-04 temos, por exemplo, que o maior valor do parâmetro P. 1 é em torno de 50% maior que o valor mínimo, já para o parâmetro P. 2 o maior valor chega a ser quase 80% maior que o menor.

Os parâmetros associados a cada coluna dessas tabelas, assim como os respectivos limites definidos para o OPTUNA, são dados por:

- P. 1: fração da população que é considerada elite, limites $[0,1; 0,25]$;
- P. 2: probabilidade de um herdeiro herdar um gene do pai elite no cruzamento,

Instância	T_{OPT}(min)	T_{BRKGA1}(min)	T_{BRKGA2}(min)
f30x3-01	30,462	0,0269	0,0535
f30x3-02	30,372	0,0325	0,0547
f30x3-03	30,711	0,0335	0,0560
f30x3-04	31,154	0,0144	0,0495
f30x3-05	31,156	0,0208	0,0512
f40x5-01	30,626	0,1654	0,1937
f40x5-02	31,014	0,0462	0,1900
f40x5-03	30,537	0,1382	0,1973
f40x5-04	33,275	0,1556	0,1938
f40x5-05	30,987	0,1819	0,1777
f55x5-01	33,071	0,1381	0,4832
f55x5-02	33,198	0,3302	0,4520
f55x5-03	36,242	0,1936	0,4688
f55x5-04	32,960	0,1710	0,4553
f55x5-05	39,107	0,2872	0,4792
f60x5-01	34,450	0,2685	0,5748
f60x5-02	34,667	0,3970	0,6302
f60x5-03	35,574	0,1533	0,6543
f60x5-04	36,942	0,0913	0,6508
f60x5-05	34,164	0,1858	0,6433

Tabela 21 – Comparação tempos computacionais OPTUNA e BRKGA fixo, problema sem ociosidade de berços.

limites $[0,5; 0,8]$;

- P. 3: fração da população reservada para os indivíduos mutantes, limites $[0,1; 0,3]$;
- P. 4: multiplicador da quantidade de navios que define o tamanho da população, limites $[10; 50]$;
- P. 5: tamanho da população (definido pelo Param. 4 multiplicado pela quantidade de navios), limites $[10 \times n; 50 \times n]$;
- P. 6: quantidade máxima de gerações a serem geradas (critério de parada), limites $[100; 5000]$;
- P. 7: quantidade máxima de gerações cuja variação do melhor valor objetivo entre gerações consecutivas é menor que uma tolerância (critério de parada), limites $[10; 30]$;
- P. 8: tolerância da variação do melhor valor objetivo entre uma geração e outra (associada ao Param. 7), limites $[10^{-06}; 10^{-04}]$.

Na Figura 36 podemos observar os valores dos parâmetros combinados, sem os que são associados a critério de parada, para uma das cinco instâncias de cada tamanho.

Instância	P. 1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8
f30x3-01	0,232	0,718	0,152	49	1470	3400	26	0,00005
f30x3-02	0,197	0,629	0,108	44	1320	1500	26	0,00060
f30x3-03	0,212	0,645	0,121	44	1320	2400	27	0,00077
f30x3-04	0,246	0,608	0,100	49	1470	3600	11	0,00073
f30x3-05	0,202	0,606	0,179	39	1170	2000	28	0,00056
f40x5-01	0,111	0,724	0,213	42	1680	700	19	0,00092
f40x5-02	0,126	0,528	0,193	41	1640	1800	26	0,00024
f40x5-03	0,134	0,596	0,183	49	1960	4800	21	0,00063
f40x5-04	0,150	0,550	0,198	50	2000	2500	30	0,00084
f40x5-05	0,192	0,656	0,241	43	1720	1200	18	0,00049
f55x5-01	0,114	0,703	0,128	48	2640	200	25	0,00047
f55x5-02	0,142	0,612	0,157	49	2695	2100	30	0,00030
f55x5-03	0,210	0,702	0,220	35	1925	4000	30	0,00029
f55x5-04	0,175	0,737	0,195	49	2695	1800	26	0,00085
f55x5-05	0,205	0,594	0,174	48	2640	1600	30	0,00029
f60x5-01	0,131	0,614	0,112	40	2400	1600	20	0,00040
f60x5-02	0,234	0,675	0,198	38	2280	3500	24	0,00006
f60x5-03	0,236	0,656	0,151	48	2880	2600	25	0,00040
f60x5-04	0,118	0,605	0,289	49	2940	4000	30	0,00033
f60x5-05	0,197	0,756	0,179	45	2700	3600	13	0,00009
Média	0,178	0,646	0,175	45		2445	24	0,00047
Desv.Pad.	0,045	0,063	0,048	4,5		1218,49	5,6	0,00027

Tabela 22 – Valores dos parâmetros da melhor solução do OPTUNA, PAB sem ociosidade de berços.

Cada reta representa uma combinação dos parâmetros, ou seja, os pontos de fração elite ($elite\_fraction$), probabilidade de herdar o gene do pai elite ($elite\_parent\_prob$), a fração da população mutante ($mutant\_fraction$) e o multiplicador que com a quantidade de navios define o tamanho da população ($pop\_size\_multiplier$) que estão na mesma reta, representam uma combinação. Por último, temos o valor objetivo encontrado por essa combinação ($objective\_function$).

Da Figura 36 é possível observar que para a instância de tamanho 30 navios e 3 berços, mais combinações diferentes levam para um valor objetivo menor. Já para os outros tamanhos, é possível perceber intervalos com densidade maior de linhas que levam a valores objetivo menores. Como para a instância f40x5-01, o intervalo $(0; 0,14]$ para a fração elite, o $(0,6; 0,75)$ para a probabilidade de herdar do pai elite, o $(0,2; 0,25)$ para a fração mutante e $(40, 50)$ para o multiplicador do tamanho da população costumam gerar valores objetivos menores.

Analisando cada parâmetro para diferentes instâncias, também podemos observar uma alta variabilidade nos valores. Podemos notar que o valor do desvio padrão do parâmetro da população elite é em torno de 25% da média, o que indica uma alta

variabilidade dos dados comparado com a média, assim como para a fração da quantidade de indivíduos mutantes, cujo desvio padrão é em torno de 27%, para a quantidade máxima de gerações, que representa quase 50% do valor da média, para o parâmetro P. 7 cujo desvio padrão representa em torno de 23% da média e para a tolerância, ou parâmetro P. 8, que representa em torno de 57% da média.

Como a variabilidade é alta para esses parâmetros, não podemos concluir que há um valor médio para esses parâmetros que pode ser aplicado com desempenho equivalente para diferentes instâncias. Contudo, retomando os resultados da Tabela 20 os *gaps* comparando com valores objetivos do OPTUNA com múltiplas execuções e parâmetros fixos não são tão significativos, mesmo considerando os mesmos parâmetros para todas as instâncias.

Para o parâmetro P. 7, podemos notar que muitos valores foram definidos como igual ou muito próximos do limite superior, como não há um intervalo indicado na literatura (critério de parada), seu intervalo foi definido de maneira empírica e pode ter seu limite superior aumentado para possibilitar valores mais altos.

5.4.3.1.1 Relaxando os intervalos de busca dos parâmetros:

além dos testes com os limites de busca dentro dos propostos na literatura (descritos na Tabela 6), foram feitos testes relaxando os intervalos para avaliar o desempenho do BRKGA, com o critério de parada também de trinta minutos, com a possibilidade de usar todos os CPUs disponíveis na máquina. Apesar de relaxar os tamanhos dos intervalos, buscamos fazer de tal forma a manter as características do método elitista do BRKGA como, por exemplo, manter o tamanho da população elite menor que metade da população ou a probabilidade de herdar o gene do pai elite maior do que o não elite. Os intervalos para a busca dos parâmetros foram definidos por:

- Multiplicador da quantidade de navios que define o tamanho da população: $[10; 50]$ (P. 4);
- fração que define o tamanho da população elite: $[0,01; 0,4]$ (P. 1);
- fração que define o tamanho da população mutante: $[0,01; 0,4]$ (P. 3);
- probabilidade de herdar gene do pai elite no cruzamento: $[0,5; 1,0]$ (P. 2);
- quantidade máxima de gerações: $[100, 5000]$, com tamanho de passo 100 (P. 6);
- tolerância na variação do melhor indivíduo entre gerações: $[10^{-06}; 10^{-01}]$ (P. 8);
- quantidade máxima de gerações cuja diferença entre as melhores soluções é menor que a tolerância antes do algoritmo parar: $[10; 50]$ (P. 7).

Lembrando que o parâmetro P. 5 é referente ao tamanho da população, definido pelo parâmetro P. 4 multiplicado pela quantidade de navios n. Os parâmetros encontrados pelo OPTUNA para a melhor solução estão descritos na Tabela 23.

Instância	P. 1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8
f30x3-01	0,30418	0,59288	0,11674	39	1170	600	40	0,00080
f30x3-02	0,22401	0,54261	0,16060	38	1140	1500	32	0,00340
f30x3-03	0,30060	0,55786	0,22573	46	1380	3300	45	0,00364
f30x3-04	0,31958	0,68443	0,22416	36	1080	2000	31	0,00002
f30x3-05	0,20050	0,55492	0,13827	47	1410	4800	33	0,00108
f40x5-01	0,24185	0,68314	0,11104	48	1920	4400	42	0,00098
f40x5-02	0,31048	0,66354	0,20561	50	2000	3500	48	0,00039
f40x5-03	0,29429	0,74207	0,16755	48	1920	1100	35	0,00893
f40x5-04	0,19705	0,59751	0,16498	48	1920	3700	44	0,00757
f40x5-05	0,27062	0,64573	0,12439	45	1800	3300	47	0,00353
f55x5-01	0,17225	0,57409	0,13988	34	1870	2600	42	0,00741
f55x5-02	0,20828	0,69320	0,19366	47	2585	1800	33	0,00175
f55x5-03	0,14363	0,61680	0,21500	49	2695	1700	39	0,00037
f55x5-04	0,20503	0,70091	0,25846	46	2530	2100	48	0,00671
f55x5-05	0,26993	0,65226	0,11657	40	2200	2000	32	0,00161
f60x5-01	0,18808	0,59869	0,18405	45	2700	4700	31	0,00427
f60x5-02	0,22355	0,73901	0,12338	49	2940	2700	31	0,00776
f60x5-03	0,14964	0,59266	0,11456	47	2820	4600	35	0,00541
f60x5-04	0,17288	0,64553	0,12312	35	2100	5000	18	0,00471
f60x5-05	0,08595	0,60779	0,15596	46	2760	3000	42	0,00881
Média	0,22412	0,63428	0,16319	44		2920	37	0,00395
Desv. Pad.	0,06406	0,05986	0,04467	5,11		1322,92	7,6	0,00305

Tabela 23 – Valores dos parâmetros das melhores soluções na busca por parâmetros pelo OPTUNA com intervalos de busca relaxados, PAB sem ociosidade de berços.

Com os resultados apresentados na Tabela 23, comparamos os valores dos parâmetros das melhores soluções com os intervalos indicados na literatura (Tabela 6). Dentre os parâmetros dos tamanhos das populações elite e mutante e a probabilidade de herdar o gene do pai elite, as médias estão dentro do intervalo indicado pela literatura, com o desvio padrão para o tamanho da população elite chegando a 30% da média, ocorrendo sete casos dos 20 que o parâmetro da melhor solução fica acima do valor indicado na literatura. Para a população mutante, apesar do valor do desvio padrão ser em torno de 27% da média, não houve instâncias em que para a melhor solução encontrada o valor do parâmetro está fora do indicado na literatura. Para a probabilidade de herdar gene do pai elite, nenhum dos parâmetros encontrados pelo OPTUNA estão fora do intervalo indicado na literatura, assim como o valor médio.

Comparando com os valores obtidos pelo OPTUNA com os limites dentro do indicado na literatura, encontrados na Tabela 22, as médias dos valores dos parâmetros encontrado para as instâncias são próximas para P. 2, P. 3, P. 4 e P. 5.

Pela tabela 24 podemos observar os valores objetivos encontrados com os intervalos relaxados e é possível notar que a maioria dos casos em que a fração da população que define a população elite está fora do intervalo indicado na literatura, o melhor objetivo encontrado ou é igual ou é um pouco pior, mas com uma diferença que não chega a 1% dos valores do OPTUNA para os parâmetros dentro do intervalo da literatura. A diferença entre os objetivos é menor que 1% em todos os casos, o que pode ser mais um indicador de que o BRKGA é um algoritmo mais robusto com relação ao valor objetivo. Além disso, todas as melhores soluções encontradas se mantém factíveis mesmo que entre os experimentos algumas soluções geradas sejam infactíveis.

Instância	Obj_{OPT} Relaxado	Obj_{OPT}	**gap**
f30x3-01	1763	1763	0,000
f30x3-02	2091	2091	0,000
f30x3-03	2188	2188	0,000
f30x3-04	1538	1538	0,000
f30x3-05	2114	2114	0,000
f40x5-01	2321	2321	0,000
f40x5-02	2841	2836	0,176
f40x5-03	2885	2888	-0,104
f40x5-04	2008	2013	-0,248
f40x5-05	2846	2832	0,494
f55x5-01	4707	4711	-0,085
f55x5-02	5492	5492	0,000
f55x5-03	5523	5511	0,218
f55x5-04	4178	4203	-0,595
f55x5-05	5486	5488	-0,036
f60x5-01	5777	5771	0,104
f60x5-02	6896	6896	0,000
f60x5-03	6805	6798	0,103
f60x5-04	5130	5130	0,000
f60x5-05	6729	6731	-0,030

Tabela 24 – Comparação dos melhores resultados obtidos pelo OPTUNA com os intervalos de parâmetros relaxados e dentro do sugerido pela literatura, PAB sem ociosidade de berços.

5.4.3.2 PAB com ociosidade de berços

Na Tabela 25 estão descritos os valores objetivos encontrados pelo OPTUNA, o tempo computacional total para processar cada instância (critério de parada do OPTUNA de 30 minutos), e o tempo computacional até chegar ao melhor valor objetivo na busca por bons parâmetros. A coluna *Razão* indica a porcentagem de tempo para encontrar o melhor valor objetivo com relação ao tempo total e é possível observar que, assim como para o modelo sem ociosidade de berços, não há um padrão no tempo para o OPTUNA

chegar na solução ótima, ocorrendo o caso de a melhor solução ser encontrada no final da execução (f60x5-01) ou logo no começo (f40x5-04).

Instância	Obj_{OPT}	Tempo total (min)	Tempo melhor sol. (min)	Razão (%)
f30x3-01	1764	31,049	13,343	42,975
f30x3-02	2098	31,284	23,591	75,410
f30x3-03	2188	30,501	17,100	56,062
f30x3-04	1551	30,612	30,608	99,988
f30x3-05	2114	30,697	30,659	99,878
f40x5-01	2318	32,773	4,528	13,816
f40x5-02	2849	30,813	19,293	62,613
f40x5-03	2892	32,632	21,072	64,574
f40x5-04	2058	33,586	2,585	7,696
f40x5-05	2846	32,803	32,777	99,922
f55x5-01	4709	34,994	19,419	55,490
f55x5-02	5506	34,834	21,340	61,262
f55x5-03	5515	32,614	5,268	16,153
f55x5-04	4190	37,622	29,503	78,418
f55x5-05	5490	31,473	31,469	99,987
f60x5-01	5775	36,834	36,830	99,988
f60x5-02	6912	33,358	7,083	21,235
f60x5-03	6806	36,314	14,393	39,635
f60x5-04	5138	35,439	16,743	47,245
f60x5-05	6737	36,427	28,132	77,228
Média				60,979

Tabela 25 – Resultados OPTUNA com intervalos de valores da literatura para PAB com ociosidade de berços.

Na Tabela 26 temos uma comparação entre os valores objetivo encontrados rodando o BRKGA com os parâmetros fixos, executando uma única vez, descritos na Tabela 15 (Obj_{BRKGA1}) e com múltiplas execuções, Tabela 18 (Obj_{BRKGA2}). Os valores dos *gaps* foram calculados de acordo com a fórmula 5.6, então o *gap* positivo indica que a solução encontrada pelo OPTUNA foi melhor do que o BRKGA com parâmetros fixos. Pode-se observar que o OPTUNA de fato encontra melhores valores objetivos do que definir empiricamente através alguns testes. A melhora mais relevante é de em torno de $8,2\%$ comparando com uma única execução enquanto que comparando com múltiplas execuções não chega a 1%, mas com uma diferença relevante no tempo computacional, uma vez que o tempo total de múltiplas execuções não chega a 40 segundos de execução e, mesmo somando os tempos de cada execução (caso de não usar paralelismo), não chega a 20 minutos.

Os valores para os parâmetros encontrados pelo OPTUNA que alcançaram o melhor valor objetivo não são únicos. Na Tabela 27 estão descritos uma configuração de parâmetros que alcançaram o melhor valor objetivo para cada instância. A descrição dos nomes das colunas segue abaixo:

Instância	Obj_{OPT}	Obj_{BRKGA1}	Obj_{BRKGA2}	$gap1$	$gap2$
f30x3-01	1764	1770	1766	0,3390	0,1132
f30x3-02	2098	2105	2101	0,3325	0,1428
f30x3-03	2188	2209	2194	0,9507	0,2735
f30x3-04	1551	1578	1553	1,7110	0,1288
f30x3-05	2114	2140	2124	1,2150	0,4708
f40x5-01	2318	2412	2328	3,8972	0,4296
f40x5-02	2849	2871	2862	0,7663	0,4542
f40x5-03	2892	2939	2904	1,5992	0,4132
f40x5-04	2058	2151	2076	4,3236	0,8671
f40x5-05	2846	2928	2870	2,8005	0,8362
f55x5-01	4709	4809	4717	2,0794	0,1696
f55x5-02	5506	5679	5507	3,0463	0,0182
f55x5-03	5515	5704	5519	3,3135	0,0725
f55x5-04	4190	4565	4229	8,2147	0,9222
f55x5-05	5490	5536	5502	0,8309	0,2181
f60x5-01	5775	5952	5786	2,9738	0,1901
f60x5-02	6912	7086	6913	2,4555	0,0145
f60x5-03	6806	7082	6808	3,8972	0,0294
f60x5-04	5138	5299	5147	3,0383	0,1749
f60x5-05	6737	7213	6741	6,5992	0,0593
Média				2,7192	0,2999
Desv. Padrão				2,0277	0,2869

Tabela 26 – Resultados OPTUNA com intervalos de valores da literatura para PAB com ociosidade de berços.

- P. 1: fração da população que será considerada elite, limites $[0,1; 0,25]$;
- P. 2: probabilidade de um herdeiro herdar um gene do pai elite no cruzamento, limites $[0,5; 0,8]$;
- P. 3: fração da população reservada para os indivíduos mutantes, limites $[0,1; 0,3]$;
- P. 4: multiplicador da quantidade de navios que define o tamanho da população, limites $[10; 50]$;
- P. 5: tamanho da população (definido pelo Param. 4 multiplicado pela quantidade de navios), limites $[10 \times n; 50 \times n]$;
- P. 6: quantidade máxima de gerações a serem geradas (critério de parada), limites $[100; 5000]$;
- P. 7: quantidade máxima de gerações cuja variação do melhor valor objetivo entre gerações consecutivas é menor que uma tolerância (critério de parada), limites $[10; 30]$;

- P. 8: tolerância da variação do melhor valor objetivo entre uma geração e outra (associada ao Param. 7), limites $[10^{-06}; 10^{-04}]$.

Instância	P. 1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8
f30x3-01	0,203	0,659	0,195	41	1230	3900	16	0,0008
f30x3-02	0,232	0,564	0,162	40	1200	2900	23	0,0003
f30x3-03	0,165	0,571	0,119	46	1380	3800	26	0,0008
f30x3-04	0,240	0,767	0,104	41	1230	2400	22	0,0006
f30x3-05	0,238	0,679	0,170	50	1500	2100	23	0,0007
f40x5-01	0,155	0,511	0,157	50	2000	2700	27	0,0010
f40x5-02	0,212	0,632	0,154	50	2000	4400	25	0,0002
f40x5-03	0,203	0,669	0,148	49	1960	2300	21	0,0001
f40x5-04	0,196	0,705	0,162	38	1520	1100	28	0,0005
f40x5-05	0,129	0,583	0,291	43	1720	800	26	0,0002
f55x5-01	0,103	0,711	0,134	50	2750	1400	20	0,0003
f55x5-02	0,150	0,693	0,133	48	2640	2500	26	0,0002
f55x5-03	0,124	0,629	0,249	48	2640	3700	19	0,0008
f55x5-04	0,216	0,659	0,145	45	2475	2200	29	0,0005
f55x5-05	0,199	0,713	0,216	44	2420	2800	23	0,0003
f60x5-01	0,153	0,735	0,152	47	2820	1700	22	0,0010
f60x5-02	0,221	0,668	0,146	43	2580	2900	22	0,0002
f60x5-03	0,212	0,719	0,107	45	2700	900	18	0,0003
f60x5-04	0,164	0,675	0,203	50	3000	800	20	0,00003
f60x5-05	0,220	0,736	0,148	45	2700	4400	22	0,0005
Média	0,187	0,664	0,165	46		2485	22,9	0,00047
Desv.Pad.	0,041	0,066	0,046	3,773		1151,78	3,4320	0,0003

Tabela 27 – Valores dos parâmetros da melhor solução obtida pelo OPTUNA, para o PAB com ociosidade de berços.

Nas tabelas 55 e 56 estão descritas todas as configurações de parâmetro, para cada instância, que alcançaram o valor objetivo mínimo, para o modelo com ociosidade de berços. A Tabela 55 fornece os resultados para as instâncias de 30 navios e 3 berços, e a Tabela 56 para as outras instâncias. Dessas tabelas, é possível observar que, quanto menor o tamanho da instância, mais configurações distintas de parâmetros resultaram no valor objetivo mínimo da busca, e com valores consideravelmente distintos entre si, ou seja, com uma alta variação.

Analisando, agora, a Tabela 27, que apresenta um conjunto de parâmetros para cada instância, os valores de desvio padrão para esses parâmetros são, em geral, menores em comparação com a média, comparado às variações nos resultados para o modelo sem ociosidade dos berços. A quantidade de diferentes configurações de parâmetros que alcançaram o menor valor objetivo encontrado também foi menor que a quantidade nos testes para o modelo sem ociosidade de berços.

Contudo, também não podemos afirmar que um valor médio dos parâmetros poderia ser utilizado em todas as instâncias com desempenho equivalente. A variação com relação aos valores objetivos para múltiplas execuções e parâmetros fixos é muito pequena, contudo a diferença no tempo computacional é relevante, indicando que a estratégia de múltiplas execuções é melhor.

5.5 PAB sem e com ociosidade de berços modelado como PRVJT: BRKGA como *warmstart* para resolução pelo CPLEX (BRKGA + CPLEX)

Considerando o tempo computacional do BRKGA não chegando nem a 1% do tempo computacional do CPLEX para os modelos sem e com tempo de ociosidade dos berços e o fato das soluções geradas pelo BRKGA para essas instâncias serem factíveis, consideramos testar as soluções do BRKGA como solução inicial para o CPLEX.

Para definição de solução inicial para o CPLEX, não é necessário iniciar com todas as variáveis do problema definidas. Inserindo parcialmente a solução, com apenas os valores das variáveis $x_{i,j,k}$, o CPLEX conseguiu resolver o modelo. Contudo, é possível obter um *status* do CPLEX para conferir se a solução inicial é factível para o problema caso definirmos todas as variáveis $x_{i,j,k}$, $T_{i,k}$ e, para o modelo com ociosidade dos berços, $y_{i,j,k}$.

As instâncias testadas com a solução obtida pelo BRKGA como solução inicial para o CPLEX foram as descritas na Tabela 7 com os tempos limites de busca para o CPLEX de uma e duas horas. Além disso os testes foram feitos para os modelos sem e com ociosidade de berços e considerando os resultados do BRKGA com parâmetros fixos e uma única execução e os resultados através da ferramenta OPTUNA, variando o tempo limite para que, junto com a execução do CPLEX, não ultrapasse uma hora de processamento.

Para os testes com o BRKGA com parâmetros fixos, estes valores foram os mesmo que os descritos na Tabela 13, para os modelos sem e com ociosidade dos berços.

5.5.1 BAP sem ociosidade dos berços

5.5.1.1 BRKGA com parâmetros fixos, definidos manualmente

A Tabela 28 fornece os valores objetivos encontrados pelo BRKGA (Obj_{BRKGA}) e pelo método CPLEX com *warmstart*, processado com os tempos limites de uma e duas horas (Obj_{CPLEX}1h e Obj_{CPLEX}2h). Todas as soluções encontradas pelo algoritmo BRKGA foram factíveis para o problema e, assim, o CPLEX partiu de soluções factíveis. Os *status* das últimas soluções encontradas em todas as instâncias pelo CPLEX foram

solução factível, o que indica que o CPLEX não conseguiu provar a otimalidade dentro do tempo limite definido.

Além disso, as colunas *gap1*, *gap2* e *gap3* apresentam em porcentagem as diferenças entre as funções objetivo apresentadas. Os cálculos foram feitos da seguinte forma:

$$gap1 = \frac{(Obj_{CPLEX}1h - Obj_{BRKGA}) \times 100}{Obj_{BRKGA}}, \tag{5.7}$$

$$gap2 = \frac{(Obj_{CPLEX}2h - Obj_{BRKGA}) \times 100}{Obj_{BRKGA}}, \tag{5.8}$$

$$gap3 = \frac{(Obj_{CPLEX}1h - Obj_{CPLEX}2h) \times 100}{Obj_{CPLEX}1h}. \tag{5.9}$$

Ou seja, o *gap1* representa a porcentagem da diferença entre o valor objetivo da solução pelo CPLEX, com tempo limite de uma hora, e o valor objetivo da solução do BRKGA, utilizada como *warmstart*. O *gap2* é equivalente, mas para o CPLEX com o tempo limite de duas horas. Já o *gap3* apresenta a porcentagem da diferença entre os valores objetivos encontrados pelo CPLEX com uma e duas horas de tempo limite, com relação ao resultado obtido para uma hora. Ou seja, compara o desempenho entre as execuções do CPLEX com limites de uma e duas horas, com *warmstart*. Além disso, valores negativos para os *gaps* 1 e 2 significam que o CPLEX conseguiu melhorar a solução inicial fornecida pelo BRKGA.

Pela Tabela 28 é possível verificar que a melhora na função objetivo obtida pelo CPLEX com uma hora de tempo limite com relação ao BRKGA não ultrapassa 6% e, para o tempo limite de duas horas, não ultrapassa de 7%. Contudo, o tempo computacional do BRKGA não ultrapassou 40 segundos, como pode ser observado na Tabela 29 e o CPLEX chegou nos limites de tempo definidos, ou seja, o tempo de processamento BRKGA é muito menor que do CPLEX, chegando, no máximo, a 1% do tempo do CPLEX com uma hora de limite.

Além disso, a variação nos valores objetivos das soluções pelo do CPLEX de uma para duas horas é muito pouco relevante, rodando para duas horas não chegou a variar nem 3% do valor objetivo encontrado rodando por uma hora e, em nenhum dos casos, o CPLEX conseguiu provar a otimalidade das soluções encontradas. Na Figura 37 podemos observar os valores objetivos do BRKGA e CPLEX uma e duas horas em uma visualização gráfica e perceber como os valores objetivos encontrados são muito próximos.

O impacto no custo computacional supera significativamente a melhora nos valores objetivos mas é preciso entender o quando essa melhora representa em valores para o porto pois, dependendo dos custos envolvidos, 7% pode representar um montante bastante significativo monetariamente.

Instância	Obj_{BRKGA}	Obj_{CPLEX}1h	Obj_{CPLEX}2h	*gap1*	*gap2*	*gap3*
f30x3-01	1801	1786	1784	-0,833	-0,944	0,112
f30x3-02	2161	2129	2129	-1,481	-1,481	-0,000
f30x3-03	2200	2196	2198	-0,182	-0,091	-0,091
f30x3-04	1638	1567	1567	-4,335	-4,335	-0,000
f30x3-05	2164	2153	2148	-0,508	-0,739	0,232
f40x5-01	2488	2387	2333	-4,059	-6,230	2,262
f40x5-02	2921	2899	2879	-0,753	-1,438	0,690
f40x5-03	3111	2932	2906	-5,754	-6,590	0,887
f40x5-04	2143	2050	2045	-4,340	-4,573	0,244
f40x5-05	2899	2859	2850	-1,380	-1,690	0,315
f55x5-01	4777	4749	4745	-0,586	-0,670	0,084
f55x5-02	5729	5619	5610	-1,920	-2,077	0,160
f55x5-03	5916	5702	5628	-3,617	-4,868	1,298
f55x5-04	4466	4388	4392	-1,747	-1,657	-0,091
f55x5-05	5762	5638	5616	-2,152	-2,534	0,390
f60x5-01	6093	5994	5961	-1,625	-2,166	0,551
f60x5-02	7038	7006	6999	-0,455	-0,554	0,100
f60x5-03	7150	7012	7024	-1,930	-1,762	-0,171
f60x5-04	5289	5203	5212	-1,626	-1,456	-0,173
f60x5-05	6769	6767	6765	-0,030	-0,059	0,030
Média				-1,966	-2,296	0,341

Tabela 28 – Comparação valores objetivo BRKGA e CPLEX com *warmstart*, para uma e duas horas de tempo limite, PAB sem ociosidade de berços.

Na Tabela 30 estão descritos os valores da solução obtida pelo CPLEX com *warmstart* (Obj_{warm}) e sem (Obj). As soluções sem *warmstart* são as soluções apresentadas na Tabela 8. Os *gaps* indicados nas duas últimas colunas representam, em porcentagem, a diferença entre as soluções com e sem *warmstart* com relação à solução sem *warmstart*, para cada um dos tempos limites (uma e duas horas).

$$gap_{1h} = \frac{(Obj_{warm}1h - Obj1h) \times 100}{Obj1h} \tag{5.10}$$

$$gap_{2h} = \frac{(Obj_{warm}2h - Obj2h) \times 100}{Obj2h} \tag{5.11}$$

Da Tabela 30 é possível observar que, para instâncias de tamanho menor, a solução encontrada pelo CPLEX sem *warmstart* pode apresentar resultados melhores rodando tanto para uma quanto para duas horas, apesar de ser uma diferença de nem 2%. Contudo, para as instâncias maiores com 55 e 60 navios, o CPLEX com o *warmstart* apresentou soluções melhores com diferenças um pouco mais significativas, de até 11%. Ou seja, com o aumento da complexidade da instância, o impacto da solução factível encontrada pelo BRKGA como solução inicial para o CPLEX aumenta.

Instância	Tempo (s) BRKGA
f30x3-01	1,171
f30x3-02	1,429
f30x3-03	2,274
f30x3-04	1,638
f30x3-05	1,445
f40x5-01	4,663
f40x5-02	4,448
f40x5-03	3,063
f40x5-04	6,199
f40x5-05	9,180
f55x5-01	19,305
f55x5-02	9,382
f55x5-03	5,628
f55x5-04	9,160
f55x5-05	7,288
f60x5-01	10,789
f60x5-02	17,190
f60x5-03	9,468
f60x5-04	6,570
f60x5-05	36,822

Tabela 29 – Tempo de processamento BRKGA, problema sem ociosidade de berços.

5.5.1.2 BRKGA com parâmetros definidos pelo OPTUNA

Testes foram feitos considerando o melhor resultado do BRKGA rodado pelo OPTUNA como solução inicial para o CPLEX. O tempo máximo para o OPTUNA + CPLEX considerado foi de uma hora, fazendo algumas combinações de tempos entre OPTUNA e CPLEX, para tentar identificar uma melhor combinação de tempos. Em todos os testes para todas as instâncias, as soluções encontradas pelo OPTUNA foram todas factíveis para o CPLEX.

1. Rodar OPTUNA por 15 minutos e CPLEX por 45.
2. Rodar OPTUNA por 20 minutos e CPLEX por 40.
3. Rodar OPTUNA por 30 minutos e CPLEX por 30.

As Tabelas 31, 32 e 33 indicam os valores objetivos pelo OPTUNA (Obj_{OPT}), que foi utilizado como solução inicial para o CPLEX e a solução que o CPLEX encontrou dentro do tempo limite definido para cada processo (Obj_{CPLEX}), de acordo com as configurações 1, 2 e 3, respectivamente. A coluna $Obj_{CPLEX,fixo}$ indica o valor objetivo encontrado pelo CPLEX rodando por uma hora com solução inicial definida pelo BRKGA com parâmetros fixos, indicados na Tabela 28. As soluções encontradas pelo BRKGA e usadas como *warmstart* para o CPLEX foram factíveis em todos os testes.

Instância	Obj_{warm}1h	Obj_{warm}2h	Obj1h	Obj2h	gap_{1h}	gap_{2h}
f30x3-01	1786	1784	1771	1767	0,847	0,962
f30x3-02	2129	2129	2097	2116	1,526	0,614
f30x3-03	2196	2198	2213	2201	-0,768	-0,136
f30x3-04	1567	1567	1572	1556	-0,318	0,707
f30x3-05	2153	2148	2143	2139	0,467	0,421
f40x5-01	2387	2333	2377	2343	0,421	-0,427
f40x5-02	2899	2879	2998	2864	-3,302	0,524
f40x5-03	2932	2906	2921	2947	0,377	-1,391
f40x5-04	2050	2045	2240	2159	-8,482	-5,280
f40x5-05	2859	2850	3025	2893	-5,488	-1,486
f55x5-01	4749	4745	5027	4896	-5,530	-3,084
f55x5-02	5619	5610	6221	6003	-9,677	-6,547
f55x5-03	5702	5628	6019	5826	-5,267	-3,399
f55x5-04	4388	4392	4759	4768	-7,796	-7,886
f55x5-05	5638	5616	5936	5790	-5,020	-3,005
f60x5-01	5994	5961	6379	6212	-6,035	-4,041
f60x5-02	7006	6999	7872	7508	-11,001	-6,779
f60x5-03	7012	7024	7462	7241	-6,031	-2,997
f60x5-04	5203	5212	5564	5382	-6,488	-3,159
f60x5-05	6767	6765	7562	7543	-10,513	-10,314
Média					-4,4039	-2,8352

Tabela 30 – Comparação CPLEX com e sem *warmstart* para modelo sem ociosidade.

O *gap* representa a diferença relativa, em porcentagem, entre o valor objetivo encontrado pelo CPLEX com a solução inicial gerada pelo BRKGA com parâmetros fixo e o melhor valor objetivo com a solução inicial encontrada pelo OPTUNA:

$$gap = \frac{(Obj_{CPLEX,fixo} - Obj_{CPLEX}) \times 100}{Obj_{CPLEX,fixo}}. \tag{5.12}$$

Dos resultados, é possível observar que o OPTUNA + CPLEX resultou em melhores soluções do que o $Obj_{CPLEX,fixo}$ para um mesmo tempo computacional. Já o CPLEX com o *warmstart* definido pelo OPTUNA melhorou os resultados do OPTUNA para as instâncias de 40 navios e 5 berços e uma instância de 30 navios e 3 berços mas, para as instâncias maiores, não conseguiu melhorar a solução encontrada pelo OPTUNA e também não conseguiu provar a otimalidade. Além disso, comparando com os resultados sem *warmstart*, apresentados na Tabela 30, é possível perceber que os resultados obtidos pelo OPTUNA são melhores que o CPLEX sem *warmstart*.

Na Tabela 34 estão descritos os valores objetivo encontrados pelo CPLEX com a solução inicial encontrada pelo OPTUNA para cada uma das três combinações de tempos limite e, pelos resultados, não é possível definir que alguma das configurações resultou em melhor desempenho com relação a função objetivo comparado com as outras, mas a

configuração 3 apresentou mais instâncias com melhores valores objetivo com relação a configuração 1 do que a 2.

Instância	Obj_{OPT}	Obj_{CPLEX}	$Obj_{CPLEX,fixo}$	**gap**
f30x3-01	1763	1763	1786	1,288
f30x3-02	2091	2090	2129	1,832
f30x3-03	2188	2188	2196	0,364
f30x3-04	1538	1538	1567	1,851
f30x3-05	2114	2114	2153	1,811
f40x5-01	2326	2322	2387	2,723
f40x5-02	2847	2841	2899	2,001
f40x5-03	2897	2892	2932	1,364
f40x5-04	2010	2010	2050	1,951
f40x5-05	2823	2822	2859	1,294
f55x5-01	4709	4709	4749	0,842
f55x5-02	5488	5488	5619	2,331
f55x5-03	5521	5521	5702	3,174
f55x5-04	4183	4183	4388	4,672
f55x5-05	5488	5488	5638	2,661
f60x5-01	5770	5770	5994	3,737
f60x5-02	6906	6906	7006	1,427
f60x5-03	6814	6814	7012	2,824
f60x5-04	5104	5104	5203	1,903
f60x5-05	6727	6727	6767	0,591
Média				2,032

Tabela 31 – Resultados OPTUNA + CPLEX, configuração 1, PAB sem ociosidade de berços.

Instância	Obj_{OPT}	Obj_{CPLEX}	$Obj_{CPLEX,fixo}$	**gap**
f30x3-01	1765	1765	1786	1,176
f30x3-02	2091	2090	2129	1,832
f30x3-03	2188	2188	2196	0,364
f30x3-04	1538	1538	1567	1,851
f30x3-05	2114	2114	2153	1,811
f40x5-01	2325	2319	2387	2,849
f40x5-02	2837	2837	2899	2,139
f40x5-03	2886	2886	2932	1,569
f40x5-04	2011	2011	2050	1,902
f40x5-05	2844	2844	2859	0,525
f55x5-01	4711	4711	4749	0,800
f55x5-02	5489	5482	5619	2,438
f55x5-03	5513	5513	5702	3,315
f55x5-04	4202	4202	4388	4,239
f55x5-05	5488	5488	5638	2,661
f60x5-01	5775	5775	5994	3,654
f60x5-02	6894	6894	7006	1,599
f60x5-03	6800	6796	7012	3,080
f60x5-04	5124	5124	5203	1,518
f60x5-05	6739	6739	6767	0,414
Média				1,987

Tabela 32 – Resultados OPTUNA + CPLEX, configuração 2, PAB sem ociosidade de berços.

5.5.2 PAB com ociosidade dos berços

5.5.2.1 BRKGA com parâmetros fixos, definidos manualmente

Assim como para o modelo sem ociosidade, o BRKGA encontrou solução factível para o problema para todas as instâncias para o modelo com ociosidade e, assim, o *status* das soluções encontradas pelo CPLEX foram todas de solução factível, também sem conseguir provar a otimalidade. O mesmo não ocorreu com a resolução pelo CPLEX sem o *warmstart*, como descrito na Tabela 9.

A Tabela 35 apresenta os valores objetivos encontrados pelo BRKGA para instâncias do problema com ociosidade (Obj_{BRKGA}), assim como os valores objetivos encontrados pelo CPLEX com *warmstart* rodando com uma e duas horas de limite de tempo de processamento (Obj_{CPLEX}1h e Obj_{CPLEX}2h, respectivamente). Os *gaps* calculados e apresentados nas colunas *gap1*, *gap2* e *gap3* foram calculados da mesma forma que definido para o problema sem ociosidade, pelas equações (5.7), (5.8) e (5.9), respectivamente.

Para o problema com ociosidade dos berços, o CPLEX com *warmstart* chegou em valores objetivo melhores do que o BRKGA com uma diferença percentual chegando,

Instância	Obj_{OPT}	Obj_{CPLEX}	$Obj_{CPLEX,fixo}$	*gap*
f30x3-01	1763	1763	1786	1,288
f30x3-02	2091	2090	2129	1,832
f30x3-03	2188	2186	2196	0,455
f30x3-04	1538	1538	1567	1,851
f30x3-05	2114	2114	2153	1,811
f40x5-01	2321	2321	2387	2,765
f40x5-02	2836	2836	2899	2,173
f40x5-03	2888	2888	2932	1,501
f40x5-04	2007	2007	2050	2,098
f40x5-05	2834	2830	2859	1,014
f55x5-01	4705	4705	4749	0,927
f55x5-02	5488	5488	5619	2,331
f55x5-03	5517	5517	5702	3,244
f55x5-04	4187	4187	4388	4,581
f55x5-05	5488	5484	5638	2,731
f60x5-01	5765	5765	5994	3,820
f60x5-02	6895	6895	7006	1,584
f60x5-03	6804	6798	7012	3,052
f60x5-04	5096	5096	5203	2,057
f60x5-05	6723	6723	6767	0,650
Média				2,088

Tabela 33 – Resultados OPTUNA + CPLEX, configuração 3, PAB sem ociosidade de berços.

no máximo, a 5% de melhora para o tempo limite de uma hora e 6% para duas horas, como pode ser visto na Tabela 35. Para os resultados do CPLEX com *warmstart* com uma e duas horas de tempo limite, não houve uma variação muito significativa se considerarmos o aumento no tempo computacional, assim o BRKGA + CPLEX com tempo limite de uma hora pode ser mais interessante. Na Figura 38 podemos observar graficamente a diferença nos valores objetivos do BRKGA e do CPLEX com *warmstart* para cada instância testada.

Os tempos computacionais para resolver o problema com ociosidade de berços pelo BRKGA são apresentados na Tabela 36 e o CPLEX foi rodado com limites de uma e duas horas (3600 e 7200 segundos). Nos tempos de processamento podemos notar novamente o comportamento do BRKGA de chegar na solução em um tempo que equivale em torno de 1% do tempo computacional do CPLEX para uma hora de tempo limite.

A Tabela 37 apresenta os valores objetivos do problema com ociosidade de berços resolvido pelo CPLEX com *warmstart* (Obj_{warm}) e sem. Os resultados do CPLEX sem solução inicial fornecida são os apresentados na Tabela 10 e os traços indicam que o CPLEX não encontrou uma solução factível para a instância. Os *gaps* apresentados foram calculados da mesma forma que apresentado nas equações (5.10) e (5.11), representando a diferença em porcentagem entre as soluções com e sem *warmstart* em relação à solução sem.

Instância	Obj_{CPLEX} Config. 1	Obj_{CPLEX} Config. 2	Obj_{CPLEX} Config. 3
f30x3-01	1763	1765	1763
f30x3-02	2090	2090	2090
f30x3-03	2188	2188	2186
f30x3-04	1538	1538	1538
f30x3-05	2114	2114	2114
f40x5-01	2322	2319	2321
f40x5-02	2841	2837	2836
f40x5-03	2892	2886	2888
f40x5-04	2010	2011	2007
f40x5-05	2822	2844	2830
f55x5-01	4709	4711	4705
f55x5-02	5488	5482	5488
f55x5-03	5521	5513	5517
f55x5-04	4183	4202	4187
f55x5-05	5488	5488	5484
f60x5-01	5770	5775	5765
f60x5-02	6906	6894	6895
f60x5-03	6814	6796	6798
f60x5-04	5104	5124	5096
f60x5-05	6727	6739	6723

Tabela 34 – Comparação diferentes configurações de tempos OPTUNA + CPLEX, PAB sem ociosidade de berços.

As médias dos gaps foram feitas desconsiderando as instâncias para as quais o CPLEX sem solução inicial não encontrou uma solução factível.

Dos resultados dessa tabela podemos observar que, para as instâncias menores de 30 navios e 3 berços, a solução com *warmstart* não apresentou resultados melhores em todos os casos mas, com o aumento do tamanho das instâncias, as soluções com o *warmstart* apresentaram melhora significativa, até para as instâncias em que o CPLEX sem *warmstart* não conseguiu concluir nem a factibilidade da solução. A melhora obtida pelo CPLEX com *warmstart* foi mais relevante com relação ao CPLEX sem *warmstart* com relação ao modelo sem ociosidade.

5.5.2.2 BRKGA com parâmetros definidos pelo OPTUNA

Testes foram feitos considerando o melhor resultado do BRKGA com OPTUNA como solução inicial para o CPLEX também para o modelo com ociosidade de berços. O tempo máximo para o OPTUNA + CPLEX considerado foi de uma hora, fazendo algumas combinações de tempos entre OPTUNA e CPLEX, para tentar identificar uma melhor combinação de tempos. Em todos os testes para todas as instâncias, as soluções encontradas pelo OPTUNA foram todas factíveis para o CPLEX.

Instância	Obj_{BRKGA}	Obj_{CPLEX}1h	Obj_{CPLEX}2h	*gap1*	*gap2*	*gap3*
f30x3-01	1782	1768	1764	-0,786	-1,010	0,226
f30x3-02	2185	2137	2111	-2,197	-3,387	1,217
f30x3-03	2221	2206	2192	-0,675	-1,306	0,635
f30x3-04	1581	1575	1573	-0,380	-0,506	0,127
f30x3-05	2140	2128	2128	-0,561	-0,561	-0,000
f40x5-01	2554	2427	2406	-4,973	-5,795	0,865
f40x5-02	2903	2891	2887	-0,413	-0,551	0,138
f40x5-03	2956	2936	2922	-0,677	-1,150	0,477
f40x5-04	2163	2143	2114	-0,925	-2,265	1,353
f40x5-05	3022	2952	2950	-2,316	-2,383	0,068
f55x5-01	4748	4740	4740	-0,168	-0,168	0,000
f55x5-02	5853	5666	5670	-3,195	-3,127	-0,071
f55x5-03	5726	5631	5591	-1,659	-2,358	0,710
f55x5-04	4430	4385	4356	-1,016	-1,670	0,661
f55x5-05	5752	5750	5604	-0,035	-2,573	2,539
f60x5-01	5863	5849	5835	-0,239	-0,478	0,239
f60x5-02	7105	7045	7059	-0,844	-0,647	-0,199
f60x5-03	7044	6988	6980	-0,795	-0,909	0,114
f60x5-04	5255	5235	5235	-0,381	-0,381	0,000
f60x5-05	6829	6791	6791	-0,556	-0,556	-0,000
Média				-1,140	-1,589	0,455

Tabela 35 – Comparação valores objetivo BRKGA e CPLEX com *warmstart* para uma e duas horas de tempo limite, para o problema com ociosidade de berços.

1. Rodar OPTUNA por 15 minutos e CPLEX por 45.
2. Rodar OPTUNA por 20 minutos e CPLEX por 40.
3. Rodar OPTUNA por 30 minutos e CPLEX por 30.

As Tabelas 38, 39 e 40 indicam os valores de função objetivo pelo OPTUNA (Obj_{OPT}), e a solução do OPTUNA + CPLEX (Obj_{CPLEX}) respectivamente a cada configuração de tempos descritas acima. A coluna $Obj_{CPLEX,fixo}$ indica o valor objetivo encontrado pelo CPLEX rodando por uma hora com solução inicial definida pelo BRKGA com parâmetros fixos, indicados na Tabela 37. O *gap* é calculado igual ao descrito por 5.12. Todas as soluções encontradas pelo BRKGA foram factíveis e, assim, foi possível utilizá-las como *warmstart* para o CPLEX.

Assim como no caso do PAB sem ociosidade de berços, os resultados obtidos com o *warmstart* oriundo do melhor objetivo encontrado pelo OPTUNA retorna resultados melhores do que o resultado obtido pelo BRKGA com os parâmetros fixos. O valor médio de porcentagem de melhora do OPTUNA + CPLEX para o CPLEX fixo é um pouco maior para o modelo com ociosidade de berços do que para o sem, apesar dos valores serem muito próximos.

Instância	Tempo(s) BRKGA
f30x3-01	1,020
f30x3-02	0,619
f30x3-03	2,835
f30x3-04	1,836
f30x3-05	1,725
f40x5-01	3,576
f40x5-02	2,875
f40x5-03	8,642
f40x5-04	8,353
f40x5-05	6,049
f55x5-01	20,969
f55x5-02	8,712
f55x5-03	7,370
f55x5-04	11,668
f55x5-05	9,926
f60x5-01	22,597
f60x5-02	14,200
f60x5-03	14,480
f60x5-04	10,393
f60x5-05	26,622

Tabela 36 – Tempos de processamento BRKGA, problema com ociosidade.

Comparando os resultados para as diferentes configurações de tempo do OPTUNA + CPLEX (Tabela 41), não é possível definir que alguma estratégia tem melhor desempenho que outra, os resultados variam independentemente de ter um tempo limite maior para o processo do OPTUNA ou CPLEX. O *status* de solução do CPLEX para todos os testes foi de solução factível encontrada, ou seja, não conseguiu provar a otimalidade para nenhuma instância.

Instância	Obj_{warm}1h	Obj_{warm}2h	Obj1h	Obj2h	Gap_{1h}	Gap_{2h}
f30x3-01	1768	1764	1783	1801	-0,841	-2,054
f30x3-02	2137	2111	2133	2117	0,188	-0,283
f30x3-03	2206	2192	2192	2188	0,639	0,183
f30x3-04	1575	1573	1551	1561	1,547	0,769
f30x3-05	2128	2128	2174	2134	-2,116	-0,281
f40x5-01	2427	2406	2425	2403	0,082	0,125
f40x5-02	2891	2887	2955	2893	-2,166	-0,207
f40x5-03	2936	2922	3056	2960	-3,927	-1,284
f40x5-04	2143	2114	2241	2225	-4,373	-4,989
f40x5-05	2952	2950	3054	2966	-3,340	-0,539
f55x5-01	4740	4740	5401	4976	-12,238	-4,743
f55x5-02	5666	5670	7098	6320	-20,175	-10,285
f55x5-03	5631	5591	6487	5958	-13,196	-6,160
f55x5-04	4385	4356	5520	5099	-20,562	-14,571
f55x5-05	5750	5604	–	–	–	–
f60x5-01	5849	5835	8633	6834	-32,248	-14,618
f60x5-02	7045	7059	–	7880	–	-10,419
f60x5-03	6988	6980	8326	7913	-16,070	-11,791
f60x5-04	5235	5235	6770	6224	-22,674	-15,890
f60x5-05	6791	6791	–	–	–	–
Média					-8,9099	-5,3910

Tabela 37 – Comparação CPLEX com e sem *warmstart* para o modelo com ociosidade de berços.

5.6 PAB sem e com ociosidade de berços modelados como PRVJT: abordagem BRKGA com Clusterização na fase de cruzamento (BRKGA + Clusterização)

Aproveitando do conceito de clusterização de agrupar soluções com características parecidas, podemos inserir no processo de cruzamento do BRKGA para a seleção dos indivíduos que irão cruzar, com o objetivo de, mantendo a característica elitista do método, buscar cruzar indivíduos de diferentes *clusters* explorando melhor a diversificação das alocações dentro da população elite.

Contudo, inserir mais um processo no meio do algoritmo do BRKGA traz um custo computacional adicional e, dessa forma, é importante analisar qual o melhor momento de inserir o cálculo das matrizes de distância e dos *clusters* dentro do processo. Uma forma é analisar como as distâncias entre os indivíduos evoluem ao longo das gerações e identificar quando as soluções começam a convergir o suficiente para considerar agrupá-las de acordo com suas características.

Instância	Obj_{OPT}	Obj_{CPLEX}	$Obj_{CPLEX,fixo}$	*gap*
f30x3-01	1764	1764	1768	0,226
f30x3-02	2098	2097	2137	1,872
f30x3-03	2188	2188	2206	0,816
f30x3-04	1551	1551	1575	1,524
f30x3-05	2114	2114	2128	0,658
f40x5-01	2331	2326	2427	4,162
f40x5-02	2850	2850	2891	1,418
f40x5-03	2886	2886	2936	1,703
f40x5-04	2056	2056	2143	4,060
f40x5-05	2837	2837	2952	3,896
f55x5-01	4717	4717	4740	0,485
f55x5-02	5509	5509	5666	2,771
f55x5-03	5521	5521	5631	1,953
f55x5-04	4209	4207	4385	4,059
f55x5-05	5484	5484	5750	4,626
f60x5-01	5768	5768	5849	1,385
f60x5-02	6903	6903	7045	2,016
f60x5-03	6802	6800	6988	2,690
f60x5-04	5139	5139	5235	1,834
f60x5-05	6733	6733	6791	0,854
Média				2,150

Tabela 38 – Resultados OPTUNA + CPLEX, configuração 1, PAB com ociosidade de berços.

5.6.1 Análise das distâncias entre indivíduos no BRKGA

Resolvendo o modelo sem ociosidade com o BRKGA e parâmetros fixos, salvamos algumas populações de iterações intermediárias, calculamos a matriz de distância entre os indivíduos, de acordo com o descrito na Seção 5.4, e analisamos a evolução do BRKGA com relação a geração de soluções iguais ou muito próximas. As populações foram salvas em ordem decrescente de valor *fitness*, ou seja, o primeiro indivíduo é o que possui menor (pior) valor *fitness* e o último é a melhor solução. Dessa forma, as populações elite se concentraram no canto inferior direito da matriz de distância.

Para gerar as matrizes de distância dessa análise, utilizamos os seguintes parâmetros para o BRKGA:

- Multiplicador do tamanho da população: 30.
- Fração da população elite: $0,2$.
- Fração da população mutante: $0,3$.
- Probabilidade de herdar gene do pai elite: $0,7$.
- Máximo de gerações: 1000.

Instância	Obj_{OPT}	Obj_{CPLEX}	$Obj_{CPLEX,fixo}$	*gap*
f30x3-01	1764	1764	1768	0,226
f30x3-02	2098	2097	2137	1,872
f30x3-03	2188	2186	2206	0,907
f30x3-04	1551	1551	1575	1,524
f30x3-05	2114	2114	2128	0,658
f40x5-01	2322	2318	2427	4,491
f40x5-02	2846	2846	2891	1,557
f40x5-03	2892	2892	2936	1,499
f40x5-04	2051	2051	2143	4,293
f40x5-05	2838	2838	2952	3,862
f55x5-01	4711	4711	4740	0,612
f55x5-02	5508	5508	5666	2,789
f55x5-03	5523	5523	5631	1,918
f55x5-04	4202	4202	4385	4,173
f55x5-05	5484	5484	5750	4,626
f60x5-01	5778	5778	5849	1,214
f60x5-02	6922	6922	7045	1,746
f60x5-03	6810	6802	6988	2,662
f60x5-04	5155	5155	5235	1,528
f60x5-05	6723	6723	6791	1,001
Média				2,158

Tabela 39 – Resultados OPTUNA + CPLEX, configuração 2, PAB com ociosidade de berços.

- Tolerância para variação do melhor *fitness* entre duas gerações: 10^{-04}.
- Quantidade máxima de gerações com melhor *fitness* menor que a tolerância definida: 20.

Para gerar as populações para as análises, passamos a salvar as gerações para o cálculo da matriz de distância a partir da 60 e a cada 10 gerações. Na Figura 39 apresentamos o *heatmap* de algumas das matrizes de distâncias calculadas dos dados gerados pelo BRKGA e, assim, podemos observar a magnitude das distâncias através da cor, quanto maior a distância (indivíduos mais diferentes), mais clara é a cor do mapa. Como o cálculo da distância entre dois indivíduos foi definido como sendo o quantas alocações diferentes divido pelo total, os valores encontram-se no intervalo $[0, 1]$, sendo 1 representando que os indivíduos não possuem nenhuma alocação em comum e 0 representando que os indivíduos possuem todas as alocações iguais. Para cada instância, os quatro primeiros *heatmaps* representam iterações igualmente espaçadas e o quinto representa a última iteração.

Dessa figura podemos observar que, nas primeiras iterações, o BRKGA costuma apresentar uma população bastante diversa, com alocações dos navios bastante distintas

Instância	Obj_{OPT}	Obj_{CPLEX}	$Obj_{CPLEX,fixo}$	*gap*
f30x3-01	1764	1764	1768	0,226
f30x3-02	2098	2097	2137	1,872
f30x3-03	2188	2186	2206	0,907
f30x3-04	1551	1551	1575	1,524
f30x3-05	2114	2114	2128	0,658
f40x5-01	2322	2322	2427	4,326
f40x5-02	2848	2848	2891	1,487
f40x5-03	2892	2892	2936	1,499
f40x5-04	2056	2056	2143	4,060
f40x5-05	2846	2834	2952	3,997
f55x5-01	4705	4705	4740	0,738
f55x5-02	5505	5505	5666	2,842
f55x5-03	5523	5523	5631	1,918
f55x5-04	4209	4209	4385	4,014
f55x5-05	5488	5488	5750	4,557
f60x5-01	5783	5783	5849	1,128
f60x5-02	6903	6903	7045	2,016
f60x5-03	6802	6802	6988	2,662
f60x5-04	5129	5129	5235	2,025
f60x5-05	6733	6733	6791	0,854
Média				2,165

Tabela 40 – Resultados OPTUNA + CPLEX, configuração 3, PAB com ociosidade de berços.

de uma solução para outra. Contudo, conforme o BRKGA vai avançando nas iterações, a tendência da população, principalmente elite, é ter indivíduos cada vez mais próximos entre si, o que pode ser observado também pelo fato de que o critério de parada atingido para todos os testes foi a quantidade máxima de iterações com pouca variação no valor *fitness*, não atingindo o máximo de iterações definido.

Analisando, por exemplo, a Figura 39(a) para 30 navios e 3 berços que resulta em um total de 900 indivíduos por geração/iteração. Como a população elite representa 20%, são 180 indivíduos elite que são representados a partir do indivíduo 720 nos eixos do gráfico, até o final (900). Na iteração 90 podemos perceber padrões de soluções muito parecidas dentro da população elite, principalmente entre os melhores 50% indivíduos (a partir de em torno do indivíduo 810 no gráfico). Também podemos perceber, principalmente nos outros 50% dos indivíduos elite, uma similaridade considerável com alguns não elite também.

Da Figura 39(d), podemos observar que, para essa instância, a diversificação mesmo na última iteração é maior do que para as outras instâncias apresentadas, o que poderia ser um resultado de uma maior população, uma vez que o multiplicador para todas as instâncias é o mesmo, o tamanho da população aumenta com a quantidade de navios.

Instância	Obj_{CPLEX} Config. 1	Obj_{CPLEX} Config. 2	Obj_{CPLEX} Config. 3
f30x3-01	1764	1764	1764
f30x3-02	2097	2097	2097
f30x3-03	2188	2186	2186
f30x3-04	1551	1551	1551
f30x3-05	2114	2114	2114
f40x5-01	2326	2318	2322
f40x5-02	2850	2846	2848
f40x5-03	2886	2892	2892
f40x5-04	2056	2051	2056
f40x5-05	2837	2838	2834
f55x5-01	4717	4711	4705
f55x5-02	5509	5508	5505
f55x5-03	5521	5523	5523
f55x5-04	4207	4202	4209
f55x5-05	5484	5484	5488
f60x5-01	5768	5778	5783
f60x5-02	6903	6922	6903
f60x5-03	6800	6802	6802
f60x5-04	5139	5155	5129
f60x5-05	6733	6723	6733

Tabela 41 – Comparação diferentes configurações de tempos OPTUNA + CPLEX, PAB com ociosidade de berços.

Contudo, comparando as instâncias 39(a) e 39(c), apesar da segunda ter um tamanho de população maior, a diversificação na última iteração é menor que a primeira. E mesmo entre as instâncias com 60 navios e 5 berços, podemos observar na Figura 39(e) que esta instância possui uma maior intensificação que a 39(d), apesar de possuírem o mesmo tamanho de população.

Com essas análises, podemos perceber que existe a chance de cruzar indivíduos elite e não elite que são muito parecidos conforme o algoritmo avança e, assim, gerar herdeiros iguais. Além disso, conforme avançam as iterações, os próprios indivíduos elite vão se tornando muito parecidos ou até iguais aumentando a probabilidade dos pais elite selecionados para o cruzamento representarem a mesma solução.

Assim, a *clusterização* pode ser uma forma de buscar um maior variabilidade dentro da própria estratégia elitista, buscando uma variação nos herdeiros mas sem perder a característica elitista do processo de cruzamento do BRKGA.

Como no início as iterações são mais diversas, pode ser mais interessante aplicar o cálculo da matriz de distância e a clusterização após uma certa quantidade de iterações, quando a diversificação da população elite vai reduzindo mas ainda há diferentes alocações. Analisando a Figura 39(d), podemos perceber que quanto maior o tamanho da instância, maior a quantidade de iterações até começarmos a perceber um certo padrão na formação

de indivíduos mais parecidos e, verificando as médias das matrizes de distância para as instâncias geradas, o valor médio das matrizes de distância na iteração que começam a aparecer indivíduos mais parecidos é em torno de 0.86.

5.6.2 Pai elite e não elite de *clusters* diferentes

Uma estratégia adotada foi escolher o pai elite de forma aleatória dentro da população elite e na escolha do pai não elite, escolher um que não está no mesmo *cluster* com o elite que fará o par. Ou seja, para cada pai elite escolhido, identificamos o *cluster* para o qual este pertence e excluímos ele da lista de possíveis *clusters* dos quais podemos escolher o pai não elite.

Para gerar as imagens apresentadas na Figura 40, rodamos uma única vez o BRKGA para o modelo sem ociosidade de berços, com os parâmetros fixos definidos por:

- Fração da população elite: $0,2$.
- Probabilidade de herdar gene do pai elite: $0,6$.
- Fração da população mutante: $0,3$.
- Multiplicador que gera o tamanho da população: 30.
- Máximo de gerações com mudança menor que a tolerância: 20.
- Tolerância: 10^{-04}.
- Quantidade máxima de iterações: 1000.

Os parâmetros associados à *clusterização* foram definidos por:

- Iteração a partir da qual as matrizes de distância são calculadas: 60.
- Tolerância de dissimilaridade para calcular os *clusters*: $0,86$.
- Valor de dissimilaridade da altura para o corte no dendrograma para formação dos *clusters*: $0,5$.
- Tipo de ligação: média.

Além dos parâmetros do BRKGA, incluindo os de critério de parada, foi incluído alguns parâmetros associados ao passo de clusterização. Como pôde ser observado na Figura 39 e nos *heatmaps* das outras instâncias, a variação dos indivíduos da população reduz ao longo das iterações mas, observando empiricamente, a partir da iteração 60 podemos identificar uma proximidade mais relevante entre os indivíduos para as instâncias

menores. Assim, definimos um parâmetros que delimita a partir de qual iteração o método começa a calcular as matrizes de distância, cujo valor foi fixado em 60.

Como para instâncias maiores a iteração na qual é possível observar uma proximidade mais relevante entre os indivíduos, definimos uma tolerância de $0,86$ para o valor médio da matriz de distância para delimitar a iteração a partir da qual a clusterização será calculada. Ou seja, a clusterização é calculada e considerada quando a média da matriz de distância entre os indivíduos da iteração for menor que $0,86$. Essa escolha significa que a clusterização é aplicada a partir do momento em que os indivíduos tiverem uma média de similaridade de 86% das alocações.

Esses parâmetros que delimitam quando os cálculos da matriz de distância e dos *clusters* é para reduzir o tempo computacional e focar nos passos em que a *clusterização* realmente possa trazer uma maior variabilidade dentro da fase de cruzamento.

O algoritmo de *clusterização* selecionado foi o aglomerativo, que não exige a quantidade de *clusters* pré definida e possibilita escolher um nível de similaridade a partir do qual dois indivíduos não podem estar no mesmo *cluster*.

Comparando os *heatmaps* das figuras 39 e 40, é possível perceber uma maior variabilidade no processo com a *clusterização*, além de uma convergência em menos iterações. Contudo, o tempo computacional foi maior do que rodar uma única vez com os parâmetros fixos e os valores objetivos não necessariamente foram melhores. Outro ponto é que a convergência ocorreu em menos iterações, o que indica que o fim da execução aconteceu pelo critério de mudança menor que a tolerância entre diversas iterações, ou seja, apesar de uma maior diversidade entre as soluções, o *fitness* estava muito próximo.

Como já foi possível observar nos testes apresentados na Seção 6.4.2, que rodar o BRKGA com múltiplas execuções em paralelo é uma boa estratégia para obter boas soluções, os testes considerando a *clusterização* na fase de cruzamento foram feitos com 50 execuções, em paralelo, para cada instância e com os parâmetros fixos.

5.6.2.0.1 Múltiplas execuções com parâmetros fixos, modelo sem ociosidade de berços

Os parâmetros utilizados para os resultados apresentados nas Tabelas 42 e 43 foram definidos por:

- Fração da população elite: $0,2$.
- Probabilidade de herdar gene do pai elite: $0,7$.
- Fração da população mutante: $0,2$.

- Multiplicador que gera o tamanho da população: 30.
- Máximo de gerações com mudança menor que a tolerância: 20.
- Tolerância: 10^{-04}.
- Quantidade máxima de iterações: 1000.

Os parâmetros associados à *clusterização* foram definidos por:

- Iteração a partir da qual as matrizes de distância são calculadas: 55.
- Tolerância de dissimilaridade para calcular os *clusters*: $0,86$.
- Valor de dissimilaridade da altura para o corte no dendrograma para formação dos *clusters*: $0,5$.
- Tipo de ligação: média.

Na Tabela 42 podemos observar os resultados gerais, com a comparação dos valores objetivo mínimos e máximos, o tempo total de rodar as múltiplas vezes em paralelo e as médias e desvios padrão entre os valores de função objetivo encontrados nas 50 execuções para cada instância. Comparando com os valores mínimos obtidos rodando o BRKGA múltiplas vezes sem clusterização não há padrão para podermos afirmar que com a clusterização apresenta-se como uma estratégia garantida de melhores valores de função objetivo, apesar de obter resultados melhores em mais da metade das instâncias. Com relação a uma única execução, a estratégia com clusterização garantiu melhores valores de função objetivo para todas as instâncias, com um impacto maior do que com as múltiplas execuções.

Com relação ao tempo computacional, as múltiplas execuções com clusterização superam ambas as estratégias com parâmetros fixos (única e múltiplas execuções sem clusterização), mas ainda fica bem abaixo do tempo computacional da estratégia do BRKGA com OPTUNA.

Já na Tabela 43, temos resultados específicos relacionados ao menor valor de função objetivo encontrado. A coluna clusterizações indica a quantidade de iterações que tiveram o processo de *clusterização* executado, uma vez que não é feito em todas as iterações, e o total de iterações que o algoritmo levou para convergir no melhor resultado.

Em algumas das múltiplas execuções, houve casos de execução em que nenhuma clusterização foi considerada no processo, indicando que as matrizes de distância de todas as iterações tinham um valor médio mais alto, ou seja, uma diversificação maior. Esses casos não resultaram nos melhores valores objetivos.

Instância	Obj. min	Obj. max	Dif (%)	Média	Desv. Pad.	Tempo (min)
f30x3-01	1767	1819	2,9428	1789,46	12,7427	0,6251
f30x3-02	2096	2174	3,7214	2124,34	18,2797	0,7289
f30x3-03	2194	2256	2,8259	2213,92	14,7908	0,6998
f30x3-04	1540	1613	4,7403	1577,70	15,6208	0,7633
f30x3-05	2118	2279	7,6015	2151,96	27,9744	0,5996
f40x5-01	2348	2476	5,4514	2380,80	29,5863	1,8280
f40x5-02	2846	2988	4,9895	2893,46	28,6201	2,0619
f40x5-03	2895	3132	8,1865	2938,94	32,4593	1,8469
f40x5-04	2022	2181	7,8635	2086,84	32,7628	1,9083
f40x5-05	2862	3083	7,7219	2934,68	53,3741	1,9906
f55x5-01	4719	4902	3,8779	4756,80	34,0720	6,6545
f55x5-02	5496	5674	3,2387	5557,46	35,3179	6,6464
f55x5-03	5523	5783	4,7076	5617,52	70,8442	6,0721
f55x5-04	4215	4349	3,1791	4279,68	36,3955	6,4483
f55x5-05	5500	5848	6,3273	5578,98	75,1938	6,5738
f60x5-01	5773	6013	4,1573	5825,32	38,4200	9,9254
f60x5-02	6900	7157	3,7246	6991,88	54,9239	9,1910
f60x5-03	6810	7159	5,1248	6891,58	78,8095	8,8084
f60x5-04	5137	5553	8,0981	5232,36	85,4170	8,3324
f60x5-05	6733	6923	2,8219	6796,00	40,4389	9,5728

Tabela 42 – Resultados BRKGA + Clusterização entre pai elite e não elite, modelo sem ociosidade de berço.

Instância	Obj. min	Tempo (min)	Clusterizações	Total gerações
f30x3-01	1767	0,2483	60	115
f30x3-02	2096	0,3175	46	106
f30x3-03	2194	0,5107	89	159
f30x3-04	1540	0,5410	98	160
f30x3-05	2118	0,2298	48	103
f40x5-01	2348	1,1971	63	176
f40x5-02	2846	2,0605	183	304
f40x5-03	2895	1,5840	105	236
f40x5-04	2022	1,2836	66	180
f40x5-05	2862	0,8748	22	138
f55x5-01	4719	3,2897	97	296
f55x5-02	5496	4,6326	88	259
f55x5-03	5523	5,6147	148	344
f55x5-04	4215	4,3396	41	236
f55x5-05	5500	4,7696	67	250
f60x5-01	5773	7,5511	78	286
f60x5-02	6900	6,4587	66	252
f60x5-03	6810	6,8398	94	321
f60x5-04	5137	5,9666	51	225
f60x5-05	6733	3,8762	67	260

Tabela 43 – Melhor solução BRKGA + Clusterização entre pai elite e não elite, modelo sem ociosidade de berço.

Na Tabela 44 comparamos o melhor resultado obtido pelo BRKGA + Clusterização (CLUST) com os apresentados na Tabela 20, que inclui valores de soluções rodando o BRKGA uma única vez (BRKGA1) e múltiplas vezes (BRKGA2) com parâmetros fixos e utilizando da ferramenta OPTUNA (OPTN).

Os *gaps* 1, 2 e 3 representam a porcentagem da diferença da solução encontrada pelo BRKGA + Clusterização, com as soluções BRKGA1, BRKGA2 e OPTN, respectivamente. Os valores negativos representam os casos em que o BRKGA + Clusterização obteve menor valor de função objetivo. Assim, podemos notar que a solução encontrada com a clusterização foi melhor que rodar o BRKGA uma única vez com os parâmetros fixos, mas com relação às múltiplas execuções do BRKGA com parâmetros fixos, os resultados variam com um pouco mais da metade dos casos em que o BRKGA + Clusterização encontrou um melhor valor objetivo, mas com uma diferença de menos de 1%. Já com relação aos resultados obtidos pelo OPTUNA, o BRKGA + Clusterização obteve valores objetivos mais altos em todos os casos, apesar de ser com uma diferença de no máximo $1,2\%$.

Considerando não apenas os valores objetivos mas também os tempos computacionais, o OPTUNA é o que consumiu mais tempo que todas as outras estratégias para encontrar a solução, mas encontrando soluções melhores na maioria dos casos, enquanto o BRKGA com única execução foi o que consumiu menos tempo mas resultando em piores soluções. Já o BRKGA + Clusterização levou tempos parecidos para as instâncias de tamanho menor e mais tempo nas maiores, com relação às múltiplas execuções sem clusterização, mas obtendo resultados um pouco melhores.

5.6.2.0.2 Múltiplas execuções com parâmetros fixos obtidos pelo OPTUNA, modelo sem ociosidade de berços

Testamos a estratégia de múltiplas execuções com clusterização selecionando uma configuração de parâmetros que resultou no melhor valor de função objetivo pelo OPTUNA para cada instância (das configurações descritas nas Tabelas 53 e 54). A Tabela 45 apresenta os resultados gerais de utilizar os parâmetros obtidos pelo OPTUNA que resultaram em melhores valores de função objetivo para as múltiplas execuções do BRKGA com clusterização e, na Tabela 46 podemos observar a comparação do resultado do BRKGA com clusterização e parâmetros do OPTUNA (CLUST_{OPT}), com os resultados do BRKGA com múltiplas execuções e parâmetros fixos definidos empiricamente com clusterização (CLUST) e sem clusterização (BRKGA2), apresentados em 43 e 17, respectivamente, e com os resultados obtidos pelo BRKGA + OPTUNA, apresentados em 19.

Instância	CLUST	BRKGA1	BRKGA2	OPTN	*gap1*	*gap2*	*gap3*
f30x3-01	1767	1779	1767	1763	-0,6745	0,0000	0,2269
f30x3-02	2096	2132	2094	2091	-1,6886	0,0955	0,2391
f30x3-03	2194	2222	2197	2188	-1,2601	-0,1365	0,2742
f30x3-04	1540	1615	1550	1538	-4,6440	-0,6452	0,1300
f30x3-05	2118	2124	2120	2114	-0,2825	-0,0943	0,1892
f40x5-01	2348	2384	2335	2321	-1,5101	0,5567	1,1633
f40x5-02	2846	2949	2853	2836	-3,4927	-0,2454	0,3526
f40x5-03	2895	2935	2894	2888	-1,3629	0,0346	0,2424
f40x5-04	2022	2074	2024	2013	-2,5072	-0,0988	0,4471
f40x5-05	2862	2938	2846	2832	-2,5868	0,5622	1,0593
f55x5-01	4719	4963	4705	4711	-4,9164	0,2976	0,1698
f55x5-02	5496	5654	5498	5492	-2,7945	-0,0364	0,0728
f55x5-03	5523	5627	5529	5511	-1,8482	-0,1085	0,2177
f55x5-04	4215	4502	4225	4203	-6,3749	-0,2367	0,2855
f55x5-05	5500	5536	5498	5488	-0,6503	0,0364	0,2187
f60x5-01	5773	5919	5780	5771	-2,4666	-0,1211	0,0347
f60x5-02	6900	7002	6898	6896	-1,4567	0,0290	0,0580
f60x5-03	6810	7004	6814	6798	-2,7698	-0,0587	0,1765
f60x5-04	5137	5492	5146	5130	-6,4639	-0,1749	0,1365
f60x5-05	6733	7023	6737	6731	-4,1293	-0,0594	0,0297
Média					-2,6940	-0,0202	0,2862
Desv.Pad.					1,8046	0,2670	0,3011

Tabela 44 – Comparação BRKGA + Clusterização entre pai elite e não elite com outros testes, modelo sem ociosidade de berço.

Dos resultados apresentados podemos observar que utilizar os parâmetros do OPTUNA que retornaram os melhores valores de função objetivo melhora mais da metade dos resultados da clusterização com parâmetros definidos empiricamente e também com relação às múltiplas execuções sem clusterização. Com relação ao próprio OPTUNA essa estratégia BRKGA + OPTUNA + Clusterização melhorou alguns valores de função objetivo. Os tempos computacionais foram maiores, o que pode indicar que foi possível aumentar a diversificação entre os indivíduos das populações na convergência, levando mais tempo para chegar ao ponto de não conseguir melhorar mais o valor de função objetivo de acordo com a tolerância definida. Como utilizamos os parâmetros obtidos pelo OPTUNA, o tempo total teria que considerar a soma dos tempos computacionais do OPTUNA com o BRKGA e Clusterização, resultando em tempos mais altos chegando, até, ao tempo resolvendo pelo CPLEX com uma hora de limite de tempo (para a instância f60x5-04).

5.6.2.0.3 Múltiplas execuções com parâmetros fixos, modelo com ociosidade de berços

Instância	Obj. min	Obj. max	Dif (%)	Média	Desv. Pad.	Tempo (min)
f30x3-01	1767	1808	2,3203	1780,58	11,7266	2,0955
f30x3-02	2093	2138	2,1500	2108,12	11,5770	1,5096
f30x3-03	2190	2227	1,6895	2202,40	8,7458	1,6846
f30x3-04	1546	1668	7,8913	1576,72	29,1450	1,1359
f30x3-05	2118	2177	2,7856	2141,56	13,3312	1,3158
f40x5-01	2333	2405	3,0862	2368,46	15,7249	2,4433
f40x5-02	2844	2927	2,9184	2875,84	20,0299	4,3395
f40x5-03	2892	3132	8,2988	2931,06	41,0691	5,5346
f40x5-04	2020	2256	11,6832	2057,32	37,8366	7,2515
f40x5-05	2859	3219	12,5918	2974,34	94,8973	2,9284
f55x5-01	4705	4789	1,7853	4736,92	15,1951	12,7635
f55x5-02	5492	5678	3,3867	5537,96	35,0078	19,9497
f55x5-03	5529	5744	3,8886	5577,72	51,0686	12,5183
f55x5-04	4189	4332	3,4137	4258,42	28,8749	19,2161
f55x5-05	5490	5809	5,8106	5574,92	89,5314	22,6048
f60x5-01	5776	5933	2,7181	5811,80	25,9859	13,5807
f60x5-02	6904	7392	7,0684	7023,24	106,9651	18,2080
f60x5-03	6798	7143	5,0750	6975,78	76,8599	21,6439
f60x5-04	5126	5415	5,6379	5215,90	67,0234	32,5132
f60x5-05	6737	6940	3,0132	6792,38	45,6790	17,0140

Tabela 45 – Resultados BRKGA + Clusterização com parâmetros do OPTUNA, modelo sem ociosidade de berço

Da mesma forma que feito para o modelo sem ociosidade dos berços, fizemos testes para o modelo com a ociosidade. Os parâmetros tanto do BRKGA quanto da clusterização foram os mesmos, assim como a quantidade de execuções em paralelo para cada instância (50).

As tabelas 47 e 48 trazem, também, os resultados gerais, com parando a melhor e pior solução encontrada dentro das 50 execuções, assim como a média e desvio padrão dos valores objetivos encontrados e o tempo total para rodar as 50 execuções em paralelo e os resultados específicos para a melhor solução encontrada para cada instância.

Assim como no caso do modelo sem ociosidade de berços, comparando a Tabela 47 com os resultados das múltiplas execuções sem clusterização, apresentados na Tabela 18, não podemos afirmar que com a clusterização os resultados são melhores, seja com relação aos valores objetivos mínimo e máximo encontrados ou com relação à média e desvio padrão. Além disso, para o modelo com ociosidade de berços o BRKGA + Clusterização alcançou melhores valores de função objetivo para menos da metade das instâncias, diferentemente do caso sem ociosidade de berços, o que pode ser observado pelo média dos *gap2* dos dois casos. Com relação ao tempo computacional também podemos observar o mesmo comportamento para o caso do modelo sem ociosidade de berços.

Da mesma forma que para os testes com o modelo sem ociosidade, comparamos

Instância	CLUST_{OPT}	CLUST	BRKGA2	OPT	*gap*1	*gap*2	*gap*3
f30x3-01	1767	1767	1767	1763	0,000	0,000	0,227
f30x3-02	2093	2096	2094	2091	-0,143	-0,048	0,096
f30x3-03	2190	2194	2197	2188	-0,182	-0,319	0,091
f30x3-04	1546	1540	1550	1538	0,390	-0,258	0,520
f30x3-05	2118	2118	2120	2114	0,000	-0,094	0,189
f40x5-01	2333	2348	2335	2321	-0,639	-0,086	0,517
f40x5-02	2844	2846	2853	2836	-0,070	-0,316	0,282
f40x5-03	2892	2895	2894	2888	-0,104	-0,069	0,139
f40x5-04	2020	2022	2024	2013	-0,099	-0,198	0,348
f40x5-05	2859	2862	2846	2832	-0,105	0,457	0,953
f55x5-01	4705	4719	4705	4711	-0,297	0,000	-0,127
f55x5-02	5492	5496	5498	5492	-0,073	-0,109	0,000
f55x5-03	5529	5523	5529	5511	0,109	0,000	0,327
f55x5-04	4189	4215	4225	4203	-0,617	-0,852	-0,333
f55x5-05	5490	5500	5498	5488	-0,182	-0,146	0,036
f60x5-01	5776	5773	5780	5771	0,052	-0,069	0,087
f60x5-02	6904	6900	6898	6896	0,058	0,087	0,116
f60x5-03	6798	6810	6814	6798	-0,176	-0,235	0,000
f60x5-04	5126	5137	5146	5130	-0,214	-0,389	-0,078
f60x5-05	6737	6733	6737	6731	0,059	0,000	0,089
Média					-0,112	-0,132	0,174
Desv.Pad.					0,230	0,248	0,275

Tabela 46 – Comparação BRKGA + Clusterização + OPTUNA com outros resultados, modelo sem ociosidade de berço.

com os dados de outros testes apresentados na Tabela 26 e descritos junto com o resultado do BRKGA + Clusterização na Tabela 49, na qual as colunas BRKGA1 e BRKGA2 representam os resultados de rodar o BRKGA com parâmetros fixos uma e múltiplas vezes, respectivamente. A coluna OPTN traz o melhor valor objetivo encontrado pela ferramenta OPTUNA e a CLUST representa o melhor resultado com a estratégia BRKGA + Clusterização.

As comparações com as outras estratégias de aplicação do BRKGA para o modelo com ociosidade segue na mesma linha que para o modelo sem ociosidade, tanto com relação ao melhor valor objetivo encontrado quanto com relação aos tempos de computacionais das estratégias. Mas, para esses testes, o BRKGA + Clusterização apresentou melhor desempenho do que as múltiplas execuções sem a clusterização para menos instâncias, apenas 8 das 20. Além disso, a estratégia BRKGA + Clusterização encontrou algumas soluções igual à obtida pelo OPTUNA, para instâncias de 30 navios e 3 berços, contudo a diferença nas soluções com maior objetivo é maior do que no caso sem ociosidade.

Instância	Obj. min	Obj. max	Dif (%)	Média	Desv. Pad.	Tempo (min)
f30x3-01	1766	1825	3,3409	1790,02	16,0325	0,6284
f30x3-02	2098	2206	5,1478	2133,46	22,7077	0,7212
f30x3-03	2188	2256	3,1079	2211,78	13,8833	0,6539
f30x3-04	1563	1663	6,3980	1592,26	21,9580	0,7459
f30x3-05	2114	2199	4,0208	2151,38	18,6262	0,5794
f40x5-01	2333	2426	3,9863	2375,56	19,1665	2,4047
f40x5-02	2857	2977	4,2002	2902,70	26,8627	1,8801
f40x5-03	2908	3214	10,5227	2952,54	48,2672	1,8841
f40x5-04	2071	2231	7,7257	2128,94	31,5630	2,0292
f40x5-05	2855	3209	12,3993	2931,86	55,8953	1,7385
f55x5-01	4719	4905	3,9415	4767,06	35,3429	6,6619
f55x5-02	5518	5695	3,2077	5580,14	34,0414	6,5821
f55x5-03	5537	5769	4,1900	5599,90	63,1468	6,2275
f55x5-04	4234	4373	3,2829	4289,82	30,9448	6,5757
f55x5-05	5508	5718	3,8126	5565,08	53,7537	6,9734
f60x5-01	5783	6013	3,9772	5837,22	48,9837	9,7135
f60x5-02	6936	7528	8,5352	7032,20	114,6003	8,2126
f60x5-03	6808	7124	4,6416	6894,28	77,0539	8,5526
f60x5-04	5165	5350	3,5818	5229,20	36,1527	9,0443
f60x5-05	6735	6937	2,9993	6797,68	47,3495	8,4550

Tabela 47 – Resultados BRKGA + Clusterização entre pai elite e não elite, modelo com ociosidade de berço.

5.6.2.0.4 Múltiplas execuções com parâmetros fixos obtidos pelo OPTUNA, modelo com ociosidade de berços

Selecionando um conjunto de parâmetros que resultou melhores valores de função objetivo pelo OPTUNA com clusterização e múltiplas execuções, testamos também para o problema com ociosidade de berços. Na Tabela 50 os resultados de valores mínimo e máximo da função objetivo dentre todas as 50 execuções, assim como a diferença entre esses valores e a média e desvio padrão de todos os resultados para cada instância e os valores totais de tempo para rodar as 50 execuções em paralelo também são descritos nessa tabela.

Na Tabela 51 estão descritos os resultados da melhor solução de cada instância, com o valor da função objetivo, o tempo para obter esse valor, a quantidade de clusterizações calculadas e o total de gerações. Já na Tabela 52 comparamos os resultados dos melhores valores objetivos das múltiplas execuções do BRKGA + Clusterização utilizando os parâmetros encontrados pelo OPTUNA com os melhores valores do BRKGA + Clusterização e parâmetros definidos manualmente (coluna CLUST), com o BRKGA apenas

Instância	Obj. min	Tempo(min)	Clusterizações	Total gerações
f30x3-01	1766	0,5406	94	167
f30x3-02	2098	0,4092	76	135
f30x3-03	2188	0,5936	114	181
f30x3-04	1563	0,2549	40	107
f30x3-05	2114	0,3900	60	126
f40x5-01	2333	1,6647	160	302
f40x5-02	2857	1,2371	63	175
f40x5-03	2908	1,8200	120	268
f40x5-04	2071	0,8391	26	133
f40x5-05	2855	1,0045	44	148
f55x5-01	4719	4,7623	75	263
f55x5-02	5518	5,0341	96	258
f55x5-03	5537	4,3853	57	219
f55x5-04	4234	3,8371	21	206
f55x5-05	5508	5,1219	98	282
f60x5-01	5783	6,7800	103	334
f60x5-02	6936	3,7849	20	210
f60x5-03	6808	7,8442	126	347
f60x5-04	5165	6,6338	45	249
f60x5-05	6735	6,3274	54	244

Tabela 48 – Melhor solução BRKGA + Clusterização entre pai elite e não elite, modelo com ociosidade de berço.

com múltiplas execuções (coluna BRKGA2) e com o BRKGA com a busca por parâmetros pelo OPTUNA (coluna OPTN).

Os tempos computacionais em média não variaram com relação ao BRKGA + Clusterização com os parâmetros definidos manualmente e, então, são maiores que as múltiplas execuções sem clusterização e menores que os tempos com a busca de hiper parâmetros pelo OPTUNA. Contudo, o tempo apresentado na Tabela 50 não está somando o tempo do OPTUNA mas utilizou os parâmetros resultantes da busca e, assim, se considerarmos o tempo do OPTUNA, acrescentaríamos em torno de 30 minutos no tempo total. As médias dos valores de função objetivo para cada instância utilizando os parâmetros do OPTUNA foram, em geral, menores que as médias considerando os parâmetros definidos manualmente.

Com relação a qualidade dos valores de função objetivo em comparação com outros testes, com os parâmetros obtidos pelo OPTUNA os resultados foram melhores para as instâncias a partir de 40 navios e 5 berços com relação ao BRKGA + Clusterização e os parâmetros definidos manualmente. Comparando com as múltiplas execuções e parâmetros fixos, tiveram mais instâncias que a clusterização com os parâmetros pelo OPTUNA foi melhor, mas a média das melhoras foi menor do que a média obtida pelo BRKGA + Clusterização com os parâmetros definidos empiricamente. Com relação aos

Instância	CLUST	BRKGA1	BRKGA2	OPTN	*gap*1	*gap*2	*gap*3
f30x3-01	1766	1770	1766	1764	-0,2260	0,0000	0,1134
f30x3-02	2098	2105	2101	2098	-0,3325	-0,1428	0,0000
f30x3-03	2188	2209	2194	2188	-0,9507	-0,2735	0,0000
f30x3-04	1563	1578	1553	1551	-0,9506	0,6439	0,7737
f30x3-05	2114	2140	2124	2114	-1,2150	-0,4708	0,0000
f40x5-01	2333	2412	2328	2318	-3,2753	0,2148	0,6471
f40x5-02	2857	2871	2862	2849	-0,4876	-0,1747	0,2808
f40x5-03	2908	2939	2904	2892	-1,0548	0,1377	0,5533
f40x5-04	2071	2151	2076	2058	-3,7192	-0,2408	0,6317
f40x5-05	2855	2928	2870	2846	-2,4932	-0,5226	0,3162
f55x5-01	4719	4809	4717	4709	-1,8715	0,0424	0,2124
f55x5-02	5518	5679	5507	5506	-2,8350	0,1997	0,2179
f55x5-03	5537	5704	5519	5515	-2,9278	0,3261	0,3989
f55x5-04	4234	4565	4229	4190	-7,2508	0,1182	1,0501
f55x5-05	5508	5536	5502	5490	-0,5058	0,1091	0,3279
f60x5-01	5783	5952	5786	5775	-2,8394	-0,0518	0,1385
f60x5-02	6936	7086	6913	6912	-2,1169	0,3327	0,3472
f60x5-03	6808	7082	6808	6806	-3,8690	0,0000	0,0294
f60x5-04	5165	5299	5147	5138	-2,5288	0,3497	0,5255
f60x5-05	6735	7213	6741	6737	-6,6269	-0,0890	-0,0297
Média					-2,4038	0,0254	0,3267
Desv. Pad.					1,9299	0,2868	0,2955

Tabela 49 – Comparação BRKGA + Clusterização com outros testes, modelo com ociosidade de berço

melhores valores de função objetivo obtidos pela busca do OPTUNA, a clusterização com os parâmetros do OPTUNA obteve resultados melhores para a maior parte das instâncias de 60 navios e 5 berços, mas para as outras instâncias variou, obtendo valores maiores para as de 30 navios e 3 berços.

5.7 Resumo dos testes computacionais

No presente trabalho, evoluímos a implementação feita do modelo do PAB como um PRVJT apresentada no trabalho Peteam (2016), e apresentamos, também, uma adaptação no modelo para considerar a ociosidade dos berços e não apenas o tempo total de permanência dos navios no porto, para buscar uma visão de interesse também do porto, inserindo mais variáveis e restrições, aumentando a complexidade do modelo. A linguagem de programação escolhida para o desenvolvimento das aplicações foi *Python*, por ser uma linguagem que provê bibliotecas que auxiliam em um desenvolvimento rápido, principalmente na área de aprendizagem de máquinas, com bibliotecas como *Sklearn* utilizada para a clusterização e o *Numpy*, utilizada para a implementação vetorial do BRKGA.

Instância	Obj. min	Obj. max	Dif (%)	Média	Desv. Pad.	Tempo (min)
f30x3-01	1766	1880	6,4553	1789,40	22,1101	1,0922
f30x3-02	2100	2184	4,0000	2122,64	15,9639	1,4360
f30x3-03	2188	2236	2,1938	2203,84	10,9421	1,8460
f30x3-04	1553	1643	5,7952	1585,68	17,9460	0,9123
f30x3-05	2116	2182	3,1191	2137,60	13,2419	2,1021
f40x5-01	2327	2626	12,8492	2422,52	64,8462	5,2286
f40x5-02	2848	3020	6,0393	2883,54	35,7977	7,6664
f40x5-03	2896	3163	9,2196	2951,60	57,3044	4,6691
f40x5-04	2066	2186	5,8083	2116,00	30,5113	3,1792
f40x5-05	2862	3072	7,3375	2928,48	51,8716	4,5136
f55x5-01	4705	4780	1,5940	4740,48	15,3546	12,3078
f55x5-02	5505	5615	1,9982	5551,60	27,9467	14,9036
f55x5-03	5525	5867	6,1900	5640,52	81,1432	14,2342
f55x5-04	4202	4430	5,4260	4265,28	38,1955	19,4559
f55x5-05	5492	5858	6,6642	5565,40	80,2511	17,8332
f60x5-01	5777	5863	1,4887	5811,82	17,3977	19,4271
f60x5-02	6907	7355	6,4862	7020,68	93,9541	22,5381
f60x5-03	6800	7086	4,2059	6865,00	48,5794	20,1243
f60x5-04	5127	5490	7,0802	5225,72	77,7025	24,7856
f60x5-05	6721	6887	2,4699	6776,12	29,6941	26,0453

Tabela 50 – Resultados BRKGA + Clusterização com parâmetros do OPTUNA, modelo com ociosidade de berço

Para a resolução desses modelos, foi proposta uma decodificação para o BRKGA que determina de forma independente a ordem e o berço de alocação dos navios. Uma primeira implementação mais simples foi feita e testada com instâncias geradas de maneira a favorecer a factibilidade do problema e, dessa forma, tornando-as mais fáceis de serem resolvidas. Para esses testes, a implementação mais simples do BRKGA sem nenhuma estratégia adicional não se tornou competitiva em questão de tempo computacional com relação à resolução pelo CPLEX, que encontrava soluções ótimas tão rápido quanto o BRKGA.

Diante disso, o desempenho dessa implementação foi analisada e pudemos concluir que o passo mais caro computacionalmente era o cálculo do valor *fitness* de cada indivíduo. Com isso, buscou-se uma implementação mais eficiente que fizesse os cálculos do valor *fitness* de forma vetorizada para todos os indivíduos, calculando ao mesmo tempo mas sem a necessidade de paralelizar essa fase. Para isso, utilizamos a biblioteca *Numpy* para aumentar a dimensão do tamanho da população, passando a lidar com uma dimensão do tamanho da população para representar os indivíduos, além das dimensões de navios e berços. Por ser uma implementação vetorizada, impossibilita a paralelização das operações e cálculos dos processos do BRKGA.

Com essa implementação conseguimos melhorar consideravelmente o tempo

Instância	Obj. min	Tempo min.(min)	Clusterizações	Total gerações
f30x3-01	1766	0,7838	57	150
f30x3-02	2100	0,6010	11	127
f30x3-03	2188	0,9973	99	169
f30x3-04	1553	0,3967	36	91
f30x3-05	2116	1,3927	59	158
f40x5-01	2327	4,2648	9	270
f40x5-02	2848	5,4311	82	264
f40x5-03	2896	3,8220	56	228
f40x5-04	2066	1,4207	53	136
f40x5-05	2862	2,9385	0	206
f55x5-01	4705	7,6935	142	214
f55x5-02	5505	10,4631	86	201
f55x5-03	5525	11,8738	10	306
f55x5-04	4202	14,7126	103	286
f55x5-05	5492	10,7060	42	233
f60x5-01	5777	13,0028	82	193
f60x5-02	6907	14,2586	64	335
f60x5-03	6800	18,1921	142	326
f60x5-04	5127	20,9601	20	277
f60x5-05	6721	17,6076	112	267

Tabela 51 – Melhores soluções BRKGA + OPTUNA + Clusterização, modelo com ociosidade de berço

computacional para encontrar boas soluções, se tornando bastante competitivo com relação à resolução pelo CPLEX. Para a implementação e resolução do modelo pelo CPLEX com a linguagem *Python*, utilizamos a biblioteca DOCPLEX. Pela dificuldade em resolver de forma exata o modelo nos testes feitos, foi necessário limitar o tempo computacional. Uma análise do tempo computacional foi feita comparando os limites de uma e duas horas de execução e, para o modelo sem ociosidade dos berços, o tempo limite de uma hora apresentou-se suficiente no equilíbrio de tempo com qualidade da solução, comparado a duas horas. Já para o modelo com ociosidade de berços, devido ao aumento da complexidade do modelo, o tempo limite de duas horas apresentou um melhor equilíbrio de tempo e solução, uma vez que uma hora resultou em mais instâncias para as quais o CPLEX nem encontrou uma solução factível.

Rodando o BRKGA uma única vez com os valores dos parâmetros fixos, obtivemos tempos computacionais que não passaram de 24 segundos para o modelo sem ociosidade de berços, comparado com uma hora do CPLEX e obteve soluções melhores para todas as instâncias maiores, de 55 e 60 navios e 5 berços. Para as instâncias menores, de 30 navios e 3 berços, o BRKGA obteve um melhor valor objetivo apenas para uma delas mas com uma diferença que não chegou a 3% do valor do CPLEX, e com um tempo computacional muito menor. Para o modelo com ociosidade de berços, comparamos com

Instância	CLUST_{OPT}	CLUST	BRKGA2	OPTN	*gap*1	*gap*2	*gap*3
30x3-01	1766	1766	1766	1764	0,0000	0,0000	0,1134
30x3-02	2100	2098	2101	2098	0,0953	-0,0476	0,0953
30x3-03	2188	2188	2194	2188	0,0000	-0,2735	0,0000
30x3-04	1553	1563	1553	1551	-0,6398	0,0000	0,1289
30x3-05	2116	2114	2124	2114	0,0946	-0,3766	0,0946
40x5-01	2327	2333	2328	2318	-0,2572	-0,0430	0,3883
40x5-02	2848	2857	2862	2849	-0,3150	-0,4892	-0,0351
40x5-03	2896	2908	2904	2892	-0,4127	-0,2755	0,1383
40x5-04	2066	2071	2076	2058	-0,2414	-0,4817	0,3887
40x5-05	2862	2855	2870	2846	0,2452	-0,2787	0,5622
55x5-01	4705	4719	4717	4709	-0,2967	-0,2544	-0,0849
55x5-02	5505	5518	5507	5506	-0,2356	-0,0363	-0,0182
55x5-03	5525	5537	5519	5515	-0,2167	0,1087	0,1813
55x5-04	4202	4234	4229	4190	-0,7558	-0,6384	0,2864
55x5-05	5492	5508	5502	5490	-0,2905	-0,1818	0,0364
60x5-01	5777	5783	5786	5775	-0,1038	-0,1555	0,0346
60x5-02	6907	6936	6913	6912	-0,4181	-0,0868	-0,0723
60x5-03	6800	6808	6808	6806	-0,1175	-0,1175	-0,0882
60x5-04	5127	5165	5147	5138	-0,7357	-0,3886	-0,2141
60x5-05	6721	6735	6741	6737	-0,2079	-0,2967	-0,2375
Média					-0,2405	-0,2157	0,0849
Des.Pad.					0,2670	0,1945	0,2029

Tabela 52 – Resultados BRKGA + Clusterização comparâmetros do OPTUNA, modelo com ociosidade de berço

o CPLEX rodando para duas horas e o BRKGA obteve resultados melhores, com uma melhora maior do que para o problema sem ociosidade, principalmente considerando o fato de encontrar soluções factíveis para as instâncias que o CPLEX não conseguiu e com uma diferença de tempo computacional muito maior.

Entre os modelos sem e com ociosidade, o tempo computacional do BRKGA com única execução e parâmetros fixos definidos empiricamente não diferiu muito mesmo com o aumento das dimensões nos cálculos por conta das variáveis a mais para o modelo com ociosidade. Além disso, o BRKGA encontrou soluções factíveis em todos os testes realizados para ambos os modelos, uma característica de tratar os processos de algoritmos genéticos com as soluções decodificadas.

O BRKGA não é um algoritmo determinístico, ou seja, se rodar ele com os mesmos parâmetros de entrada ou até partir mesma solução inicial, ele não garante retornar a mesma solução final. Além disso, é preciso fazer boas escolhas com relação aos valores dos parâmetros para obter boas soluções.

Assim, fizemos alguns testes analisando a variação do tamanho da população, para ver se aumentando seu valor, seria possível aumentar o espaço de busca do algoritmo

e tentar a convergência para soluções melhores ou mais rápida e, também, para analisar a variabilidade dos resultados. Destes testes pudemos perceber que não há alterações relevantes no valor da função objetivo encontrado pelo BRKGA variando o tamanho da população, nem mesmo variando o tamanho da instância, seja considerando mais navios com a quantidade de berços fixa ou aumentando os berços, mantendo a quantidade de navios fixa.

Outros testes que fizemos foi executar múltiplas vezes o BRKGA de maneira paralela, utilizando a biblioteca *joblib* do *Python*, com os parâmetros fixos e definidos empiricamente, analisando a melhor solução encontrada e a variabilidade das soluções, para os modelos sem e com ociosidade de berços.

Fixando em 50 execuções para cada instância, foi possível obter soluções melhores do que rodar uma única vez e analisando a diferença entre os valores de função objetivo mínimo e máximo encontrados para cada instância, foi de $8,5\%$ para o modelo sem ociosidade de berços e chegou a $9,5\%$ para o modelo com ociosidade. O valor médio dos valores de função objetivo encontrados também foram menores do que para a estratégia de uma única execução para a maioria das instância, principalmente para as instâncias maiores, e o desvio padrão dos valores foram bem baixos comparados à média, mas em geral aumenta com o tamanho da instância. Esse resultado indica uma certa robustez do BRKGA com relação aos valores de função objetivo encontrados. O tempo computacional somando os tempos de cada execução passou de segundos para alguns minutos nas múltiplas execuções mas considerando o tempo total do processo, que executa em paralelo, aumentou em relação ao BRKGA com uma única execução, mas não chegou na casa dos minutos.

Para os testes com múltiplas execuções, observamos os histogramas com as distribuições dos dados, para entender se o BRKGA proposto apesar de variar, tenderia a concentrar a maioria das soluções em torno do menor valor objetivo. Mas não foi um padrão observado, para diferentes instâncias obtivemos diferentes comportamentos das distribuições, apesar de em nenhum dos testes, a concentração das soluções se deram em torno dos valores objetivos mais altos, ou estão mais centralizados ou mais para a esquerda, que indica os menores valores. Essas conclusões aplicam-se tanto para o modelo sem como para o modelo com ociosidade de berços.

Outra estratégia testada para o BRKGA proposto foi utilizar a ferramenta OPTUNA, mais comumente usada para determinação de hiper parâmetros de modelos de aprendizagem de máquinas. Para essa ferramenta, é necessário definir um intervalo de busca para cada parâmetro que se quer definir, um critério de parada (como tempo de execução), e o algoritmo de busca que a ferramenta utilizará. Nos testes feitos, consideramos o método de busca padrão, o *TPESampler*, com o critério de parada sendo o tempo limite de 30 minutos.

Nos experimentos computacionais realizados, os parâmetros que definem os

critérios de parada também foram inseridos na busca, junto com os parâmetros dos BRKGA (fração da população elite e mutante, da probabilidade de herdar o gene do pai elite e o multiplicador que define o tamanho da população). Para os modelos sem e com ociosidade de berços, rodamos com os intervalos de busca para os parâmetros definidos de acordo com os apresentados na literatura e, para o modelo sem ociosidade, também testamos relaxando os intervalos para além dos indicados na literatura a fim de analisar se as melhores combinações encontradas são de parâmetros dentro do intervalo indicado na literatura.

A ferramenta OPTUNA obteve valores de função objetivo melhores para quase todas as instâncias com relação às múltiplas execuções e que, consequentemente, o BRKGA com uma única execução, para os modelos sem e com ociosidade de berços. Uma única instância para o modelo sem ociosidade de berços que o OPTUNA retornou um valor mais alto, mas com uma porcentagem de variação muito pequena, de $0,1\%$. Para as instâncias maiores de 55 e 60 navios a melhora obtida pelo OPTUNA tende a ser maior, mas não é um padrão.

Com relação ao tempo computacional, temos que o OPTUNA levou em torno de 30 minutos para todas as instâncias, um tempo muito maior do que para as múltiplas execuções com parâmetros fixos. O tempo que o OPTUNA levou para encontrar a melhor solução variou bastante dentre os modelos e instâncias, podendo ter ocorrido dentro dos três primeiros minutos até o último minuto possível.

Através dos gráficos da Figura 36 representam as combinações geradas pelo OPTUNA na busca para o modelo sem ociosidade. Pela figura podemos perceber intervalos de valores para os parâmetros que, combinados, tendem a gerar valores objetivo menores, o que também ocorre para o modelo com ociosidade. Contudo, para as instâncias de 30 navios e 3 berços as variações nas combinações dos parâmetros é maior do que para as instâncias maiores, o que pode ser confirmado pelas Tabelas 53 e 54 para o modelo sem ociosidade e 55 e 56) para o problema com ociosidade.

Para o modelo sem ociosidade de berços, relaxamos os intervalos dos parâmetros sugeridos na literatura mas sem perder as características elitistas do BRKGA, ou seja, a população mutante ser menor que a metade da total, a probabilidade de herdar o gene do pai elite maior que $0,5$ e a população mutante menor que a total também para dar espaço para a fase de cruzamento. Dos testes feitos, o único parâmetro que retornou valores fora do intervalo recomendado foi o que define o tamanho da população elite e, em geral, para as instâncias menores de 30 e 40 navios, com valor maior que indicado na literatura e, para uma instância maior, de 60 navios, retornou um valor menor que o indicado. Os valores objetivos não variaram muito entre os testes para os intervalos relaxados ou não e, para as configurações de parâmetros que contém valores fora do intervalo da literatura também não obtivemos um padrão de melhora ou piora.

Em todos os testes realizados com o BRKGA, para todas as estratégias, as soluções encontradas são factíveis para o modelos. Dessa forma, testamos utilizar as soluções obtidas pela meta-heurística como uma solução inicial factível (ou *warmstart*) para o CPLEX. Testamos rodar o BRKGA uma única vez com os parâmetros fixos, cuja execução ocorre em segundos e não acrescenta muito no tempo total somando a resolução pelo CPLEX e combinando com os tempos limites do OPTUNA, de forma a somar os limites definidos antes somente para o CPLEX, de uma e duas horas.

Como já visto, rodar o BRKGA uma única vez com parâmetros fixos não retorna os melhores valores objetivos e, utilizando essas soluções como *warmstart* para o CPLEX também não resultou em muitas melhoras. As soluções encontradas pelo CPLEX não são melhores do que as obtidas pelo OPTUNA, com metade do tempo de execução. Isso porque o CPLEX não consegue evoluir muito além da solução inicial fornecida pelo BRKGA e, além disso, para algumas instâncias menores, executar o CPLEX sem o *warmstart* fornecido pelo BRKGA, resultou em soluções melhores. Para as instâncias maiores de 55 e 60 navios, todas performaram melhor com o *warmstart*.

Dessa forma, o desempenho com relação ao valor objetivo para a combinação BRKGA + CPLEX foi melhor incluindo a estratégia com o OPTUNA e, como essa estratégia foi testada de forma que a soma do tempo computacional estivesse dentro do limite de uma hora, os tempos computacionais com o OPTUNA e executando uma única vez foram equivalentes. Dentre as diferentes configurações de tempo testadas para OPTUNA + CPLEX, que soma uma hora de tempo total, não foi possível identificar uma melhor, a qualidade da solução varia entre elas. Com relação ao CPLEX sem *warmstart*, as soluções encontradas pelo OPTUNA + CPLEX foram todas melhores, mantendo o tempo total de execução em torno de uma hora.

Uma característica do BRKGA por lidar com vetores de chaves aleatórias e decodificá-las em soluções para o problema, é que podemos gerar as mesmas soluções a partir de diferentes chaves aleatórias. A ideia de buscar uma decodificação que defina de forma independente a ordem e o local de alocação dos navios, é reduzir essa ocorrência. Além disso, conforme o algoritmo vai convergindo, por conta da característica elitista de copiar a população elite e aumentar a probabilidade de herdar um gene do pai elite, as soluções tendem a convergir para uma (boa) alocação, como pode ser observado na Figura 39. Assim, pensando em uma forma de diversificar os cruzamentos mas mantendo a característica elitista destes, propusemos a aplicação de uma clusterização na população e da seleção de indivíduos de diferentes *clusters* para o cruzamento.

Nos testes propostos, aplicamos uma clusterização considerando a distância entre dois indivíduos como a porcentagem de similaridade entre as alocações, isto é, a porcentagem de alocações iguais entre eles, tanto com relação à ordem quanto ao local. Os processo de inserção dos *clusters* na fase de cruzamento ocorreu escolhendo aleatoriamente

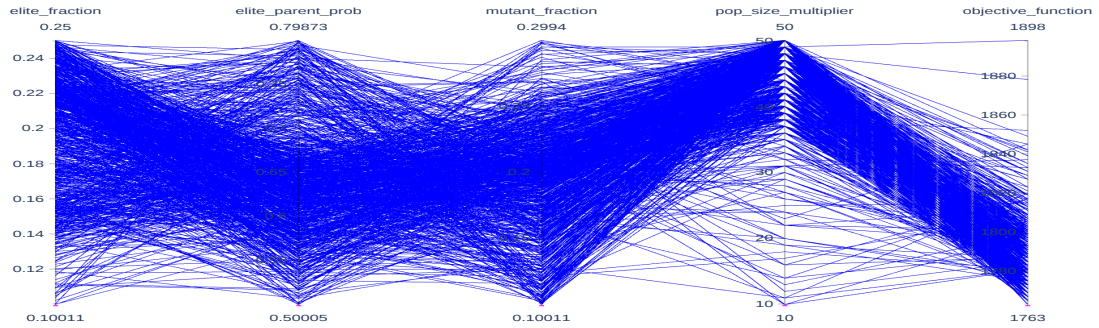
os pais elite e, para cada um, escolhendo um pai não elite de um *cluster* diferente.

Inserindo mais um processo na estratégia de solução, acrescentamos mais alguns parâmetros como o valor que determina a partir de qual iteração as matrizes de distância seriam calculadas, a partir de qual tolerância com relação à média dessas matrizes a clusterização seria calculada e os parâmetros associados ao tipo selecionado de clusterização, que é o tipo de ligação para o cálculo da distância entre dois *clusters* e o nível do corte no dendrograma que define a quantidade de *clusters* e em quais os indivíduos serão alocados. Os testes foram feitos para os modelos sem e com ociosidade de berços e para os testes usados para comparação com as outras estratégias, foram feitas múltiplas execuções (50) em paralelo.

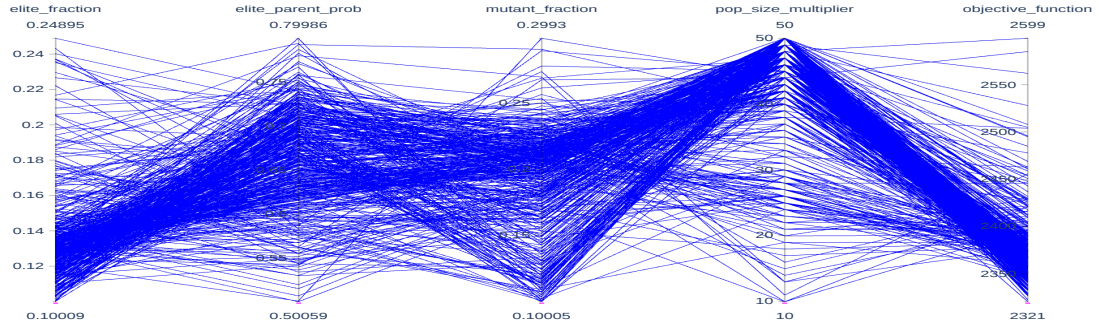
Um resultado inicial foi gerado com uma única execução do BRKGA + Clusterização, com parâmetros fixos, para podermos analisar os *heatmaps* das matrizes de distância e visualizar as diferenças na variação dos indivíduos ao longo das iterações, que pode ser visto na Figura 40. De fato pela figura, podemos perceber que a similaridade entre os indivíduos considerando a clusterização reduziu, gerando populações um pouco mais diversas mesmo nas últimas iterações.

Comparando as diferenças entre os valores objetivo mínimos e máximos encontrados, a variação não está muito distinta do que foi obtido com as múltiplas execuções sem a clusterização. Para o modelo sem ociosidade de berços, o BRKGA + Clusterização obteve melhores valores de função objetivo para mais instâncias do que para o modelo com ociosidade mas, ainda assim, a diferença nas melhoras foram abaixo de 1%. Comparando com os resultados do OPTUNA, o BRKGA + Clusterização só ganhou com relação ao valor da função objetivo na instância f60x5-05 e com uma porcentagem de $0,1\%$. Os resultados para o modelo com ociosidade de berço se apresentaram menos promissores comparando com as múltiplas execuções sem clusterização e, consequentemente, com os resultados obtidos pelo OPTUNA. Contudo, os tempos computacionais do BRKGA + Clusterização para os dois modelos foram menores do que para o OPTUNA mas maiores do que os das múltiplas execuções sem clusterização.

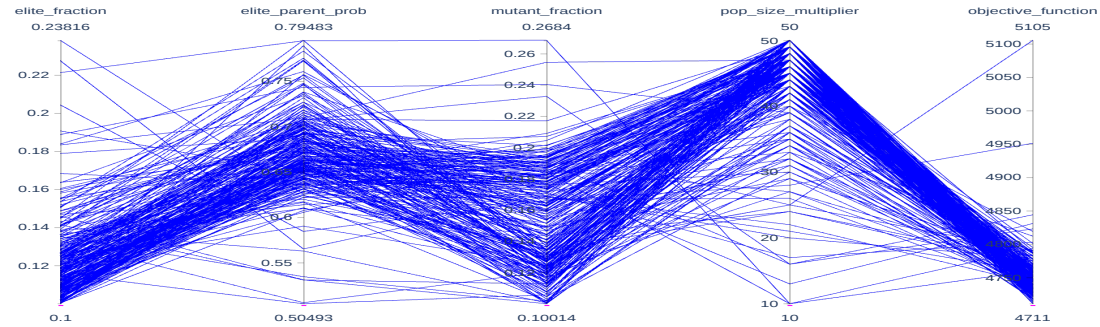
Portanto, das estratégias testadas para o PAB modelado como um PRVJT sem e com ociosidade de berços, a com múltiplas execuções em paralelo sem clusterização apresentaram melhores resultados com relação ao equilíbrio entre valores de função objetivo e tempos computacionais. Caso pequenas melhoras nos valores objetivos resultem maior economia financeira e maior confiabilidade dos portos, então podemos considerar o BRKGA + Clusterização que pode apresentar melhoras e com um tempo computacional intermediário entre as múltiplas execuções sem clusterização e o OPTUNA. Contudo, se levarmos em consideração apenas a qualidade dos valores de função objetivo, podendo dispender um tempo maior, podemos considerar a terceira configuração de tempo para a estratégia OPTUNA + CPLEX.



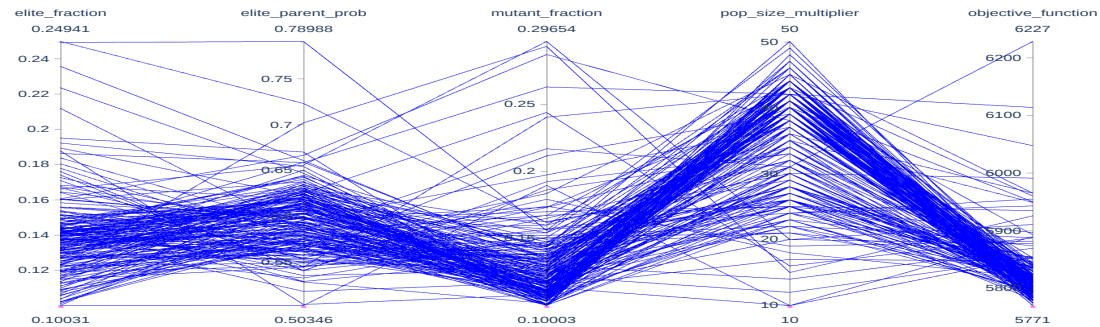
(a) Instância f30x3-01



(b) Instância f40x5-01



(c) Instância f55x5-01



(d) Instância f60x5-1

Figura 36 – *Heatmap* matrizes de distâncias populações BRKGA para diferentes instâncias, modelo sem ociosidade de berços

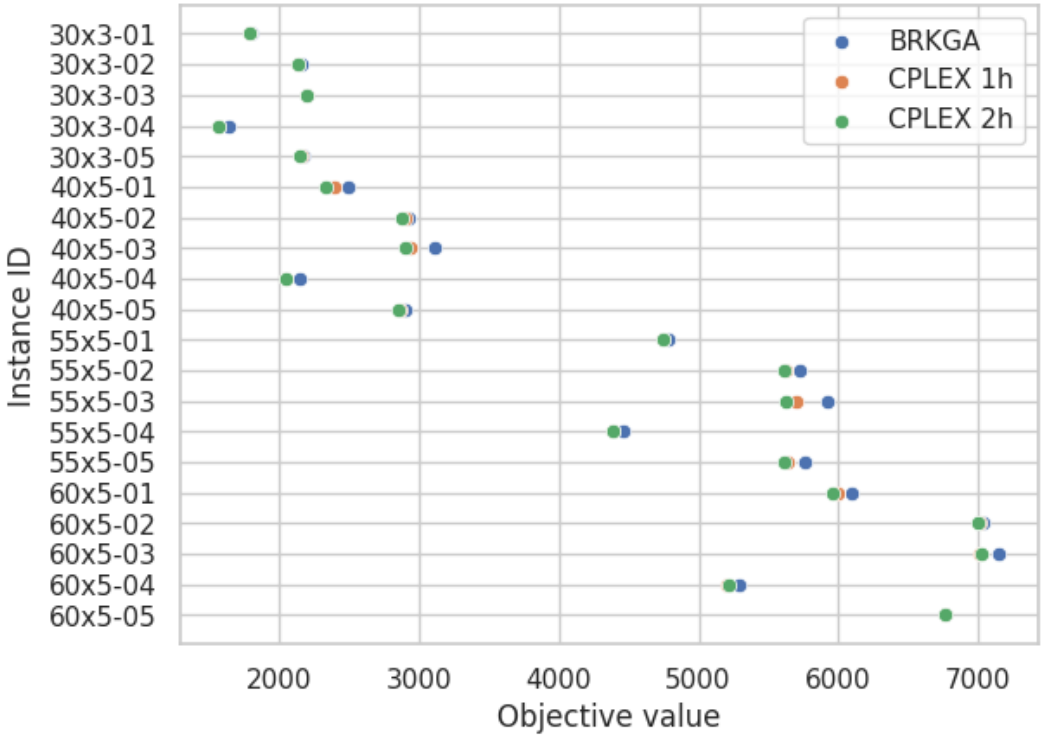


Figura 37 – Valores objetivos das soluções do BRKGA e do CPLEX com *warmstart*, sem ociosidades de berços.

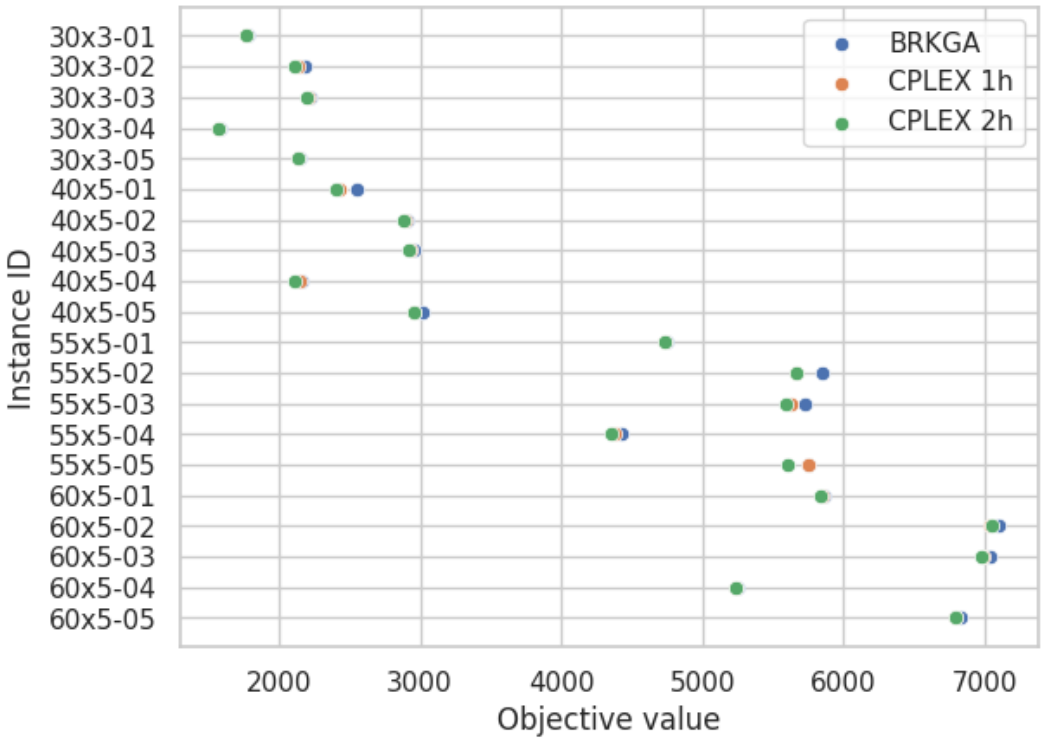
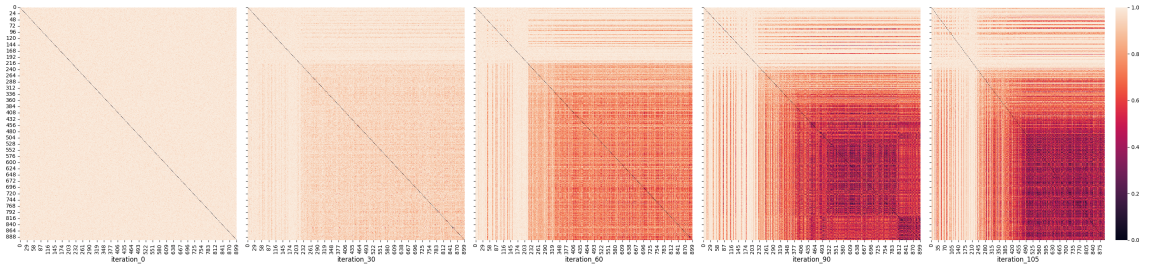
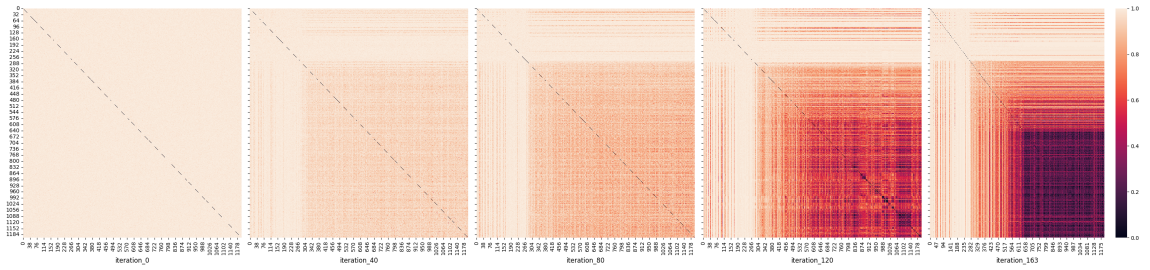


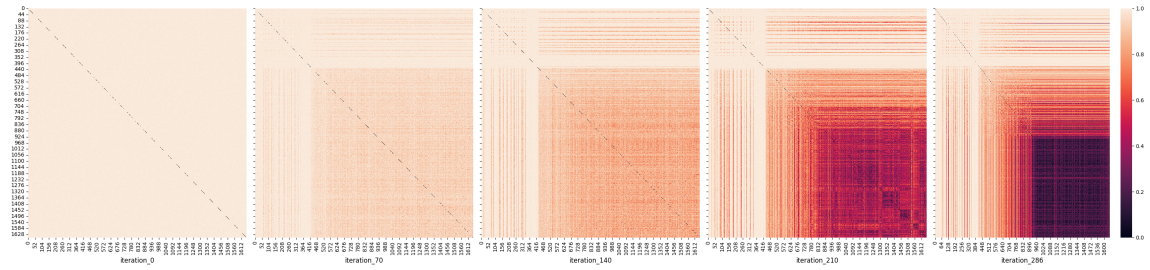
Figura 38 – Valores objetivos das soluções do BRKGA e do CPLEX com *warmstart*, com ociosidades de berços.



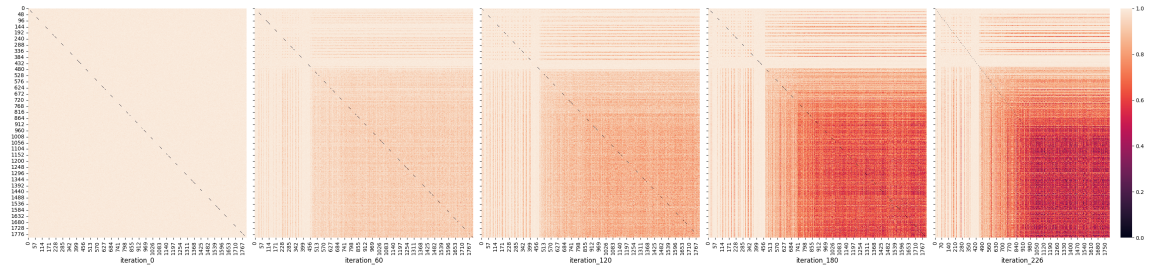
(a) Instância f30x3-03



(b) Instância f40x5-03



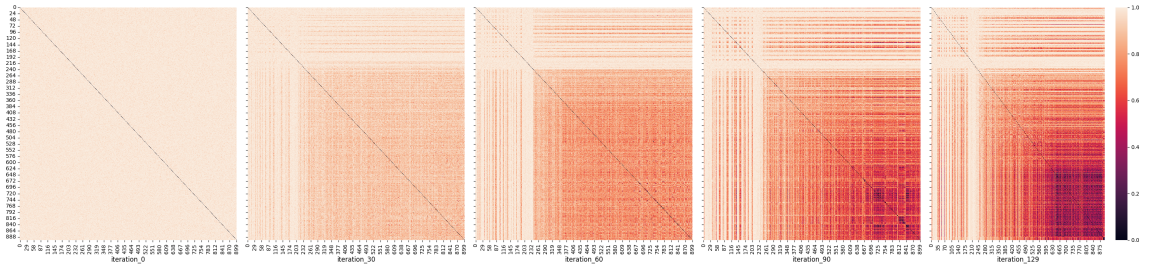
(c) Instância f55x5-03



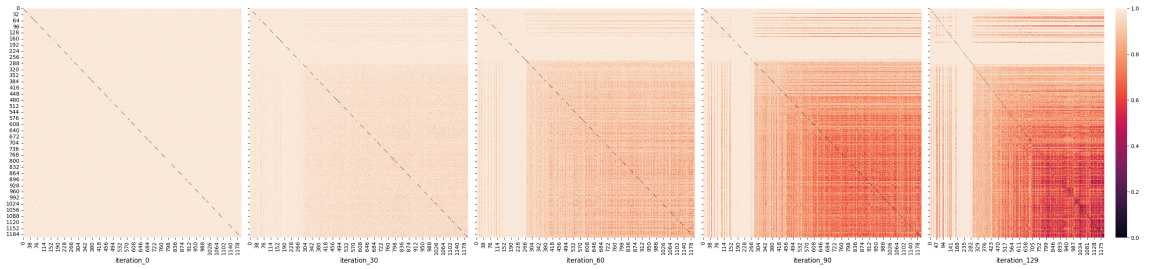
(d) Instância f60x5-03

(e) Instância f60x5-04

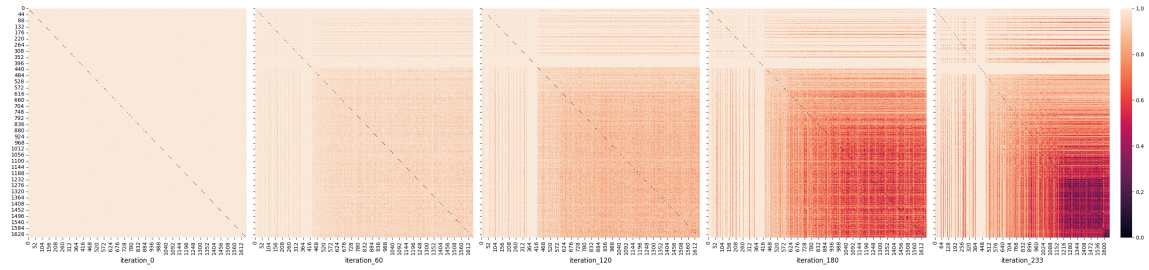
Figura 39 – *Heatmap* matrizes de distâncias populações BRKGA para diferentes intâncias, modelo sem ociosidade de berços.



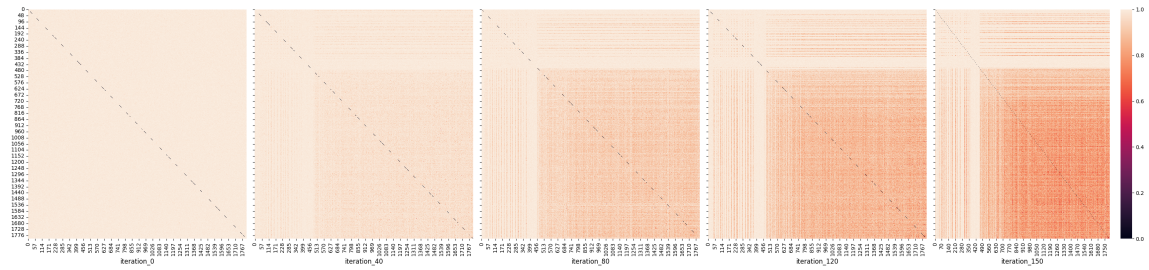
(a) Instância f30x3-03



(b) Instância f40x5-03



(c) Instância f55x5-03



(d) Instância f60x5-03

Figura 40 – *Heatmap* matrizes de distâncias populações BRKGA com clusterização, modelo sem ociosidade de berços.

6 Conclusões

Neste trabalho propusemos uma adaptação do modelo para o Problema de Alocação de Berços (PAB), dinâmico e discreto, proposto por Cordeau et al. (2005), baseado no Problema de Roteamento de Veículos com Janela de Tempo e múltiplas garagens, que minimiza a soma dos tempos de permanência dos navios no porto, permite considerar prioridade de atendimento. Com essa adaptação passamos a minimizar também o tempo de ociosidade dos berços, uma vez que podemos obter diferentes alocações que possuem a mesma soma dos tempos dos navios no porto, mas que resultam em diferentes horários de finalização dos atendimentos dos berços. Dessa forma, buscamos garantir a consideração dos interesses dos portos, finalizando mais cedo os serviços nos berços.

Além da adaptação do modelo para considerar o tempo de ociosidade dos berços, propusemos um Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA) cuja decodificação das ordens e locais de alocação são feitas de forma independente, com uma implementação vetorizada a fim de realizar os cálculos de forma matricial e melhorar o desempenho do algoritmo. A partir desse BRKGA, testamos diferentes estratégias para buscar melhores resultados, incluindo a utilização da ferramenta OPTUNA para busca de hiper parâmetros e uma proposta de clusterização na fase de cruzamento do BRKGA para buscar uma maior diversificação das soluções sem perder a caraterística elitista do algoritmo. Também testamos estratégia de múltiplas execuções em paralelo e combinação com a resolução pelo *solver* CPLEX, uma vez que as soluções obtidas pelo BRKGA foram todas factíveis.

A abordagem que utiliza clusterização na fase de cruzamento foi proposta com a intenção de aumentar a diversidade dos indivíduos, uma vez que verificamos que, com o avanço das iterações, os indivíduos de uma população vão ficando muito parecidos ou até mesmo iguais. A métrica proposta para agrupar indivíduos foi definida de acordo com a quantidade de alocações distintas entre indivíduos, sendo 0 a distância entre soluções iguais e 1 a distâncias entre soluções diferentes. O agrupamento foi considerando que a distância média entre dois indivíduos seja, no máximo, $0,5$. Com essa abordagem, de fato conseguimos aumentar a variabilidade entre os indivíduos e mantivemos a característica elitista, mantendo uma determinada convergência para indivíduos muito parecidos ou iguais com o avanço das iterações.

Os testes computacionais foram realizados para os modelos sem e com ociosidade de berços e o desempenho das estratégias variaram um pouco entre os problemas mas, em geral, estratégias que combinam diferentes abordagens obtiveram melhores resultados. Para o modelo sem ociosidade de berços, considerando um equilíbrio entre qualidade da

solução e tempo computacional, podemos ordenar as três melhores estratégias da seguinte forma:

1. BRKGA com clusterização e parâmetros definidos empiricamente;
2. BRKGA com múltiplas execuções em paralelo;
3. BRKGA com busca de hiper-parâmetros pela ferramenta OPTUNA.

Com relação apenas à qualidade da solução, da melhor para a pior, as abordagens podem ser ordenadas da seguinte forma:

1. CPLEX + BRKGA com OPTUNA;
2. BRKGA com OPTUNA;
3. BRKGA com OPTUNA e Clusterização na fase de cruzamento;
4. BRKGA com Clusterização na fase de cruzamento;
5. BRKGA com múltiplas execuções;
6. CPLEX + BRKGA simples com parâmetros definidos manualmente;
7. BRKGA simples, com parâmetros definidos manualmente e única execução;
8. CPLEX com limite de tempo de processamento.

Considerando o tempo computacional para o modelo sem ociosidade de berços, as abordagens podem ser classificadas do menor para o maior tempo da seguinte forma:

1. BRKGA simples, com única execução e parâmetros definidos manualmente;
2. BRKGA com múltiplas execuções e parâmetros definidos manualmente;
3. BRKGA com Clusterização e parâmetros definidos manualmente;
4. BRKGA com a ferramenta OPTUNA para busca de hiper-parâmetros;
5. BRKGA com OPTUNA e Clusterização na fase de cruzamento;
6. CPLEX e CPLEX com solução inicial fornecida pelo BRKGA.

Para o modelo que considera a ociosidade de berços, concluímos que as três estratégias com melhor equilíbrio entre tempo computacional e qualidade de solução, classificadas de acordo com sua eficácia, são:

1. BRKGA com múltiplas execuções em paralelo e parâmetros definidos manualmente;
2. BRKGA com Clusterização e parâmetros definidos manualmente;
3. BRKGA com parâmetros definidos pela ferramenta OPTUNA.

Classificação de acordo com o menor para o maior valor objetivo, em média, para o modelo com o ociosidade de berços:

1. CPLEX com solução inicial definida pelo BRKGA com a busca de hiper-parâmetros pela ferramenta OPTUNA;
2. BRKGA com Clusterização e parâmetros definidos pelo OPTUNA;
3. BRKGA com a busca de parâmetros pelo OPTUNA;
4. BRKGA com múltiplas execuções em paralelo e parâmetros definidos manualmente;
5. BRKGA com Clusterização e parâmetros definidos manualmente;
6. CPLEX com solução inicial fornecida pelo BRKGA com única execução e parâmetros definidos manualmente;
7. BRKGA com única execução e parâmetros definidos manualmente;
8. CPLEX com limite de tempo de processamento.

Classificando com relação ao tempo computacional, do mais rápido ao mais lento, para o modelo com ociosidade de berços, temos:

1. BRKGA com única execução e parâmetros definidos manualmente;
2. BRKGA com múltiplas execuções em paralelo e parâmetros definidos manualmente;
3. BRKGA com Clusterização e parâmetros definidos manualmente;
4. BRKGA com a busca de parâmetros pelo OPTUNA;
5. BRKGA com Clusterização e parâmetros definidos pelo OPTUNA;
6. CPLEX e CPLEX com solução inicial fornecida pelo BRKGA.

Referências

Agra, A. and Oliveira, M. (2018). Mip approaches for the integrated berth allocation and quay crane assignment and scheduling problem. *European Journal of Operational Research*, 264(1):138–148. Citado na página 35.

Aiex, R. M., Resende, M. G., and Ribeiro, C. C. (2007). Ttt plots: a perl program to create time-to-target plots. *Optimization Letters*, 1(4):355–366. Citado na página 66.

Arefi, M. S. and Rezaei, H. (2015). Problem solving of container loading using genetic algorithm based on modified random keys. *Journal of Advanced Computer Science & Technology*, 4(1):190. Citado na página 62.

Bacalhau, E. T., Casacio, L., and de Azevedo, A. T. (2021). New hybrid genetic algorithms to solve dynamic berth allocation problem. *Expert Systems with Applications*, 167:114198. Citado na página 38.

Bäck, T. and Hoffmeister, F. (1991). Extended selection mechanisms in genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications*. Citado 2 vezes nas páginas 49 e 50.

Barbosa, F. (2014). O problema de alocaçao de berços: Aspectos teóricos e computacionais. Citado na página 66.

Barbosa, F., Moretti, A. C., de Azevedo, A. T., and Neto, L. L. S. (2016). A brief survey of the berth alllocation problem. *PODes*, 8(1):39–56. Citado na página 34.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2):154–160. Citado 2 vezes nas páginas 56 e 57.

Bierwirth, C. and Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202(3):615–627. Citado 6 vezes nas páginas 19, 22, 23, 24, 30 e 33.

Bierwirth, C. and Meisel, F. (2015). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 244(3):675–689. Citado 3 vezes nas páginas 19, 24 e 33.

Brown, G. G., Lawphongpanich, S., and Thurman, K. P. (1994). Optimizing ship berthing. Technical report, Naval Research Logistics. Citado 2 vezes nas páginas 24 e 25.

Buhrkal, K., Zuglian, S., Ropke, S., Larsen, J., and Lusby, R. (2011). Models for the discrete berth allocation problem: A computational comparison. *Transportation Research*

Part E: Logistics and Transportation Review, 47(4):461–473. Citado 7 vezes nas páginas 30, 31, 32, 34, 36, 37 e 83.

Buriol, L. S., Resende, M. G., Ribeiro, C. C., and Thorup, M. (2005). A hybrid genetic algorithm for the weight setting problem in ospf/is-is routing. *Networks: An International Journal*, 46(1):36–56. Citado na página 61.

Cheimanoff, N., Féniès, P., Kitri, M. N., and Tchernev, N. (2023). Exact and metaheuristic approaches to solve the integrated production scheduling, berth allocation and storage yard allocation problem. *Computers & Operations Research*, 153:106174. Citado na página 39.

Cordeau, J.-F., Laporte, G., Legato, P., and Moccia, L. (2005). Models and tabu search heuristics for the berth-allocation problem. *Transportation science*, 39(4):526–538. Citado 18 vezes nas páginas 19, 20, 22, 23, 24, 28, 29, 30, 31, 32, 36, 37, 38, 77, 80, 83, 87 e 174.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 52(8):928–936. Citado na página 28.

Correcher, J. F. and Alvarez-Valdes, R. (2017). A biased random-key genetic algorithm for the time-invariant berth allocation and quay crane assignment problem. *Expert Systems with Applications*, 89:112–128. Citado 2 vezes nas páginas 35 e 67.

Correcher, J. F., Alvarez-Valdes, R., and Tamarit, J. M. (2017). A new mixed integer linear model for the berth allocation and quay crane assignment problem. *Department of Statistics and Operations Research. University of Valencia, Valencia.* Citado na página 69.

Correcher, J. F., Alvarez-Valdes, R., and Tamarit, J. M. (2019). New exact methods for the time-invariant berth allocation and quay crane assignment problem. *European Journal of Operational Research*, 275(1):80–92. Citado na página 36.

Correia, M. B. (2004). Algoritmos genéticos. *Dos Algarves*, (12):36–43. Citado na página 53.

De Jong, K. A. (1975). Analysis of the behavior of a class of genetic adaptive systems. Technical report. Citado 2 vezes nas páginas 41 e 42.

de Lacerda, E. G. and De Carvalho, A. (1999). Introdução aos algoritmos genéticos. *Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais*, 1:99–148. Citado 3 vezes nas páginas 42, 53 e 54.

de Oliveira, R. M., Mauri, G. R., and Lorena, L. A. N. (2012). Clustering search for the berth allocation problem. *Expert Systems with Applications*, 39(5):5499–5505. Citado 3 vezes nas páginas 32, 65 e 97.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354. Citado na página 30.

Elwany, M. H., Ali, I., and Abouelseoud, Y. (2013). A heuristics-based solution to the continuous berth allocation and crane assignment problem. *Alexandria Engineering Journal*, 52(4):671–677. Citado na página 32.

Fogel, D. B. (1994). An introduction to simulated evolutionary optimization. *IEEE transactions on neural networks*, 5(1):3–14. Citado na página 44.

Frojan, P., Correcher, J. F., Alvarez-Valdes, R., Koulouris, G., and Tamarit, J. M. (2015). The continuous berth allocation problem in a container terminal with multiple quays. *Expert Systems with Applications*, 42(21):7356–7366. Citado na página 36.

Gen, M. and Lin, L. (2007). Genetic algorithms. *Wiley Encyclopedia of Computer Science and Engineering*, pages 1–15. Citado 7 vezes nas páginas 19, 43, 44, 45, 46, 47 e 49.

Gendreau, M., Potvin, J.-Y., et al. (2010). *Handbook of metaheuristics*, volume 2. Springer. Citado na página 50.

Giallombardo, G., Moccia, L., Salani, M., and Vacca, I. (2010). Modeling and solving the tactical berth allocation problem. *Transportation Research Part B: Methodological*, 44(2):232–245. Citado 4 vezes nas páginas 33, 64, 65 e 66.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addilson-Wesley. Citado na página 54.

Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Kluwer Academic Publishers-Plenum Publishers*. Citado 2 vezes nas páginas 42 e 46.

Goldberg, D. E., Korb, B., Deb, K., et al. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems*, 3(5):493–530. Citado 4 vezes nas páginas 41, 42, 54 e 55.

Goldberg, D. E., Lingle, R., et al. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of an international conference on genetic algorithms and their applications*, volume 154, pages 154–159. Carnegie-Mellon University Pittsburgh, PA. Citado na página 55.

Gonçalves, J. F. and Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525. Citado 9 vezes nas páginas 40, 56, 57, 58, 60, 61, 63, 94 e 98.

Gonçalves, J. F. and Resende, M. G. C. (2018). *Random-Key Genetic Algorithms*, pages 703–715. Springer International Publishing, Cham. Citado 8 vezes nas páginas 56, 57, 58, 59, 62, 63, 96 e 98.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics*, 16(1):122–128. Citado 2 vezes nas páginas 43 e 51.

Guo, L., Zheng, J., Liang, J., and Wang, S. (2023). Column generation for the multi-port berth allocation problem with port cooperation stability. *Transportation Research Part B: Methodological*, 171:3–28. Citado na página 39.

Hammouti, I., Lajjam, A., Merouani, M., and Tabaa, Y. (2019). A modified sailfish optimizer to solve dynamic berth allocation problem in conventional container terminal. *International Journal of Industrial Engineering Computations*, 10(4):491–504. Citado na página 36.

Hansen, P. and Oğuz, C. (2003). A note on formulations of static and dynamic berth allocation problems. *Les Cahiers du GERAD ISSN*, 711:2440. Citado 4 vezes nas páginas 27, 37, 82 e 106.

Hansen, P., Oğuz, C., and Mladenović, N. (2008). Variable neighborhood search for minimum cost berth allocation. *European Journal of Operational Research*, 191(3):636–649. Citado na página 37.

Holland, J. H. (1975). Adaptation in natural and artificial systems. *The University of Michigan Press, Ann Arbor*, 1:975. Citado 3 vezes nas páginas 41, 42 e 52.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73. Citado 8 vezes nas páginas 19, 41, 42, 44, 45, 49, 52 e 53.

Imai, A., Nagaiwa, K., and Tat, C. W. (1997). Efficient planning of berth allocation for container terminals in asia. *Journal of Advanced Transportation*, 31(1):75–94. Citado 3 vezes nas páginas 25, 27 e 78.

Imai, A., Nishimura, E., and Papadimitriou, S. (2001). The dynamic berth allocation problem for a container port. *Transportation Research Part B: Methodological*, 35(4):401–417. Citado 16 vezes nas páginas 22, 25, 26, 27, 28, 29, 30, 31, 37, 77, 78, 79, 82, 83, 101 e 106.

Imai, A., Nishimura, E., and Papadimitriou, S. (2003). Berth allocation with service priority. *Transportation Research Part B: Methodological*, 37(5):437–457. Citado na página 26.

Imai, A., Nishimura, E., Papadimitriou, S., et al. (2005a). Corrigendum to "The dynamic berth allocation problem for a container port", published in Transportation Research Part B 35 (2001) 401–417. *Transportation research. Part B, Methodological*, 39(3):197–197. Citado 2 vezes nas páginas 83 e 105.

Imai, A., Sun, X., Nishimura, E., and Papadimitriou, S. (2005b). Berth allocation in a container port: using a continuous location space approach. *Transportation Research Part B: Methodological*, 39(3):199–221. Citado na página 30.

Iyoda, E. M. (2000). Inteligência computacional no projeto automático de redes neurais híbridas e redes neurofuzzy heterogêneas. Master's thesis, Universidade Estadual de Campinas, Faculdade de Engenharia Eletrica e de Computação, Campinas, SP. Citado 7 vezes nas páginas 41, 45, 48, 50, 52, 53 e 54.

James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning*, volume 112. Springer. Citado 3 vezes nas páginas 71, 72 e 74.

Jos, B. C., Harimanikandan, M., Rajendran, C., and Ziegler, H. (2019). Minimum cost berth allocation problem in maritime logistics: new mixed integer programming models. *Sādhanā*, 44(6):149. Citado na página 37.

Karafotias, G., Hoogendoorn, M., and Eiben, Á. E. (2014). Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187. Citado na página 51.

Kovač, N., Stanimirović, Z., and Davidović, T. (2018). Metaheuristic approaches for the minimum cost hybrid berth allocation problem. In *Modeling, Computing and Data Handling Methodologies for Maritime Transportation*, pages 1–47. Springer. Citado na página 35.

Kramer, A., Lalla-Ruiz, E., Iori, M., and Voß, S. (2019). Novel formulations and modeling enhancements for the dynamic berth allocation problem. *European Journal of Operational Research*, 278(1):170–185. Citado na página 37.

Kramer, O. (2017). *Genetic algorithm essentials*, volume 679. Springer. Citado 2 vezes nas páginas 46 e 51.

Lalla-Ruiz, E., L.González-Velarde, J., Melián-Batista, B., and Moreno-Vega, J. M. (2014). Biased random key genetic algorithm for the tactical berth allocation problem. *Applied Soft Computing*, 22:60–76. Citado 5 vezes nas páginas 33, 63, 64, 65 e 67.

Lalla-Ruiz, E., MeliáN-Batista, B., and Moreno-Vega, J. M. (2012). Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem. *Engineering Applications of Artificial Intelligence*, 25(6):1132–1141. Citado 7 vezes nas páginas 31, 32, 37, 38, 101, 102 e 116.

Lee, D.-H., Chen, J. H., and Cao, J. X. (2010). The continuous berth allocation problem: A greedy randomized adaptive search solution. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):1017–1029. Citado na página 36.

Li, B., Elmi, Z., Manske, A., Jacobs, E., Lau, Y.-y., Chen, Q., and Dulebenets, M. A. (2023). Berth allocation and scheduling at marine container terminals: A state-of-the-art review of solution approaches and relevant scheduling attributes. *Journal of Computational Design and Engineering*, 10(4):1707–1735. Citado na página 39.

Lim, A. (1998). The berth planning problem. *Operations research letters*, 22(2-3):105–110. Citado 2 vezes nas páginas 19 e 30.

Lin, S.-W., Ting, C.-J., and Wu, K.-C. (2018). Simulated annealing with different vessel assignment strategies for the continuous berth allocation problem. *Flexible Services and Manufacturing Journal*, 30(4):740–763. Citado na página 36.

Londe, M. A., Pessoa, L. S., Andrade, C. E., and Resende, M. G. (2024). Biased random-key genetic algorithms: A review. *European Journal of Operational Research*. Citado na página 20.

Lopes, A. T., Schulz, V. M. L., and Mauri, G. R. (2011). Grasp com path relinking para o problema de alocação de berços. *Pesquisa Operacional para o Desenvolvimento*, 3(3):218–229. Citado na página 97.

Lucena, M. L., Andrade, C. E., Resende, M. G., and Miyazawa, F. K. (2014). Some extensions of biased random-key genetic algorithms. *XLVI Simpósio Brasileiro de Pesquisa Operacional [Internet], Salvador - BA*, 28(3):2469–80. Citado na página 62.

Martin, M. P., Cavalheiro, E. M. B., Barbosa, F., Lyra Filho, C., and Moretti, A. C. (2015). Metodos exatos e uma abordagem heuristica para o problema de alocação de berço. *XLVII SBPO-Simpósio Brasileiro de Pesquisa Operacional. Porto de Galinhas - PE*. Citado na página 66.

Mauri, G. R., Oliveira, A. C., and Lorena, L. A. N. (2008a). A hybrid column generation approach for the berth allocation problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 110–122. Springer. Citado 4 vezes nas páginas 32, 93, 95 e 97.

Mauri, G. R., Oliveira, A. C. M., and Lorena, L. A. N. (2008b). Heurística baseada no simulated annealing aplicada ao problema de alocação de berços. *Gestão de Produção, Operação e Sistemas (GEPROS)*, 3(1):113–127. Citado 7 vezes nas páginas 22, 23, 24, 29, 32, 65 e 96.

Mauri, G. R., Oliveira, A. C. M. d., and Lorena, L. A. N. (2010). Resolução do problema de alocação de berços através de uma técnica de geração de colunas. *Pesquisa Operacional*, 30(3):547–562. Citado 2 vezes nas páginas 29 e 45.

Mauri, G. R., Ribeiro, G. M., Lorena, L. A. N., and Laporte, G. (2016). An adaptive large neighborhood search for the discrete and continuous berth allocation problem. *Computers & Operations Research*, 70:140–154. Citado na página 97.

Meisel, F. and Bierwirth, C. (2009). Heuristics for the integration of crane productivity in the berth allocation problem. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):196–209. Citado na página 69.

Michalewicz, Z. (1995). Genetic algorithms, numerical optimization, and constraints. In *Proceedings of the sixth international conference on genetic algorithms*, volume 195, pages 151–158. Citeseer. Citado na página 42.

Michalewicz, Z. (1996). *Genetic Algorithms+ Data Structures= Evolution Programs*. Springer Science & Business Media. Citado 8 vezes nas páginas 40, 44, 45, 46, 48, 49, 50 e 96.

Michalewicz, Z. and Janikow, C. Z. (1991). Genetic algorithms for numerical optimization. *Statistics and Computing*, 1(2):75–91. Citado na página 42.

Michalewicz, Z. and Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32. Citado 5 vezes nas páginas 42, 44, 47, 48 e 55.

Monaco, M. F. and Sammarra, M. (2007). The berth allocation problem: a strong formulation solved by a lagrangean approach. *Transportation Science*, 41(2):265–280. Citado 2 vezes nas páginas 30 e 31.

Nerurkar, P., Shirke, A., Chandane, M., and Bhirud, S. (2018). Empirical analysis of data clustering algorithms. *Procedia Computer Science*, 125:770–779. Citado 3 vezes nas páginas 71, 72 e 73.

Nishi, T., Okura, T., Lalla-Ruiz, E., and Voß, S. (2020). A dynamic programming-based matheuristic for the dynamic berth allocation problem. *Annals of operations research*, 286(1):391–410. Citado 2 vezes nas páginas 37 e 38.

Nishimura, E., Imai, A., and Papadimitriou, S. (2001). Berth allocation planning in the public berth system by genetic algorithms. *European Journal of Operational Research*, 131(2):282–292. Citado na página 26.

Osman, I. H. and Laporte, G. (1996). Metaheuristics: A bibliography. Citado na página 41.

Oyelade, J., Isewon, I., Oladipupo, O., Emebo, O., Omogbadegun, Z., Aromolaran, O., Uwoghiren, E., Olaniyan, D., and Olawole, O. (2019). Data clustering: Algorithms and its applications. In *2019 19th International Conference on Computational Science and Its Applications (ICCSA)*, pages 71–81. IEEE. Citado 2 vezes nas páginas 71 e 73.

Pan, Q.-Q., Tasgetiren, M. F., and Liang, Y.-C. (2007). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 126–133. Citado na página 69.

Park, Y. M. and Kim, K. H. (2003). A scheduling method for berth and quay cranes. *OR Spectrum*, 25:1–23. Citado 3 vezes nas páginas 24, 27 e 69.

Peteam, F. B. (2016). Relaxação lagrangiana aplicada a um modelo de alocação de navios em berços. Master's thesis, University of Campinas. Citado 3 vezes nas páginas 22, 84 e 162.

Pomari, C. Z. (2014). Algoritmo genérico com chaves aleatórias viciadas para problemas de otimização em portos. Master's thesis, Universidade Federal de São Paulo (UNIFESP). Citado na página 65.

Prasetyo, H., Fauza, G., Amer, Y., and Lee, S.-H. (2015). Survey on applications of biased-random key genetic algorithms for solving optimization problems. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 863–870. IEEE. Citado 5 vezes nas páginas 61, 62, 63, 119 e 124.

Rashidi, H. and Tsang, E. P. (2013). Novel constraints satisfaction models for optimization problems in container terminals. *Applied Mathematical Modelling*, 37(6):3601–3634. Citado 2 vezes nas páginas 19 e 36.

Resende, M. G., Toso, R. F., Gonçalves, J. F., and Silva, R. (2012). A biased random-key genetic algorithm for the Steiner Triple Covering Problem. *Optimization Letters*, 6(4):605–619. Citado 2 vezes nas páginas 57 e 62.

Rizzi, M. M., Pomari, C. Z., de Oliveira, R. M., Chaves, A. A., and Lorena, L. A. N. (2015). Metaheurística híbrida aplicada ao problema de alocação tática de berços. *XLIX SBPO-Simpósio Brasileiro de Pesquisa Operacional. Blumenau - SC*. Citado na página 66.

Rodrigues, F. and Agra, A. (2022). Berth allocation and quay crane assignment/scheduling problem under uncertainty: A survey. *European Journal of Operational Research*, 303(2):501–524. Citado na página 38.

Saadaoui, Y., Umang, N., and Frejinger, E. (2015). *A column generation framework for berth scheduling at port terminals*. CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise .... Citado 2 vezes nas páginas 33 e 34.

Shukla, A., Pandey, H. M., and Mehrotra, D. (2015). Comparative review of selection techniques in genetic algorithm. In *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, pages 515–519. IEEE. Citado na página 50.

Silva, R., Resende, M. G., and Pardalos, P. M. (2014). Finding multiple roots of a box-constrained system of nonlinear equations with a biased random-key genetic algorithm. *Journal of Global Optimization*, 60(2):289–306. Citado na página 62.

Silva, R. M. d. A., Resende, M. G., Pardalos, P. M., and Faco, J. L. (2013). Biased random-key genetic algorithm for linearly-constrained global optimization. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 79–80. Citado na página 62.

Stahlbock, R. and Voß, S. (2008). Operations research at container terminals: a literature update. *OR spectrum*, 30(1):1–52. Citado 2 vezes nas páginas 19 e 29.

Steenken, D., Voß, S., and Stahlbock, R. (2004). Container terminal operation and operations research-a classification and literature review. *OR spectrum*, 26(1):3–49. Citado 3 vezes nas páginas 18, 28 e 29.

Tangpattanakul, P., Jozefowiez, N., and Lopez, P. (2012). Multi-objective optimization for selecting and scheduling observations by agile earth observing satellites. In *International Conference on Parallel Problem Solving from Nature*, pages 112–121. Springer. Citado na página 63.

Thurman, K. P. (1989). Optimal ship berthing plans. Technical report, Naval Post Graduate School-USA. Citado na página 25.

Toso, R. F. and Resende, M. G. (2015). A C++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 30(1):81–93. Citado 3 vezes nas páginas 62, 98 e 119.

Türkoğulları, Y. B., Taşkın, Z. C., Aras, N., and Altınel, İ. K. (2014). Optimal berth allocation and time-invariant quay crane assignment in container terminals. *European Journal of Operational Research*, 235(1):88–101. Citado 2 vezes nas páginas 36 e 69.

Umang, N., Bierlaire, M., and Vacca, I. (2013). Exact and heuristic methods to solve the berth allocation problem in bulk ports. *Transportation Research Part E: Logistics and Transportation Review*, 54:14–31. Citado na página 34.

UNCTAD (2018). 50 years of review of maritme transport, 1968-2018: Reflecting on the past, exploring the future. Report, United Nations Conference on Trade and Development. Citado 2 vezes nas páginas 17 e 18.

Vacca, I., Bierlaire, M., and Salani, M. (2008). Optimization at container terminals: Status, trends and perspectives (revised version). Technical report. "https://infoscience.epfl.ch/record/138893?v=pdf". Citado na página 19.

Vacca, I., Salani, M., and Bierlaire, M. (2011). Container terminal management: integrated models and large-scale optimization algorithms. In *Seminar series, POST_TALK*. Citado 2 vezes nas páginas 33 e 65.

Vacca, I., Salani, M., and Bierlaire, M. (2013). An exact algorithm for the integrated planning of berth allocation and quay crane assignment. *Transportation Science*, 47(2):148–161. Citado 2 vezes nas páginas 33 e 65.

Wawrzyniak, J., Drozdowski, M., and Sanlaville, É. (2020). Selecting algorithms for large berth allocation problems. *European Journal of Operational Research*, 283(3):844–862. Citado na página 38.

Whitley, D., Rana, S., and Heckendorn, R. B. (1999). The island model genetic algorithm: On separability, population size and convergence. *Journal of computing and information technology*, 7(1):33–47. Citado 2 vezes nas páginas 62 e 119.

World Trade Organization (2016). Trade profiles 2016. Report, World Trade Organization. Citado na página 11.

World Trade Organization (2017). Trade profiles 2017. Report, World Trade Organization. Citado na página 11.

World Trade Organization (2018). Trade profiles 2018. Report, World Trade Organization. Citado na página 11.

World Trade Organization (2019). Trade profiles 2019. Report, World Trade Organization. Citado na página 11.

Xiang, X. and Liu, C. (2021). An expanded robust optimisation approach for the berth allocation problem considering uncertain operation time. *Omega*, 103:102444. Citado na página 38.

Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2:165–193. Citado 2 vezes nas páginas 72 e 73.

Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678. Citado 4 vezes nas páginas 71, 72, 73 e 74.

Yao, X. (1993). An empirical study of genetic operators in genetic algorithms. *Microprocessing and Microprogramming*, 38(1-5):707–714. Citado na página 47.

Zheng, J.-N., Chien, C.-F., and Gen, M. (2015). Multi-objective multi-population biased random-key genetic algorithm for the 3-d container loading problem. *Computers & Industrial Engineering*, 89:80–87. Citado na página 63.

Apêndices

APÊNDICE A – Tabelas completas parâmetros obtidos pelo OPTUNA

Para cada melhor solução encontrada pelo OPTUNA para os parâmetros do BRKGA, temos diferentes configurações encontradas pelo OPTUNA. Nas tabelas a seguir podemos analisar para cada instância essas diferentes configurações que forneceram o menor valor objetivo encontrado pelo OPTUNA, para os modelos sem e com ociosidade de berços.

Lembrando que as nomenclaturas dos parâmetros ns forma $P.x$ representam:

- P. 1: fração da população que será considerada elite;
- P. 2: probabilidade de um herdeiro herdar um gene do pai elite no cruzament;
- P. 3: fração da população reservada para os indivíduos mutantes;
- P. 4: multiplicador da quantidade de navios que define o tamanho da população;
- P. 5: tamanho da população (definido pelo Param. 4 multiplicado pela quantidade de navios);
- P. 6: quantidade máxima de gerações (critério de parada);
- P. 7: quantidade máxima de gerações cuja variação do melhor valor objetivo entre gerações consecutivas é menor que uma tolerância (critério de parada);
- P. 8: tolerância da variação do melhor valor objetivo entre uma geração e outra (associada ao Param. 7).

Como as quantidades de conjuntos de parâmetros que resultaram no melhor valor objetivo encontrado para as instâncias de 30 navio e 3 berços são maiores, dividimos as tabelas para esse tamanho de instância e para o modelo sem e com ociosidade de berços, nas tabelas 53 e 55, respectivamente. Para as outras instâncias, as informações separadas para os modelos sem e com ociosidade de berços encontram-se nas tabelas 54 e 56, respectivamente.

Parâmetros do OPTUNA que resultaram no melhor valor de função objetivo, modelo sem ociosidade de berços

Instância	P. 1	P. 2	P. 3	P. 4	P. 5	P. 5	P. 6	P. 7
f30x3-01	0,159	0,652	0,189	50	1500	3400	27	0,00013
f30x3-01	0,156	0,663	0,248	48	1440	3500	17	0,00012
f30x3-01	0,153	0,546	0,250	49	1470	3500	15	0,00013
f30x3-01	0,153	0,791	0,192	50	1500	3500	29	0,00004
f30x3-01	0,232	0,718	0,152	49	1470	3400	26	0,00005
f30x3-02	0,197	0,629	0,108	44	1320	1500	26	0,00060
f30x3-03	0,169	0,519	0,273	36	1080	2700	28	0,00044
f30x3-03	0,212	0,645	0,121	44	1320	2400	27	0,00077
f30x3-03	0,167	0,519	0,137	37	1110	3300	27	0,00034
f30x3-03	0,208	0,683	0,189	38	1140	2200	30	0,00075
f30x3-03	0,213	0,649	0,262	41	1230	3200	30	0,00071
f30x3-04	0,171	0,558	0,122	49	1470	1500	30	0,00086
f30x3-04	0,246	0,608	0,100	49	1470	3600	11	0,00073
f30x3-04	0,181	0,536	0,115	45	1350	1700	29	0,00090
f30x3-04	0,172	0,518	0,127	49	1470	1200	28	0,00021
f30x3-04	0,125	0,603	0,105	47	1410	1300	29	0,00090
f30x3-04	0,247	0,584	0,193	49	1470	1400	30	0,00042
f30x3-04	0,209	0,620	0,116	48	1440	1300	17	0,00045
f30x3-04	0,141	0,648	0,169	49	1470	4200	16	0,00075
f30x3-04	0,249	0,580	0,184	50	1500	1300	30	0,00045
f30x3-04	0,166	0,626	0,155	50	1500	800	18	0,00090
f30x3-04	0,171	0,592	0,109	47	1410	1500	30	0,00098
f30x3-04	0,163	0,568	0,128	49	1470	1300	30	0,00091
f30x3-04	0,150	0,614	0,109	48	1440	300	30	0,00081
f30x3-04	0,176	0,585	0,264	49	1470	600	29	0,00097
f30x3-04	0,241	0,624	0,177	50	1500	900	30	0,00081
f30x3-04	0,140	0,563	0,158	33	990	1300	15	0,00059
f30x3-04	0,179	0,645	0,151	48	1440	1600	30	0,00092
f30x3-04	0,212	0,590	0,108	48	1440	1400	30	0,00091
f30x3-05	0,220	0,565	0,161	41	1230	2300	30	0,00066
f30x3-05	0,218	0,551	0,161	41	1230	2300	29	0,00065
f30x3-05	0,205	0,639	0,189	47	1410	4800	27	0,00058
f30x3-05	0,176	0,657	0,218	46	1380	1400	26	0,00083
f30x3-05	0,202	0,606	0,179	39	1170	2000	28	0,00056

Tabela 53 – Todas as combinações dos parâmetros da melhor solução obtida pelo OPTUNA, modelo sem ociosidade de berços, para as instâncias de 30 navios e 3 berços.

Instância	P. 1	P. 2	P. 3	P. 4	P. 5	P. 5	P. 6	P. 7
f40x5-01	0,111	0,724	0,213	42	1680	700	19	0,00092
f40x5-02	0,126	0,528	0,193	41	1640	1800	26	0,00024
f40x5-03	0,134	0,596	0,183	49	1960	4800	21	0,00063
f40x5-04	0,150	0,550	0,198	50	2000	2500	30	0,00084
f40x5-05	0,192	0,656	0,241	43	1720	1200	18	0,00049
f55x5-01	0,123	0,697	0,193	49	2695	3600	26	0,00044
f55x5-01	0,114	0,703	0,128	48	2640	200	25	0,00047
f55x5-01	0,121	0,684	0,192	49	2695	3600	27	0,00046
f55x5-02	0,142	0,612	0,157	49	2695	2100	30	0,00030
f55x5-03	0,210	0,702	0,220	35	1925	4000	30	0,00029
f55x5-04	0,175	0,737	0,195	49	2695	1800	26	0,00085
f55x5-04	0,195	0,706	0,180	46	2530	1900	28	0,00080
f55x5-04	0,163	0,668	0,146	46	2530	2300	27	0,00079
f55x5-04	0,168	0,698	0,166	47	2585	1700	29	0,00096
f55x5-05	0,205	0,594	0,174	48	2640	1600	30	0,00029
f60x5-01	0,131	0,614	0,112	40	2400	1600	20	0,00040
f60x5-02	0,234	0,675	0,198	38	2280	3500	24	0,00006
f60x5-03	0,236	0,656	0,151	48	2880	2600	25	0,00040
f60x5-04	0,118	0,605	0,289	49	2940	4000	30	0,00033
f60x5-05	0,197	0,756	0,179	45	2700	3600	13	0,00009

Tabela 54 – Todas as combinações dos parâmetros da melhor solução obtida pelo OPTUNA, modelo sem ociosidade de berços.

Parâmetros do OPTUNA que resultaram no melhor valor de função objetivo, modelo com ociosidade de berços

Instância	P. 1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8
f30x3-01	0,241	0,666	0,189	29	870	4700	16	0,0004
f30x3-01	0,172	0,644	0,172	36	1080	4000	15	0,0004
f30x3-01	0,203	0,659	0,195	41	1230	3900	16	0,0008
f30x3-02	0,232	0,564	0,162	40	1200	2900	23	0,0003
f30x3-02	0,176	0,532	0,108	49	1470	2900	23	0,00001
f30x3-03	0,165	0,571	0,119	46	1380	3800	26	0,0008
f30x3-03	0,239	0,582	0,159	49	1470	3600	27	0,0009
f30x3-03	0,180	0,532	0,181	50	1500	1700	26	0,0008
f30x3-03	0,242	0,569	0,147	48	1440	1400	27	0,0008
f30x3-04	0,207	0,771	0,102	40	1200	800	17	0,0009
f30x3-04	0,242	0,724	0,109	39	1170	2500	20	0,0007
f30x3-04	0,140	0,552	0,120	41	1230	4800	24	0,0001
f30x3-04	0,240	0,767	0,104	41	1230	2400	22	0,0006
f30x3-04	0,210	0,614	0,126	43	1290	1800	22	0,0008
f30x3-04	0,133	0,620	0,199	40	1200	4200	21	0,00003
f30x3-04	0,160	0,532	0,204	42	1260	900	23	0,0001
f30x3-05	0,151	0,677	0,176	47	1410	4000	27	0,0003
f30x3-05	0,210	0,654	0,111	50	1500	5000	27	0,00003
f30x3-05	0,210	0,662	0,112	50	1500	4600	27	0,0009
f30x3-05	0,126	0,526	0,268	49	1470	4600	28	0,0003
f30x3-05	0,238	0,679	0,170	50	1500	2100	23	0,0007
f30x3-05	0,231	0,716	0,193	49	1470	2500	21	0,0004
f30x3-05	0,231	0,707	0,190	49	1470	2400	22	0,0004
f30x3-05	0,245	0,759	0,157	50	1500	4600	28	0,0003
f30x3-05	0,202	0,656	0,132	47	1410	3900	23	0,0003
f30x3-05	0,163	0,777	0,202	47	1410	4700	17	0,0002
f30x3-05	0,191	0,608	0,178	47	1410	4300	27	0,0003
f30x3-05	0,234	0,634	0,133	49	1470	3900	20	0,0001

Tabela 55 – Todas as combinações dos parâmetros da melhor solução obtida pelo OPTUNA, modelo com ociosidade de berços, para as instâncias de 30 navios e 3 berços.

Instância	P. 1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8
f40x5-01	0,155	0,511	0,157	50	2000	2700	27	0,0010
f40x5-02	0,212	0,632	0,154	50	2000	4400	25	0,0002
f40x5-03	0,203	0,669	0,148	49	1960	2300	21	0,0001
f40x5-04	0,196	0,705	0,162	38	1520	1100	28	0,0005
f40x5-05	0,129	0,583	0,291	43	1720	800	26	0,0002
f55x5-01	0,143	0,693	0,209	47	2585	700	21	0,0003
f55x5-01	0,103	0,711	0,134	50	2750	1400	20	0,0003
f55x5-02	0,150	0,693	0,133	48	2640	2500	26	0,0002
f55x5-03	0,124	0,629	0,249	48	2640	3700	19	0,0008
f55x5-04	0,216	0,659	0,145	45	2475	2200	29	0,0005
f55x5-05	0,199	0,713	0,216	44	2420	2800	23	0,0003
f60x5-01	0,153	0,735	0,152	47	2820	1700	22	0,0010
f60x5-02	0,221	0,668	0,146	43	2580	2900	22	0,0002
f60x5-03	0,212	0,719	0,107	45	2700	900	18	0,0003
f60x5-04	0,164	0,675	0,203	50	3000	800	20	0,00003
f60x5-05	0,220	0,736	0,148	45	2700	4400	22	0,0005

Tabela 56 – Todas as combinações dos parâmetros da melhor solução obtida pelo OPTUNA, modelo com ociosidade de berços.