UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA
CAMPUS DE ILHA SOLTEIRA

Bruna Gonçalves de Lima

META-HEURÍSTICA AGE-E APLICADA A PROBLEMAS DE CARREGAMENTO DE CONTÊINERS

Ilha Solteira

2017

META-HEURÍSTICA AGE-E APLICADA A PROBLEMAS DE CARREGAMENTO DE CONTÊINERS

Bruna Gonçalves de Lima[1]

Tese apresentada à Faculdade de Engenharia do Campus de Ilha Solteira-UNESP como parte dos requisitos para a obtenção do título de Doutora em Engenharia Elétrica. Área do conhecimento: Automação.

Prof. Dr. Rubén Augusto Romero Lázaro
Orientador

Ilha Solteira
2017

[1]Contato:bruna.glv.lima@gmail.com

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

L732m

Lima, Bruna Gonçalves de.
Meta-heurística age-e aplicada a problemas de carregamento de contêiners Bruna Gonçalves de Lima. -- Ilha Solteira: [s.n.], 2017
199 f. : il.

Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia. Área de conhecimento: Automação, 2017

Orientador: Ruben Augusto Romero Lázaro
Inclui bibliografia

1. Problema de carregamento de contêiners. 2. Problema da embalagem. 3. Meta-heurística. 4. Algoritmo genético.

CRB8 4740

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA TESE: Meta-heurística AGE-E aplicada a problemas de carregamento de contêiners

AUTORA: BRUNA GONÇALVES DE LIMA
ORIENTADOR: RUBEN AUGUSTO ROMERO LAZARO

Aprovada como parte das exigências para obtenção do Título de Doutora em ENGENHARIA ELÉTRICA, área: AUTOMAÇÃO pela Comissão Examinadora:

Prof. Dr. RUBEN AUGUSTO ROMERO LAZARO
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. JOSE ROBERTO SANCHES MANTOVANI
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. SILVIO ALEXANDRE DE ARAUJO
Departamento de Matematica Aplicada / Instituto de Biociências, Letras e Ciências Exatas

LINA PAOLA GARCES N.
Profa. Dra. LINA PAOLA GARCES NEGRETE
Departamento de Engenharia Elétrica / Universidade Federal de Goias

Prof. Dr. LUIS GUSTAVO WESZ DA SILVA
Departamento de Industrias / Instituto Federal de Educação, Ciência e Tecnologia de Goiás

Ilha Solteira, 06 de setembro de 2017

Ao meu amado filho Pietro.
Dedico.

AGRADECIMENTOS

À Deus, sempre, pela vida, oportunidades e conquistas.

Ao meu professor Rubén pela orientação, dedicação e conselhos durante toda a elaboração deste trabalho. Obrigada pela compreensão e auxílio durante todo o período de estudo. O senhor com certeza facilitou a evolução de meu conhecimento e foi o diferencial de minha caminhada dentro da Universidade. Não poderia ter tido orientador melhor. Muito obrigada, mesmo. E por confiar em meu trabalho.

Ao meu filho amado, Pietro, por facilitar a minha vida, tornando-a mais leve e com momentos de descontração. Você é a razão de existência desse trabalho, é o que faz tudo fazer sentido na vida da mamãe. É por você que continuo estudando e me aprimorando, para proporcionar a você melhores condições de vida, e uma educação onde eu possa exercer o papel de mãe com mais qualidade. Obrigada, meu filho, por você existir e poder compartilhar comigo todas as nossas conquistas, desde o momento que você nasceu. Obrigada por você ser meu companheiro e amigo em todos os momentos, de maiores ou menores dificuldades. Você, meu filho, até me lembrava que eu precisava estudar, mesmo quando queria que eu assistisse Harry Potter com você. Obrigada por tanto carinho e compreensão.

À minha mãe, Sandra, pela vida e por me apoiar e auxiliar em tudo que faço com tanto amor e carinho, em todos os momentos de minha vida e, em particular, em minha evolução educacional. Quantas vezes você ficou com o Pietro para que eu pudesse estudar! Você mãe é o motivo de eu ter conseguido concluir este trabalho em tempo factível! Muito obrigada mãe amada! Ao meu pai em especial que, mesmo não tendo conseguido vivenciar o final de mais essa conquista, sempre está me dando forças através das lembranças em meu coração que me ensinaram a sempre acreditar em minhas vitórias, apesar das dificuldades. A vontade de vencer superou a saudade de não tê-lo, pai, assim como você sempre me ensinou.

À minha irmã Laura, uma criança que só traz alegrias e amor à nossa família, sempre com um sorriso no rosto, pronta a nos fazer sorrir também. À minha avó Alice e tias, Maraísa e Liliamaura, por me apoiarem no decorrer desta jornada, com palavras de incentivo e amor, que me fizeram permanecer no caminho da persistência e perseverança.

À todos os amigos, conquistados durante a vida, que tanto me auxiliaram no decorrer destes anos. À Fernanda e Thaís, minhas amigas incondicionais e irmãs de coração, que trazem o companherismo e cumplicidade para minha vida. Em especial, gostaria de agradecer à Cristina por sempre estar presente nas horas que mais precisei. Por, mesmo estando tão longe, ser minha

amiga e auxiliar nos momentos de estudo. Nunca poderei agradecer o suficiente à você Cristina, por me receber de braços abertos desde o início do curso e por facilitar tanto a minha vivência na Universidade. Obrigada por ser uma amiga tão carinhosa e generosa.

Não poderia também deixar de agradecer ainda ao meu companheiro, ao meu amor, que ilumina todos os dias de minha vida e de meu filho: Vinicius. Você merece todos os créditos por eu ter conseguido terminar meus estudos. Você, além de companheiro, foi meu amigo, meu ajudante, meu leal namorado. Passou noites comigo, ouviu minhas angústias e me fez voltar ao entusiasmo de terminar este trabalho. Soube respeitar meu tempo de estudos sozinha, e soube intervir nos momentos necessários para me fazer descontrair. Obrigada, obrigada! Nunca poderei agradecer o suficiente! Sem você, este processo teria sido muito mais árduo e difícil. Você fez a diferença para que este trabalho chegasse até o final!

À todos os professores que colaboraram para a minha formação.

"A Matemática apresenta invenções tão sutis que poderão servir não só para satisfazer os curiosos, como também para auxiliar as artes e poupar trabalho aos homens."

Descartes

RESUMO

Neste trabalho apresenta-se uma nova meta-heurística, o Algoritmo Genético Evolucionário Especializado (AGE-E) para resolver uma das categorias dos Problemas de Carregamento de Contêiners, objeto de estudo que pertence à otimização, na Pesquisa Operacional. Considera-se a existência de múltiplos contêiners de iguais dimensões que permitem o carregamento completo da carga disponível em um contexto de transporte industrial. Esta carga é composta por caixas de sortimento fortemente hete-rogêneo e que permite a rotação em qualquer das seis possibilidades, tornando o problema ainda mais complexo, e, por isso, menos estudado na literatura. Uma revisão bibliográfica é também apresentada, contendo uma visão geral das classificações do problema e, em particular, um estudo aprofundado sobre algoritmos genéticos. A implementação do AGE-E foi realizada, e os resultados computacionais foram comparados com as melhores soluções já apresentadas na literatura, demonstrando o potencial do AGE-E para estudos futuros.

Palavras-chave: Problema de carregamento de contêiners. Problema da embalagem. Meta-heurística. Algoritmo genético.

ABSTRACT

This work presents a new meta-heuristic, the Specialized Evolutionary Genetic Algorithm (AGE-E), which solves one of the categories of Container Loading Problems, object of study that belongs to Optimization, within the Operational Research. It's considered the existence of multiple containers of the equal dimensions that promote the full loading of the available cargo in industrial transportation context. This load is composed of strongly heterogeneous assortment to the boxes, and allows rotation in any of the six possibilities, making the problem even more complex, and therefore less studied in the literature. A bibliographic review is also presented, containing an overview of the classifications of the problem and, in particular, an deepened study on genetic algorithms. The implementation of AGE-E was performed, and the computational results were compared with the best solutions already determined by the bibliography, demonstrating the potential of AGE-E for future studies.

Keywords: Container loading problem. Bin packing problem. Metaheuristic. Genetic algorithm.

SUMÁRIO

1 INTRODUÇÃO

A expansão da eletrônica fez surgir, em 1970, a chamada Terceira Revolução Industrial, marco da evolução dos processos tecnocientíficos, e que foi determinada pela integração física entre a tecnologia e os processos industriais. Esse movimento de mudanças na cadeia de produção, justificado pela necessidade de impulsionar o progresso tão desgastado pela até então recente Segunda Guerra Mundial, foi determinado pelo desenvolvimento da informática e da tecnologia, que aqueceu o mercado e a economia mundial, e que proporcinou a globalização destes processos, desde a compra de matéria-prima até a distribuição da mercadoria para qualquer lugar do mundo.

O avanço tecnológico atingiu os diversos segmentos da estrutura produtiva de alguns blocos de países, tais como Estados Unidos, Japão e principais países da Europa, em especial a Alemanha, expandindo-se, posteriormente, para os demais países, e, em especial, para Coreia e China, "alterando os padrões de organização e gerando um forte aumento da produtividade e uma acentuada redução dos custos unitários de produção" [1]. Com o aumento da demanda destes produtos e com o problema de serem fabricados em locais diferentes, foram exigidas soluções em como transportar estes produtos até os clientes em potencial, com o objetivo de minimizar os custos empresariais e de maximizar a satisfação do cliente. Além disso, a possibilidade de se fabricar produtos em larga escala a preços inferiores tornou indispensável a ampliação da capacidade de estocagem de produtos em maiores quantidades à logística empresarial.

Por retratarem demandas de custo elevado, problemas relacionados ao transporte e à estocagem de mercadorias preconizaram caracterizações distintas para as concepções e propósitos do vigente contexto manufatureiro que se transformava. E, para acompanhar as necessidades deste cenário econômico cada vez mais exigente por inovações e novas estratégias de competitividade, teve início a estruturação de estudos científicos aplicados ao melhoramento dos processos produtivos, aos quais foram denominados *Pesquisa Operacional* (PO). Previamente considerada um termo militar, a PO surgiu em 1938 como um segmento de estudo da matemática, aplicada às táticas de guerra, e com o intuito de inovar e otimizar as estratégias de defesa e de análise de operações no uso dos recursos tecnológicos militares. Após a Segunda Guerra Mundial, a PO expandiu-se para os setores civis, disseminada por matemáticos especializados com uma Pesquisa Operacional que deixava de existir, e que, portanto, estava apta a ser transplantada para a propícia brecha de demandas do incipiente sistema produtivo que necessitava ser preenchida. O desenvolvimento de algoritmos simples, tal como o Método Simplex de George Dantizig, em 1947, bem como o surgimento de microcomputadores, que possuíam alto desempenho de processamento,

impulsionaram ainda mais a disseminação da PO, que foi introduzida dentro de cursos de graduação e pós-graduação em todo o mundo, e, em particular, no Brasil, a partir de 1960.

No que concerne aos problemas de estocagem, a Pequisa Operacional produziu algumas soluções, tal como o processo de "se estocar uma única vez um produto, por exemplo, em um armazém de distribuição em local intermediário entre a empresa e o cliente" [2], e uma vez que o custo de estocagem representa grande parte dos custos logísticos totais da empresa. Outras duas soluções encontradas foram: a unitilização da carga, ou seja, fazer com que vários itens devam ser arranjados de forma a poderem ser manuseados como um único volume; e a paletização da carga, "que tem como característica reunir vários itens em uma plataforma, em geral de madeira, disposta horizontalmente, a fim de que a carga possa ser empilhada e estabilizada" [2]. Em particular, na paletização da carga, há um melhor controle de estoque, "a carga fica protegida contra roubos e danos físicos e pode ser transportada de uma só vez por meio de empilhadeiras, guindastes, caminhões, diminuindo o tempo de carga e descarga" [3].

Para os problemas de transporte, o surgimento de empresas de logística especializadas na vinculação entre clientes e fornecedores, fez com que, segundo Cecílio [4], houvesse uma maior integração entre ambos, devido à potencialização dos resultados provenientes do melhoramento das tomadas de decisão fornecidas pela PO. Estratégias de planejamento, de implementação e de controle do fluxo eficiente dos produtos a serem transportados foram desenvolvidas com o objetivo de facilitar as operações e disponibizar ganhos econômicos ao propiciar menores custos às empresas. A conquista mais importante neste segmento foi a criação de contêiners, que modificaram o sistema de carregamento e transporte de mercadorias, como por exemplo, em navios, que até há 50 anos, exigiam o trabalho de centenas de homens para promover seu total preenchimento com a carga disponível. Atualmente, com a utilização de contêiners, um único operador faz o trabalho, diminuindo significantemente os custos com mão-de-obra, frete, roubos e avarias. Além disso, facilita-se "a movimentação, armazenagem e transporte da carga, reduz-se o número de volume a ser manipulado, minimiza-se o tempo de operação de embarque e desembarque, e diminuem-se os custos com embalagem" [5].

Utida [5] define o contêiner como sendo uma caixa retangular de grandes dimensões, construída na maioria das vezes com metal ou madeira, e que é utilizado para transportar pequenas caixas, nas quais, cada uma destas, contém mercadorias. Em geral, os carregamentos são realizados por funcionários contratados pela empresa, e que são gerenciados por um responsável que orienta e superviosiona a alocação das caixas dentro do contêiner, dependendo de aproximações visuais. Ou seja, de modo geral, os carregamentos são realizados de acordo com a experiência de um responsável, e ficam limitados a um conhecimento não-técnico, que dependem do fator humano.

Essa dependência impacta de forma significativa o modelo produtivo de empresas de transporte e influencia diretamente nas perdas quantitativas e qualitativas, ocasionadas pela falta de otimização na alocação dos produtos dentro do contêiner. Além disso, o crescimento exponencial do sistema de transporte de carregamentos através do embarque de navios - incluindo a articulação entre os possíveis transportes por terra ou ar -, transformou-o em uma rede complexa, repleta de ramificações, e, portanto, vulnerável e suscetível a gargalos, desvios e atrasos. Segundo Wang *et. al* [6], ataques terroristas no contrabando de mercadorias, armamento e furtos, são exemplos de atrasos que podem ser gerados, devido ao tempo disponibilizado à intensa vistoria para a segurança dos carregamentos.

Estas questões e a busca pela minimização dos custos empresariais fizeram com que a Pesquisa Operacional explorasse possibilidades alternativas de transporte de mercadorias dentro de contêiners, que diminuíssem o número de viagens dentro deste amplo sistema de transportes ao otimizar o aproveitamento dos espaços e perdas. Estes problemas foram chamados de Problemas de Carregamento de Contêiners (PCC) e foram subdivididos em diversas categorias que, em sua maioria, trabalham com o encaixe de itens menores dentro de objetos maiores. Por quase sempre retratarem problemas de grande porte, a resolução dos PCC envolve a utilização de métodos computacionais e algoritmos eficientes que buscam melhores configurações para os carregamentos através de estratégias inteligentes, que definem uma disposição adequada dos itens que formam a carga. A categorização dos Problemas de Carregamento de Contêiners foram atualizadas no decorrer das décadas, mas, os primeiros trabalhos científicos que abordaram o tema foram apresentados por Gilmore e Gomory, em seus estudos [7] e [8], propostos, respectivamente, em 1961 e em 1963. A última atualização das categorias de PCC foram realizadas por Bortfeldt e Wäscher [15], em 2013.

Este trabalho apresenta um novo algoritmo, o Algoritmo Genético Evolucionário Especializado (AGE-E), que resolve uma das categorias dos Problemas de Carregamento de Contêiners, o Problema da Embalagem em Objetos de um Único Tamanho (SBSBPP, do inglês, *Single Bin Size Bin Packing Problem*), para o caso tridimensional. Visando promover uma contribuição científica neste segmento da Pesquisa Operacional, o método de solução desenvolvido foi construído a partir de estratégias de qualidade que, até então, não haviam sido inseridas em um mesmo contexto, todas ao mesmo tempo. A análise de resultados computacionais provenientes da implementação computacional também foi realizada para a verificação da eficiência do algoritmo.

1.1 JUSTIFICATIVA

Assim como demonstrado em Martello [9], Garey e Johnson [10] e em Beasley [11], Problemas de Carregamento de Contêiners - PCC têm classificação *NP-Difícil*, o que traduz a elevada complexidade computacional (não polinomial) na apresentação de métodos exatos que determinem soluções ótimas para esses problemas. Em geral, os algoritmos exatos que resolvem o PCC utilizam métodos de busca em árvore, e são aplicados apenas a problemas de pequeno porte. Para aplicações do mundo real, não é suficiente considerar-se estas realidades limitadas, e assim, os algoritmos exatos deixam de ser importantes no contexto prático, dando espaço ao desenvolvimento de heurísticas que apresentam soluções de boa qualidade, em tempo computacional compatível, mesmo não havendo garantias de otimalidade.

Nas últimas décadas, estudos acadêmicos e pesquisas industriais têm sido intensificados, em todo o mundo, principalmente através da Pesquisa Operacional, para fornecer o melhoramento do desempenho computacional por meio de algoritmos heurísticos que apresentem novas metodologias e restrições para o Problema de Carregamento de Contêiners. A evolução da tecnologia e o surgimento de computadores mais eficientes têm ainda contribuído para maximizar a qualidade das soluções encontradas. A necessidade de se aprimorar as formas de se obter boas soluções para o PCC incide sobre a possibilidade de enquadrá-lo em várias aplicações industriais, que envolvam não apenas o carregamento de cargas dentro de contêiners, mas também o carregamento em caminhões, trens, ou veículos quaisquer.

Assim, a justificativa deste trabalho em apresentar uma nova heurística para o Problema de Carregamento de Contêiners é que, além de contribuir com a comunidade acadêmica, a pesquisa poderá promover possíveis reduções dos custos de se transportar e manusear as mercadorias, incentivando o desenvolvimento de novas tecnologias na área, e apresentando a necessidade imprescindível de se criar inovações eficientes para o transporte internacional no processo de se incorporar a pesquisa científica à prática empresarial.

1.2 NOÇÕES PRELIMINARES DO PROBLEMA DE CARREGAMENTO DE CONTÊINERS

O Problema de Carregamento de Contêiners é um problema de engenharia de produção encontrado na Pesquisa Operacional, no qual utiliza-se matemática aplicada para definir e resolver o problema. O PCC consiste em carregar, sem sobreposição, um dado conjunto de itens menores (caixas) dentro de objetos maiores (contêiners) de forma a otimizar o espaço dentro dos contêiners

da melhor forma possível.

Várias restrições podem ser impostas ao problema, como por exemplo, sobre as dimensões, quantidades, tipos, orientação e formas dos itens e objetos, ou sobre limites de peso ou quantidade de caixas dentro dos contêiners.

A carga carregada pode ser colocada diretamente dentro dos contêiners, ou pode ser previamente empilhada sobre chapas de metal ou madeira (*paletes*), para depois ser colocada dentro dos contêiners. Porém, em ambos os casos, o planejamento do formato de carregamento da carga é importante, dado que as estratégias que dependem do fator humano, em geral, não apresentam um bom desempenho.

Além disso, garantir um melhor aproveitamento do espaço dentro dos contêiners pode impactar significativamente na redução dos custos totais envolvidos no processo de carregamento. A escolha do tipo de contêiner utilizado também pode ser importante, pois a qualidade pode variar de acordo com o tipo de produto a ser carregado.

Assim como descrito no trabalho de Utida [5], os contêiners mais utilizados são os de 20 pés (padronizada em: 5,94m de comprimento, 2,37m de largura e 2,28m de altura), em geral destinados às cargas mais pesadas e menos volumosas, e os de 40 pés (padronizada em: 12,04m de comprimento, 2,32m de largura e 2,38m de altura), em geral, usados para cargas mais leves e mais volumosas.

Utida [5] também descreve alguns tipos de contêiners, dentre os vários tipos existentes, como sendo os mais utilizados:

- *Dry Box* : contêiner totalmente fechado, com portas somente nos fundos, que é adequado para transportar cargas secas, tais como roupas, móveis, calçados, etc.

- *Open Top*: contêiner totalmente fechado, com aberturas no teto, usado para transporte de cargas, tais como produtos agrícolas.

- *Flat Rack* : contêiner sem as paredes laterais e sem teto, utilizado para cargas grandes e pesadas.

- *Reefer*: contêiner totalmente fechado, com portas no fundo, adequado para cargas que necessitam de controle de temperatura.

- *Contêiners Modulares*: contêiners que servem como escritórios, dormitórios, guaritas, ambulatórios, sanitários, depósitos, estandes para feiras e eventos, etc. Os tamanhos variam de 2,00m a 6,00m de comprimento por 2,25m de largura e 2,50m de altura.

Várias categorizações e especificidades podem ser atribuídas ao PCC, tais como: tipos das caixas, números de contêiners, posicionamento e orientação das caixas, centro de gravidade, limites de peso ou de dimensões, tipos de atribuição, dentre outros. As combinações de restrições estabelecidas formam subproblemas para o PCC, que serão posteriormente descritos na Seção 2.1.

1.3 OBJETIVOS

1.3.1 Objetivo geral

O objetivo deste trabalho é apresentar um método de resolução que resolve uma das categorias mais complexas do Problema de Carregamento de Contêiners: o *Problema da Embalagem em Objetos de um Único Tamanho*, em três dimensões. Este problema visa promover o carregamento de todas as caixas que compõem uma carga e que têm diferentes dimensões, dentro de contêiners idênticos, e de tal forma que as caixas possam ser rotacionadas em todas as direções. Em muitos trabalhos, esse problema é conhecido simplesmente por *Problema da Embalagem* (BPP, do inglês *Bin Packing Problem*), cujas caixas podem possuir sortimento fraca ou fortemente heterogêneo. Para o caso tridimensional, frequentemente é utilizada a sigla 3D-BPP. Uma vez que métodos exatos não são eficientes para resolver estes problemas, principalmente quando se considera uma realidade de grande porte, a pesquisa teve como objeto de estudo o desenvolvimento de uma meta-heurística que pudesse fornecer boas soluções.

1.3.2 Objetivos específicos

- Realizar uma revisão bibliográfica sobre o estado da arte de Problemas de Carregamento de Contêiners e suas categorias de subproblemas;
- Revisar na literatura os estudos mais recentes sobre meta-heurísticas que determinam boas aproximações para o *Problema da Embalagem em Objetos de um Único Tamanho*;
- Implementar o melhor algoritmo atual que resolve este problema, realizando os primeiros testes computacionais, com o intuito de se efetuar futuras comparações;
- Desenvolver e implementar uma nova meta-heurística fundamentada em algoritmos genéticos e com a introdução de novas estratégias (a qual foi chamada, posteriormente, de Algoritmo Genético Evolucionário Especializado – AGE-E);

- Realizar testes computacionais que possam verificar a qualidade das soluções obtidas, comparando-as com as apresentadas pelo melhor algoritmo atual encontrado na literatura.

1.4 METODOLOGIA

Inicialmente, foi realizado o levantamento bibliográfico sobre o estado da arte dos Problemas de Carregamento de Contêiners e sobre os estudos mais recentes de meta-heurísticas que apresentaram os melhores resultados para o *Problema da Embalagem em Objetos de um Único Tamanho*. Após concluída esta etapa, decidiu-se optar pelo estudo de Algoritmos Genéticos como a principal meta-heurística norteadora dos estudos posteriores. Para isso, outros algoritmos genéticos foram estudados, contendo novas estratégias, e que ainda não foram aplicados a problemas de carregamento, mas sim a outros problemas da Pesquisa Operacional.

Também foi realizada a implementação computacional da melhor meta-heurística que resolve o *Problema da Embalagem em Objetos de Tamanho Único*, a saber: Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA, do inglês: *Biased Random-Key Genetic Algorithm*) proposto por Resende e Gonçalves, em 2012, em seu trabalho [12]. Testes computacionais iniciais foram aplicados em uma das classes de problemas contida neste artigo, visando a verificação de compatibilidade entre a implementaçao realizada no presente trabalho e a apresentada em [12].

Após, foi desenvolvida uma nova meta-heurística, chamada de AGE-E, que tem como fundamento o embasamento teórico adquirido no decorrer da trajetória de estudos e, em seguida, foi realizada sua implementação. Testes computacionais foram aplicados sobre as classes de problemas do artigo [12], bem como uma comparação entre os resultados obtidos com a nova meta-heurística e com aqueles obtidos por Resende e Gonçalves, completando a sequência de objetivos propostos a serem alcançados.

Os principais trabalhos que nortearam as pesquisas foram:

- *A biased random-key genetic algorithm for a 2D and 3D bin packing problem*, trabalho de Maurício G. C. Resende e José Fernando Gonçalves, 2012, [12].
- *A typology of cutting and packing problems*, trabalho de Harald Dyckhoff, 1990, [13].
- *An improved typology of cutting and packing problems*, trabalho de Gerhard Wäscher, Heike Haubner e Holger Schumann, 2007, [14].
- *Constraints in container loading problem - A state-of-the-art review*, trabalho de

Andreas Bortfeldt e Gerhard Wäscher, 2013, [15].

- *Heuristics for the container loading problem*, trabalho de David Pisinger, 2002, [16].
- *A parallel multi-population biased random-key genetic algorithm for a container loading problem*, trabalho de José Fernando Gonçalves e Maurício G. C. Resende, 2012, [17].
- *Biased random-key genetic algorithms for combinatorial optimization*, trabalho de Maurício G. C. Resende e José Fernando Gonçalves, 2011, [18].
- *Finding multiple roots of a box-constrained system of nonlinear equations with a biased random-key genetic algorithm*, trabalho de Maurício G. C. Resende e Ricardo Martins de Abreu Silva, 2013, [19].
- *Developing a simulated annealing algorithm for the cutting stock problem*, trabalho de Kin Keung Lai e Jimmy W. M. Chan, 1997, [20].
- *A genetic algorithm for the generalised assigment problem*, trabalho de P. C. Chu e J. E. Beasley, 1997, [21].

1.5 ORGANIZAÇÃO DO TRABALHO

No Capítulo 2 apresenta-se o levantamento bibliográfico do estado da arte dos Problemas de Carregamento de Contêiners e, em particular, categorizando o problema estudado neste trabalho. Além da descrição das categorias do PCC, também são destacadas algumas especificidades quanto ao tipo de carga, formas de carregamento, orientação e posicionamento dos itens e principais métodos de resolução encontrados na literatura.

No Capítulo 3, definem-se algoritmos genéticos de maneira geral, e, realiza-se o detalhamento de dois algoritmos genéticos de interesse para o presente trabalho: o BRKGA, de Resende e Gonçalves [12], e o Algoritmo Genético de Chu-Beasley, de Chu e Beasley [21]. Esta atenção dada às duas variantes de algoritmo genético é justificada pelo fato de que o BRKGA é a meta-heurística mais eficiente da literatura que resolve o problema estudado, e que o Algoritmo Genético de Chu-Beasley tem mostrado resultados eficientes na resolução de problemas de atribuição generalizada, e que ainda não foi explorado em problemas de carregamento.

O Capítulo 4 contém os dados e detalhamento dos primeiros testes computacionais realizados, referentes ao BRKGA, para a verificação da compatibilidade dos resultados, de acordo

com o que se obteve no trabalho de Resende e Gonçalves [12]. E, no Capítulo 5 descreve-se a nova meta-heurística AGE-E e sua estrutura teórica, apresenta o detalhamento da implementação e resultados computacionais e os compara com os apresentados em [12].

Finalmente, no Capítulo 6, o trabalho é concluído, apresentando-se uma breve descrição do que se pretende realizar após a conclusão desta tese de doutorado.

2 PRESSUPOSTOS TEÓRICOS

Neste capítulo, apresenta-se uma revisão bibliográfica sobre os principais temas que formam o arcabouço de conhecimentos necessários ao entendimento da meta-heurística desenvolvida: o Algoritmo Genético Evolucionário Especializado (AGE-E). Uma revisão sobre Problemas de Carregamento de Contêiners e principais métodos de solução e, em particular, sobre aqueles que resolvem o *Problema da Embalagem em Objetos de Tamanho Único*, compõem o conteúdo deste capítulo.

2.1 O PROBLEMA DE CARREGAMENTO DE CONTÊINERS

O Problema de Carregamento de Contêiners (PCC) foi classificado pela primeira vez na literatura como uma das categorias dos Problemas de Corte e Empacotamento (PCE), de acordo com a tipologia de Dyckhoff [13], em 1990. Esta tipologia foi, posteriormente, aprimorada por Wäscher et al. [14], em 2007. Um maior detalhamento sobre a categorização dos PCC foi ainda descrito e apresentado por Bortfeldt e Wäscher, em 2013, em seu trabalho [15].

A classificação como um PCE deriva do fato de que, segundo Martínez [22], o PCC verifica as seguintes condições comuns ao problema geral de corte e empacotamento: (1) existem dois grupos distintos de elementos - objetos grandes e itens pequenos; (2) o objetivo é selecionar alguns ou todos os itens, para atribui-los e localizá-los dentro de um conjunto de objetos (definição do padrão de corte) de forma a otimizar a função objetivo; e (3) objetos e itens devem ser ambos definidos geometricamente, de acordo com as dimensões exigidas, e de tal forma que cada item esteja totalmente inserido dentro do objeto e sem sobreposição entre os itens.

2.1.1 Problemas de corte e empacotamento

De acordo com Dyckhoff [13], vários são os problemas que podem ser categorizados como um problema de corte e empacotamento, e que preservam as três condições comuns supracitadas: problemas de carregamento, problemas da mochila, problemas de particionamento, problemas de atribuição, dentre outros. A distribuição destes problemas em categorias e subcategorias obedece uma classificação que considera os cinco critérios descritos abaixo:

1. *Dimensionalidade*:

a) unidimensional;

b) bidimensional;
c) tridimensional;
d) N-dimensional, N>3.

2. *Tipos de Atribuição*:
a) Maximização do valor de saída: Todos os objetos são utilizados e há a seleção de itens, logo seleciona-se um conjunto de itens que irá maximizar a função objetivo;
b) Minimização de entrada: Há a seleção de objetos e todos os itens são utilizados, logo o objetivo é minimizar o número de objetos grandes.

3. *Sortimento de itens pequenos*:
a) Sortimento homogêneo de itens: os itens são idênticos (em forma e tamanho);
b) Sortimento fracamente heterogêneo: os itens podem ser agrupados em algumas classes de itens idênticos;
c) Sortimento fortemente heterogêneo: poucos itens podem ser idênticos.

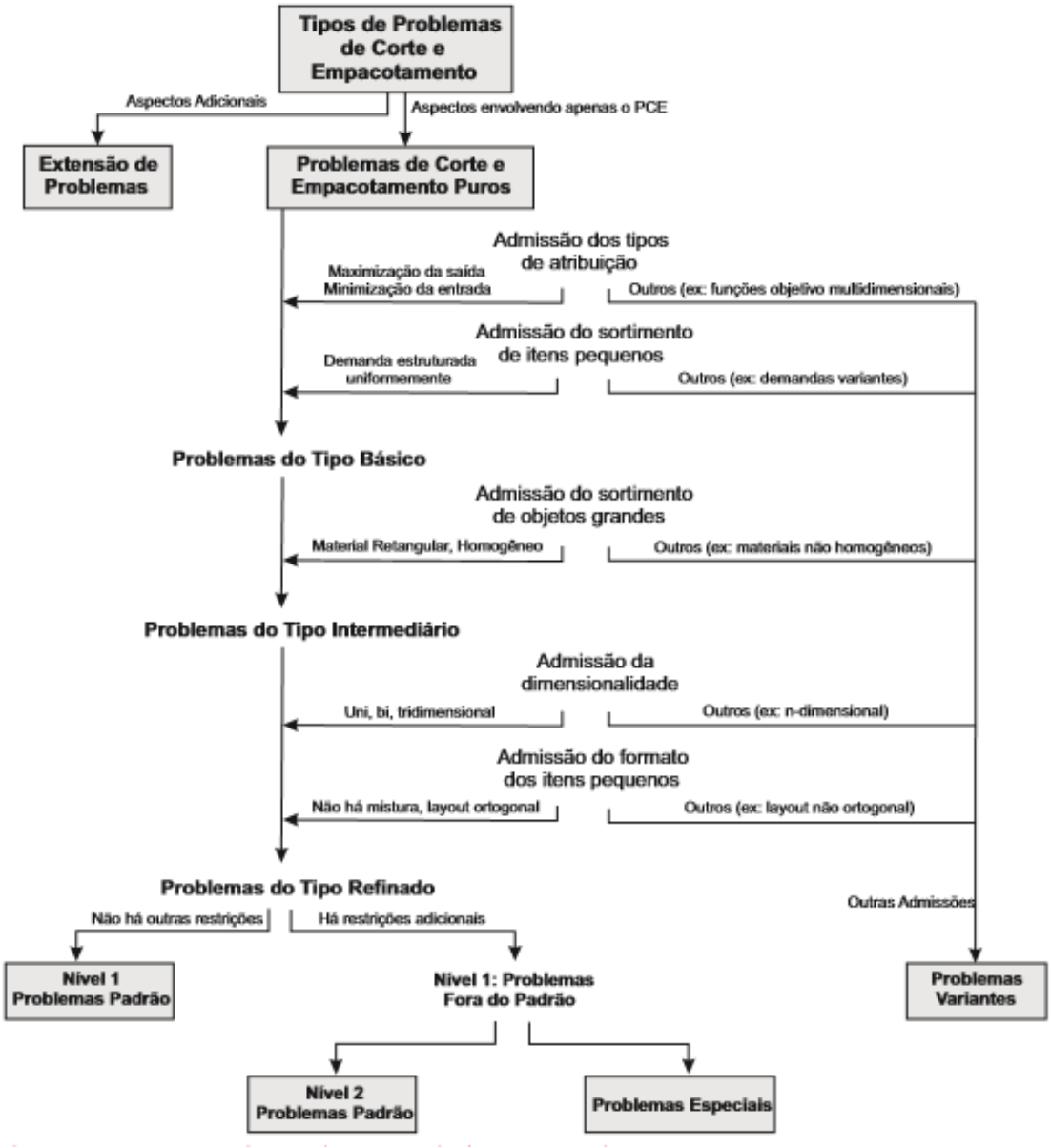
4. *Sortimento de objetos grandes*:
a) Há apenas um único objeto;
b) Há muitos objetos: analogamente ao sortimento de itens pequenos, o sortimento de múltiplos objetos pode ser classificado em homogêneo, fracamente heterogêneo ou fortemente heterogêneo.

5. Formato dos itens pequenos:
a) Os itens são regulares: retângulos, círculos, caixas, cilindros, dentre outros;
b) Os itens são irregulares.

As classificações de Dyckhoff [13] para os problemas, a partir destes critérios, foram substituídas (melhor arranjadas) por Wäscher et al. [14], que consideraram uma divisão dos PCE em dois grandes grupos, e de acordo com os cinco critérios de Dyckhoff: (1) *Extensão de Problemas* - problemas que fazem correlação não apenas com os PCE; e (2) *Problemas de Corte e Empacotamento Puros* (PCEP), que são estruturados relacionando apenas as estratégias de PCE. Os autores ainda subdividem os PCE Puros em três subgrupos, os quais foram chamados de *Problemas do Tipo Básico, Problemas do Tipo Intermediário e Problemas do Tipo Refinados*. Na Figura 2.1, apresenta-se essa divisão e subdivisão descritas, bem como a classificação geral de

acordo com os cinco critérios apresentados previamente. É possível observar na figura que aos PCEP vão sendo atribuídos os cinco critérios até que problemas do tipo *Básico* transformam-se em do tipo *Intermediário* e finalizam como problemas do tipo *Refinado*, afunilando à medida que mais critérios vão sendo verificados.

Figura 2.1 - Visão geral dos tipos de Problema de Corte e Empacotamento.

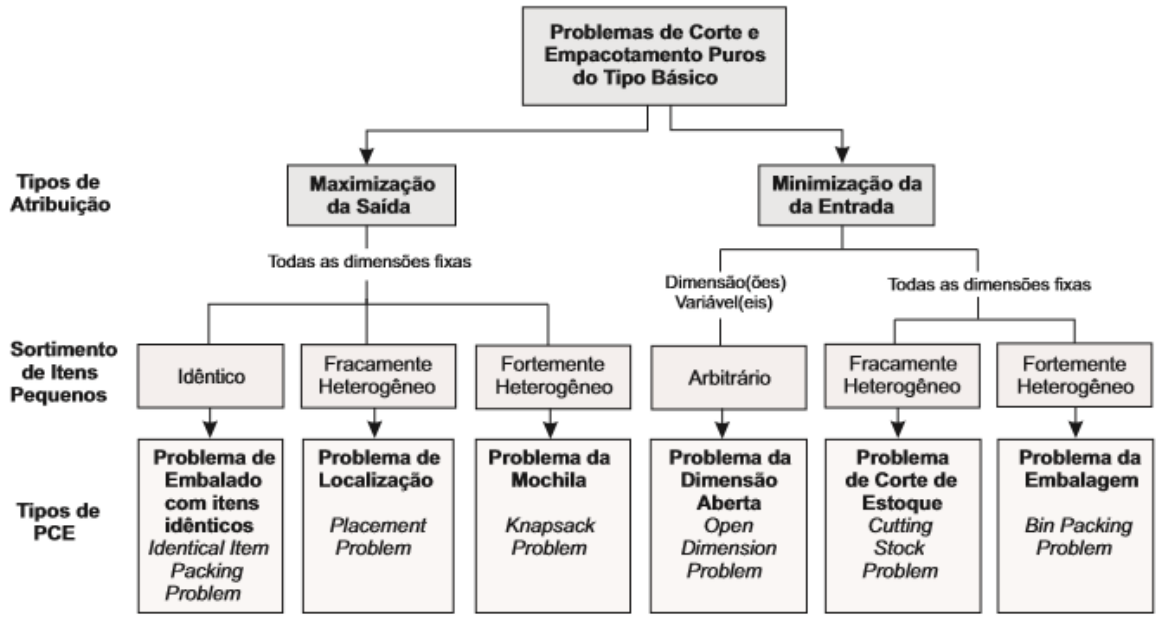


Fonte: Adaptado de Wäscher et al.(2007) [14].

Vários problemas da literatura são classificados dentro dessa evolução de admissão de características particulares. Inicialmente, os autores classificam alguns problemas como sendo do tipo *Básicos* (definidos como Problemas de Corte e Empacotamento Puros que admitem os critérios 2

e 3 -*Tipos de Atribuição* e *Sortimento de Itens Pequenos*, respectivamente -), tal como apresentado na Figura 2.2. Nestes, é realizada uma divisão entre problemas que consideram a maximização da saída como forma de atribuição ou como minimização da entrada. Em seguida, cada uma destas duas categorias de atribuição ramificam-se em três subcategorias de problemas, com relação ao sortimento (variedadedos itens pequenos, formando a classificação de seis problemas da literatura.

Figura 2.2 - Tipos *Básicos* dos Problemas de Corte e Empacotamento



Fonte: Adaptado de Wäscher et al. (2007) [14].

Os três primeiros problemas *Básicos* da Figura 2.2 são ramificações provenientes da categoria *Maximização da Saída*: *Problema da Embalagem com Itens Idênticos*, *Problema de Localização* e *Problema da Mochila*. O objetivo é designar o maior número possível de itens pequenos a um número limitado de objetos grandes. O que diferencia os três problemas é o tipo de itens pequenos que os compõem: itens idênticos, fracamente heterogêneos ou fortemente heterogêneos, respectivamente. Problemas que consideram itens idênticos também são chamados na literatura como do tipo *homogêneo* (*carga homogênea*).

Os três últimos problemas *Básicos* são subcategorias da *Minimização da Entrada*: *Problema da Dimensão Aberta, Problema de Corte de Estoque* e *Problema da Emba-lagem,* nos quais deseja-se alocar todos os pequenos itens a um número ilimitado de objetos grandes, de forma que toda a demanda de itens seja satisfeita. A diferença entre estes problemas é que no primeiro, as dimensões dos objetos grandes são variáveis (indeterminadas), e nos dois últimos, os

itens devem ser do tipo fraca ou fortemente heterogêneos, respectivamente, cujos objetos possuem dimensões fixas.

Os Problemas do Tipo Intermediários são formados ao subdividir cada um dos seis problemas *Básicos*, de acordo com o critério 4 - *Sortimento de Objetos Grandes*. As Figuras 2.3 e 2.4 apresentam os problemas da literatura que são classificados quanto a essa nova categorização. Para finalizar, os problemas do tipo *Refinados* são determinados ao acrescentar-se os critérios de *1 - Dimensionalidade* e *5 - Formato dos itens pequenos* a cada um dos problemas intermediários, adicionando antes dos nomes dispostos nas Figuras 2.3 e 2.4, as especificações: {1, 2, 3}-D ou {*retangular, circular, ...*}, respectivamente.

Figura 2.3 - Tipos *Intermediários* dos PCE - Tipo *Maximização da Saída.*

Sortimento de itens pequenos / Sortimento de objetos grandes		Itens idênticos	Itens fracamente heterogêneos	Itens fortemente heterogêneos
Todas as dimensões fixas	Objetos idênticos	**Problema do acondicionamento itens idênticos** *Identical Item Packing Problem* IIPP	**Problema de localização num único objeto** *Single Large Object Placement Problem* SLOPP	**Problema de uma única mochila** *Single Knapsack Problem* SKP
	Objetos fracamente heterogêneos		**Problema de localização em múltiplos objetos idênticos** *Multiple Identical Large Object Placement Problem* MILOPP	**Problema das múltiplas mochilas idênticas** *Multiple Identical Knapsack Problem* MIKP
	Objetos fortemente heterogêneos		**Problema de localização em múltiplos objetos heterogêneos** *Multiple Heterogeneous Large Object Placement Problem* MHLOPP	**Problema das múltiplas mochilas heterogêneas** *Multiple Heterogeneous Knapsack Problem* MHKP

Fonte: Adaptado de Wäscher et al. (2007) [14].

Assim como especificado no esquema da Figura 2.1, desde que não se verifique outras restrições ou particularidades dos cinco critérios, aos problemas descritos anteriormente são denotados por *Problemas Padrão - Nível 1*, que formam a maioria dos problemas de Corte e Empacotamento.

Outras características que se podem atribuir aos PCE formam outros tipos de problema, chamados de *Problemas Fora do Padrão* ou *Problemas Variantes*. Maior detalhamento sobre estes problemas pode ser encontrado no trabalho de Wäscher et al. [14].

2.1.2 Problemas de carregamento de contêiners como problemas de corte e empacotamento

O trabalho de Bortfeldt e Wäscher [15], de 2013, apresenta classificações para o Problema de Carregamento de Contêiners, segundo as tipologias de Wäscher et al. [14], ou seja, considerando todos os tipos de problemas de carregamento como problemas de corte e empacotamento. Bortfeldt e Wäscher apresentam uma descrição detalhada sobre como cada um dos problemas do tipo *Intermediários* de Wäscher et al. podem ser enquadrados como um PCC. Essas classificações são definidas da seguinte forma:

Figura 2.4 - Tipos *Intermediários* dos PCE - Tipo *Minimização da Saída.*

<table>
<tr><td colspan="2">Sortimento de itens pequenos
Sortimento de objetos grandes</td><td>Itens fracamente heterogêneos</td><td>Itens fortemente heterogêneos</td></tr>
<tr><td rowspan="3">Todas as dimensões fixas</td><td>Objetos idênticos</td><td>Problema de corte de estoque em objetos de tamanhos idênticos
Single Stock Size Cutting Stock Problem
SSSCSP</td><td>Problema da embalagem em objetos de um único tamanho
Single Bin Size Bin Packing Problem
SBSBPP</td></tr>
<tr><td>Objetos fracamente heterogêneos</td><td>Problema de corte de estoque em múltiplos objetos
Multiple Stock Size Cutting Stock Problem
MSSCSP</td><td>Problema da embalagem em objetos de múltiplos tamanhos
Multiple Bin Size Bin Packing Problem
MBSBPP</td></tr>
<tr><td>Objetos fortemente heterogêneos</td><td>Problema residual de corte de estoque
Residual Cutting Stock Problem
RCSP</td><td>Problema residual da embalagem
Residual Bin Packing Problem
RBPP</td></tr>
<tr><td colspan="2">Um único objeto com dimensões variáveis</td><td colspan="2">Problema de dimensão aberta
Open Dimension Problem
ODP</td></tr>
</table>

Fonte: Adaptado de Wäscher et al. (2007) [14].

1. **Problemas intermediários do tipo *Maximização da Saída* aplicados ao PCC (apenas um conjunto de caixas é carregado)**:

(1.1) IIPP: Carregamento de um único contêiner com um número máximo de caixas idênticas;

(1.2) SLOPP: Carregamento de um único contêiner com a seleção de um conjunto de caixas fracamente heterogêneas e no qual o valor dos itens carregados é maximizado;

(1.3) MILOPP: Carregamento de um conjunto de contêiners idênticos com a seleção de um conjunto de caixas fracamente heterogêneas e no qual o valor dos itens carregados é maximizado;

(1.4) MHLOPP: Carregamento de um conjunto de contêiners fraca ou fortemente heterogêneos com a seleção de um conjunto de caixas fracamente heterogêneas e no qual o valor dos itens carregados é maximizado;

(1.5) SKP: Carregamento de um único contêiner com a seleção de um conjunto de caixas fortemente heterogêneas e no qual o valor dos itens carregados é maximizado; (1.6) MIKP: Carregamento de um conjunto de contêiners idênticos com a seleção de um conjunto de caixas fortemente heterogêneas e no qual o valor dos itens carregados é maximizado;

(1.7) MHKP: Carregamento de um conjunto de contêiners fracamente ou fortemente heterogêneos com a seleção de um conjunto de caixas fortemente heterogêneas e no qual o valor dos itens carregados é maximizado.

2. **O PCC como um problema *Intermediário* do tipo *Minimização da Entrada* (todas as caixas são carregadas)**:

(2.1) SSSCSP: Carregamento de caixas fracamente heterogêneas dentro de um número mínimo de contêiners idênticos;

(2.2) MSSCSP: Carregamento de caixas fracamente heterogêneas dentro de contêiners com sortimento fracamente heterogêneo e cuja função objetivo é de minimizar os valores dos contêiners;

(2.3) RCSP: Carregamento de caixas fracamente heterogêneas dentro de contêiners com sortimento fortemente heterogêneo e cuja função objetivo é de minimizar os valores dos contêiners;

(2.4) SBSBPP: Carregamento de caixas fortemente heterogêneas dentro de um número mínimo de contêiners idênticos;

(2.5) MBSBPP: Carregamento de caixas fortemente heterogêneas dentro de contêiners com sortimento fracamente heterogêneo e cuja função objetivo é de minimizar os valores

dos contêiners;

(2.6) RBPP: Carregamento de caixas fortemente heterogêneas dentro de contêiners com sortimento fortemente heterogêneo e cuja função objetivo é de minimizar os valores dos contêiners;

(2.7) ODP: Carregamento de caixas dentro de um único contêiner com uma mais dimensões variáveis e no qual o volume do contêiner é minimizado.

É importante ressaltar que, mesmo com a divisão dos PCC em 14 categorias distintas (1.1 a 1.7 e 2.1 a 2.7), as condições genéricas dos problemas de corte e empacotamento ainda são preservadas quando aplicadas especificamente aos PCC, através de uma correspondência paralela. Isto quer dizer que os Problemas de Carregamento de Contêiners: (1) possuem dois grupos distintos de elementos, os objetos grandes que são os contêiners, e os itens pequenos que são as caixas (carga) que serão carregadas dentro dos contêiners; (2) o objetivo é selecionar algumas ou todas as caixas a serem alocadas dentro de um ou mais contêiners e de forma a otimizar a função objetivo; e (3) os contêiners e caixas são definidos geometricamente por paralelepípedos (para o caso tridimensional) ou retângulos (para o caso bidimensional), em que cada caixa carregada deve estar completamente inserida dentro do(s) contêiner(s) e caixas carregadas não podem sobrepor-se.

A palavra *caixa* é utilizada, na maioria dos trabalhos da literatura, ao invés de *itens pequenos*, com exceção apenas quando os itens pequenos são diferentes de retângulo ou paralelepípedo, para os casos bidimensional ou tridimensional, respectivamente. O mesmo acontece com a palavra *contêiner* no lugar de *objetos grandes*, ou, analogamente, com as palavras *caminhão*, *trem* ou *palete*, dentre outros. Considera-se também, em quase todos os trabalhos relacionados aos PCC, que o carregamento das caixas é ortogonal, i.e, no qual as superfícies (os lados) das caixas devem estar alinhadas paralelamente às superfícies (aos lados) do contêiner.

Devido à grande aplicabilidade do uso de contêiners nos problemas de transporte da logística industrial, muitos pesquisadores da Pesquisa Operacional têm direcionado seus esforços na descrição de problemas reais a partir de Problemas de Carregamento de Contêiners e, principalmente, no aprofundamento de técnicas e de métodos matemáticos que apresentem boas soluções para esses problemas. Os estudos têm aumentado significativamente a cada década (como é possível observar na Tabela 2.1), justificando a necessidade da categorização dos PCC que sistematize e organize os tipos existentes, para a proposição de formas de resolução para cada um deles.

Tabela 2.1 - Número de publicações sobre problemas de carregamento de contêiners.

Décadas anteriores	Intervalo de Tempo	Número de Trabalhos	Total de Publicações
3	1980-1984	3	7
	1985-1989	4	
2	1990-1994	21	47
	1995-1999	26	
1	2000-2004	29	104
	2005-2011	75	

Fonte: Adaptado de Bortfeldt e Wäscher (2007) [14].

2.2 PRINCIPAIS CARACTERÍSTICAS DOS PCC

Na literatura, algumas características particulares vem sendo atribuídas às várias categorias de PCC, na tentativa de ajustar à teoria as particularidades reais dos problemas de carregamento de contêiners. O trabalho de Bortfeldt e Wäscher [15] também apresenta uma descrição destas características comumente encontradas nas principais publicações relacionadas ao tema, e que foram nomeadas pelos autores como *Restrições*. Uma síntese destas restrições adicionais é apresentada a seguir.

2.2.1 Restrições relacionadas ao container

- **Restrições de Limite de Peso**: Em torno de 14% dos problemas da literatura incluem restrições de limite de peso não excedido às cargas dentro dos contêiners. Em geral, estas restrições podem ser modeladas linearmente através de *Problemas da Mochila*, em que a somatória dos pesos dos itens carregados não pode fornecer um valor superior ao peso limite permitido no contêiner. Pode-se citar os seguintes trabalhos que apresentam tais considerações: Liu e Chan [23]; Gehring e Bortfeldt [24]; Terno et al. [25]; Chan et al. [26]; Egeblad et al. [27]; e Lui et al. [28].
- **Restrições de Distribuição de Peso**: Estas restrições tratam problemas de carregamento de contêiners em que o peso da carga carregada deve ser distribuída de maneira mais uniforme possível dentro do contêiner (carregamento balanceado). De acordo com Bortfeldt e Wäscher [15], estima-se que 12% dos PCC encontrados na literatura impõem este tipo de restrição, tal como pode ser encontrado nos trabalhos de: Chan et al. [26], Liu et al. [28], Sommerweib [29] e Balakirsky et al. [30].

2.2.2 Restrições relacionadas aos itens

- **Restrições de Prioridade no Carregamento**: Estas restrições são incorporadas a problemas do tipo *Maximização da Saída*, em que não é possível carregar todos o itens e deve-se decidir qual subconjunto selecionar. Várias são as regras de prioridade que podem ser definidas (menor volume, maior valor financeiro, menor custo, etc), e, nos trabalhos da literatura, realiza-se ainda uma subdivisão em regras de prioridade chamadas *inferiores* e *superiores*. Porém, pouca bibliografia é atribuída a esse tipo de restrição, cerca de 1%, no qual destacam-se os trabalhos de Bortfeldt e Gehring [31] e de Ren et al. [32].

- **Restrições de Orientação dos Itens**: A maioria dos trabalhos publicados à respeito de Problemas de Carregamento de Contêiners aborda questões quanto à orientação dos itens carregados (caixas), em torno de 71%. Existem seis possibilidades de rotação ortogonal de uma caixa tridimensional e duas para caixas bidimensionais. Muitas vezes, restrições são impostas a estas rotações para evitar prejuízos da mercadoria dentro das caixas. Bortfeldt e Wäscher [15] separam na literatura, cinco casos de restrições de orientação dos itens para o caso tridimensional:

  Caso 1: As caixas não podem ser rotacionadas, ou seja, só é permitida uma orientação - a orientação original da caixa. Exemplos: Martello et al. [9], Morabito e Arenales [33] e Junqueira et al. [34].

  Caso 2: As caixas podem sofrer apenas uma rotação ortogonal, desde que uma al tura fixa seja mantida. Exemplos: Chien e Deng [35], Fuellerer et al. [36] e Iori e Martello [37].

  Caso 3: As caixas podem ser rotacionadas em apenas 90 graus para qualquer orien tação, uma única vez, incluindo portanto o caso 2. Exemplos: Bischoff et al. [38], Parreño et al. [39] e He e Huang [40].

  Caso 4: Não há uma regra específica para a orientação das caixas. Porém, algumas restrições em particular - dentre as outras cinco possíveis, excluindo a orientação original -, pode ser atribuída a cada caixa (ou tipo de caixa). Exemplos: Liu et al. [28], Ren et al. [32] e Fanslau e Bortfeldt [41].

Caso 5: Não há restrições de orientação, ie, as caixas podem ser carregadas em qual quer uma das seis posibilidades. Exemplos: Wang et al. [6], Faina [42] e Padberg [43].

- **Restrições de Arranjo do Carregamento**: Aproximadamente 15% dos trabalhos da literatura aborda este tipo de restrições em Problemas de Carregamento de Contêiners, no qual avalia-se como organizar os arranjos entre as caixas, alocando-as umas sobre as outras, em uma disposiçao que minimize prejuízos na parte inferior da carga carregada. Existem muitas formas de considerar estas restrições e a mais comum leva em consideração o tipo de fragilidade das caixas (quanto ao peso, massa, tamanho). Muitas vezes, há inclusive a comparação entre caixas para definir as fragilidades envolvidas. É possível citar os seguintes trabalhos que se destacam neste segmento: Junqueira et al. [34], Fuellerer et al. [36], Sciomachen e Tanfani [44], Bischoff [45] e Ceschia e Schaerf [46].

2.2.3 Restrições relacionadas à carga

- **Restrições de Embarque Completo**: Para problemas de *Maximização da Saída*, onde deve-se otimizar o espaço utilizado dentro de um contêiner, itens podem não ser carregados devido à indisponibilidade de espaço dentro do contêiner. Dessa forma, estas restrições de embarque completo indicam que um subconjunto de caixas devem ser carregadas obrigatoriamente juntas. Isto quer dizer que se um item não é carregado, então nenhum outro item do subconjunto a que pertence pode ser carregado. De forma análoga, se uma caixa de um determinado subconjunto de caixas for carregada dentro do contêiner, então, obrigatoriamente, deve haver o carregamento completo de todos os outros itens do mesmo subconjunto. Estas restrições são importantes, pois muitos produtos são carregados com as peças que o compõem em separado, mas que devem fazer parte do mesmo carregamento. Entretanto, apenas 0, 6% da literatura consideram estas questões, como é o caso do trabalho de Eley [47].

- **Restrições de Alocação**: Estas restrições aparecem apenas em problemas de carregamento de múltiplos contêiners nas duas condições possíveis:

  Caso 1 (*Restrições de Conectividade*): Um mesmo conjunto de itens devem ser carregados no mesmo contêiner, em geral, quando o destino dos itens é para o mesmo local.
  Caso 2 (*Restrições de Separação*): Alguns tipos de itens não podem ser misturados com outros tipos, como, por exemplo, itens contendo comida não podem ser misturados com

itens contendo perfumes.

Ambos os casos estão contidos em 8% dos trabalhos estudados por Bortfeldt e Wäscher [15], e, em sua maioria, referem-se às Restrições de Conectividade, tais como pode ser encontrado nos trabalhos de Terno et al. [25], Liu et al. [28] e Tsai et al. [48], e em Fuellerer et al. [36] e Iori e Martello [37], envolvendo roteamento de veículos e carregamento de contêiners.

2.2.4 Restrições de posicionamento

Só há um tipo de conjunto de restrições relacionado ao posicionamento e localização de itens dentro de contêiners, divididos em dois casos:

- *Caso 1* (*Restrições de Posicionamento Absoluto*): itens devem ser carregados em uma posição ou área específica dentro do contêiner, considerando-se o tamanho, peso ou conteúdo dos itens. Em geral, 2,5% dos trabalhos da literatura admitem tais restrições, como é o caso dos trabalhos de: Egeblad et al. [27], Bortfeldt e Gehring [31] e Hodgson [49].
- *Caso 2* (*Restrições de Posicionamento Relativo*): um conjunto de itens que devem ser carregados um ao lado do outro, ou, ao contrário, com ao menos uma quantidade de distância uns dos outros. Encontradas em cerca de 4, 4% da bibliografia, é possível destacar os seguintes trabalhos: Terno et al. [25], Egeblad et al. [27] e Makarem e Haraty [50].

2.2.5 Restrições relacionadas ao carregamento

- **Restrições de Estabilidade**: Considerada uma das questões mais importantes da literatura, a estabilidade no arranjo final do carregamento dentro de um contêiner irá coibir a existência de danos na carga durante o transporte dos produtos ou no decorrer do procedimento de carregamento. A estabilidade pode ser vertical (*estabilidade estática*), evitando que itens caiam no chão do contêiner ou por cima de outros itens, ou pode ser horizontal (*estabilidade dinâmica*), impedindo que os itens movimentem-se durante o transporte da carga. Ambos os tipos podem ser aplicados à toda a carga ou de forma parcial, e em geral, estão presentes em problemas de carregamento de contêiners do tipo *Maximização de Saída*. Cerca de 37% da literatura correlata considera estas restrições em uma vasta gama de trabalhos publicados, e através de várias estratégias, tais como em:

Resende e Gonçalves [17], Liu et al. [28] e Fanslau e Bortfeldt [41], para o primeiro tipo; e Liu et al. [28], Sommerweib [29] e também Eley [47], para o segundo.

- **Restrições de Complexidade**: Tais restrições, nem sempre consideradas por todos os pesquisadores, referem-se aos casos em que as restrições impostas tornam o problema extremamente complexo de se resolver, de se visualizar as propriedades, e, principalmente, tal que o tempo computacional não permite a sua resolução. A importância em considerar estes tipos de PCC, é que, mesmo sem ainda poder apresentar aproximações para a solução, a evolução da tecnologia e a construção de computadores com maiores potencilidades podem fazer com que num futuro próximo soluções possam vir a ser apresentadas. Cerca de 9,5% dos trabalhos encontrados na literatura tratam de alguma forma tais restrições, como por exemplo, em Morabito e Arenales [33], Hifi [51] e Egeblad e Pisinger [52].

2.3 PRINCIPAIS MÉTODOS DE SOLUÇÃO DOS PCC

Problemas de Carregamento de Contêiners são particularmente considerados como problemas complexos, não apenas porque apresentam formulações matemáticas complexas, mas principalmente por possuírem alto grau de dificuldade na resolução exata destes problemas. Muitas são as possibilidades de arranjos entre as caixas, principalmente, à medida que se aumenta o número de caixas.

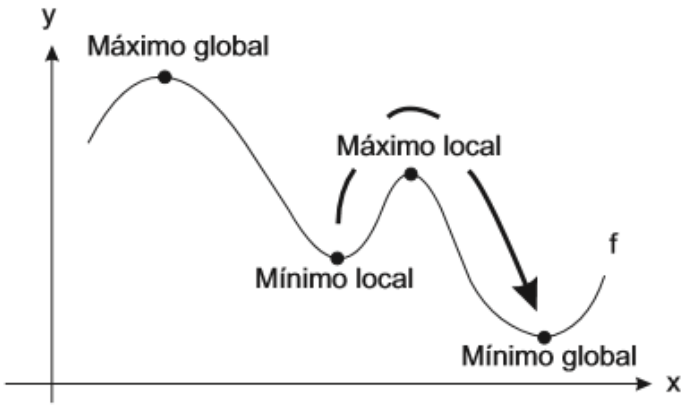
Na literatura, os PCC são classificados como NP-difíceis, contendo poucos trabalhos que apresentam algoritmos exatos para sua resolução, ou seja, algoritmos que dependem de um modelo matemático para o problema em questão, e que utiliza as características da modelagem para resolvê-lo através de um método que garante a convergência para a melhor solução possível (a solução ótima). É possível citar os trabalhos de Martello et al. [9] que apresentam dois métodos Branch and Bound (B&B) para resolver problemas do tipo SKP e SBSBPP e o de Hifi [53] que apresenta um algoritmo exato de programação dinâmica para resolver o problema SLOPP tridimensional. O trabalho de Fekete e Schepers [54] apresenta uma estrutura geral dos métodos exatos de resolução de Problemas de Carregamento de Contêiners multidimensionais.

Entretanto, em geral, apenas problemas de carregamento de pequeno porte podem ser resolvidos otimamente, contrariando a variedade de problemas complexos encontrados na vida real. Para problemas de grande porte, não somente para os PCC, os pesquisadores da Pesquisa Operacional vem desenvolvendo no decorrer das últimas décadas alternativas que permitam determinar soluções de boa qualidade, com esforço computacional compatível, mesmo que não garantam a otimalidade

das soluções. À essas alternativas foi dado o nome de *Heurísticas* e *Meta-heurísticas*.

Heurísticas são definidas como procedimentos iterativos práticos que contém um conjunto de regras e estratégias que resolvem um problema específico, não necessariamente de maneira ótima. As heurísticas foram amplamente estudadas a partir da década de 60, mas tinham a dificuldade de não conseguir superar ótimos locais, na busca por ótimos globais. Para evitar este impedimento, um novo conceito foi criado pelos estudiosos da otimização: as *Meta-heurísticas*. As *Meta-heurísticas* são heurísticas que possuem novas estratégias que promovem essa capacidade de escape, formando um algoritmo inteligente e eficiente na busca por soluções cada vez melhores. A Figura 2.5 ilustra essa característica de escape que diferencia as heurísticas das meta-heurísticas, para um problema de minimização e função objetivo *f*.

Figura 2.5 - Estratégia de escape das meta-heurísticas.



Fonte: Elaboração do próprio autor

As meta-heurísticas devem ser utilizadas quando: (i) a formulação matemática para o problema é desconhecida ou muito complexa; (ii) a solução ótima não é possível de ser determinada de maneira exata em tempo computacional eficiente; (iii) pretende-se determinar uma solução de maior qualidade como ponto de partida para o início de um algoritmo exato; ou (iv) não existe uma importância significativa na determinação de uma solução ótima ou quando os dados são imprecisos. A utilização de heurísticas, tais como as conhecidas *Algoritmo Heurístico Construtivo* e *Algoritmo Guloso*, por possuírem uma estrutura mais simples e fácil de se implementar quando comparadas às meta-heurísticas, em geral, estão associadas a estas, no sentido de fornecer um ponto de partida para as meta-heurísticas.

No que concerne, em particular, aos problemas de carregamento de contêiners, o bom desempenho de heurísticas e meta-heurísticas desenvolvidas nos últimos anos tem favorecido o aparecimento de uma crescente gama de trabalhos que consideram estes procedimentos para a sua

resolução. Vários tipos de algoritmos heurísticos e meta-heurísticos têm sido capazes de apresentar soluções de excelente qualidade em tempo computacional eficiente, porém, é importante ressaltar que nem sempre isto acontece. Muitas vezes, não é possível realizar uma análise comparativa das soluções obtidas por alguma heurística ou, ainda, dado que a amostra não possui sempre uma quantidade suficiente de experimentos numéricos que permitam classificar a qualidade dos resultados.

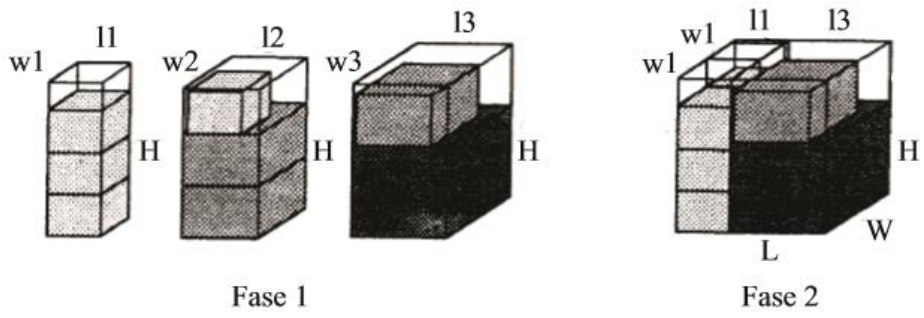
Fanslau e Bortfeldt [41] propõem uma classificação dos algoritmos da literatura que resolvem os PCC, de acordo com a meta-heurística de carregamento utilizada e pelo tipo de método, agrupando-os em cinco categorias para o primeiro caso, e em três, para o segundo, respectivamente. Quanto à *Heurística de Carregamento*, os autores destacam os cinco tipos existentes na literatura:

1. **Algoritmos de Construção em Parede** (*wall-building approaches*): Introduzidos na literatura por George e Robinson [55], este algoritmo promove o carregamento de caixas através de um número de camadas verticais ("paredes") ao longo do comprimento de um único contêiner. As espessuras destas camadas são selecionadas segundo um conjunto de regras que considera que caixas de maiores tamanhos e dimensões pequenas são aquelas que são mais difíceis de se acomodar dentro doprocesso de carregamento de caixas. Uma vez que uma espessura para a camada vertical i é definida, tiras horizontais são construídas ao longo da altura da camada (do contêiner). Para preencher as tiras horizontais, o algoritmo insere primeiramente as caixas com maior dificuldade de carregamento, definindo a altura da tira como sendo igual à menor dimensão da caixa de maior tamanho. É possível citar os seguintes trabalhos que apresentam heurísticas relacionadas a este algoritmo: Pisinger [16], Bischoff e Marriot [56], Hemmiki [57], Gehring et al. [58] e Loh e Nee [59].

2. **Algoritmos de Construção em Pilhas** (*stack-building approaches*): Proposto pela primeira vez por Gilmore e Gomory [60], este algoritmo arranja caixas em pilhas convenientes no chão do contêiner, em um procedimento que envolve duas fases. Na Fase 1, determina-se a altura H das pilhas, geralmente, pela resolução de vários problemas da mochila unidimensionais. Na Fase 2, o carregamento é tratado como um problema bidimensional (visto de cima), em que pilhas de mesmo comprimento e largura (w_i e l_i, respectivamente) seguem o mesmo padrão de carregamento tipo i, isto é: k_1 pilhas no padrão 1 (pilha de dimensões (w_1, l_1)), k_2 pilhas no padrão 2 (pilha de dimensões (w_2, l_2)), e assim, sucessivamente. A Figura 2.6 ilustra as fases do carregamento de um contêiner contendo 2 pilhas com padrão tipo 1, 0 pilhas do padrão tipo 2, e 1 pilha do

padrão tipo 3. O trabalho de Bischoff e Ratcliff [61] é um exemplo de trabalho que apresenta este algoritmo, além do trabalho de Gehring e Bortfeldt [24], que incorpora ainda um algoritmo genético a esta ideia.

Figura 2.6 - Carregamento de caixas por estacas.



Fonte: Adaptado de Bischoff e Ratcliff (1995) [61].

3. **Algoritmos de Construção em Camadas Horizontais** (*Horizontal layer-building approaches*): Similarmente aos algoritmos de construção em pilhas, estas heurísticas realizam o preenchimento de contêiners através de camadas horizontais, incluindo um processo que também se apresenta através de duas fases. Na Fase 1, camadas horizontais de tamanho (L, W) são arranjados com caixas de mesma altura em um padrão de carregamento do tipo h_i, em geral, envolvendo a resolução de problemas da mochila bidimensionais. Na Fase 2, algumas dessas camadas são escolhidas para preencher o contêiner ao longo de sua altura H, de forma a maximizar sua ocupação (em geral, utilizando-se problemas da mochila unidimensionais). Este algoritmo foi incorporado, por exemplo, aos trabalhos de Terno et al. [25] e de Bischoff et al. [38].

4. **Algoritmos de Construção em Blocos** (*Block-building approaches*): Usualmente utilizados para Problemas de Carregamento de Contêiners que possuem caixas de mesmo tipo, estes algoritmos objetivam promover o carregamento de contêiners através de blocos de caixas em formato de cubo ou paralelepípedo, este último, para o caso que existem caixas de diferentes tipos. O algoritmo considera inicialmente uma lista de possibilidades de arranjos de blocos com mesma orientação, quando as caixas são do mesmo tipo, ou, para alguns autores, contendo ainda uma segunda lista de possíveis arranjos entre caixas fortemente heterogêneas, que promovam um arranjo adequado em forma de paralelepípedo. Em seguida, algum método construtivo de carregamento é efetuado no decorrer das iterações, até que um tempo limite seja atingido. Muitos

trabalhos ainda consideram os espaços residuais dos contêiners utilizados, preenchendo-os com caixas livres (do inglês, *free boxes*), em que não haja arranjos em blocos. São exemplos de trabalho: Bortfeldt e Gehring [62], que utilizam uma busca tabu para realizar arranjos em cubo de caixas similares para o carregamento de contêiners, Eley [63], que utiliza um método de busca em árvore, e Mack et al. [64], que utilizam um método híbrido entre Simulated Annealing e Busca Tabu.

5. **Algoritmos de Cortes Guilhotinados** (*Guillotine-cutting approaches*): São algoritmos que consideram os *cortes guilhotinhados* como sendo um conjunto de restrições que garante que todas as caixas carregadas podem ser representadas por cortes em guilhotina, isto é, por cortes que são paralelos aos lados do contêiner. Algoritmos de busca em árvore através de camadas são adaptados para fornecer os padrões de corte, considerando-se os seguintes pressupostos: (i) cada árvore em camada corresponde a um particionamento guilhotinado sucessivo do contêiner em partes menores; e (ii) cada nó representa uma caixa a ser carregada. Esta ideia pode ser encontrada no algoritmo proposto no trabalho de Morabito e Arenales [33].

Quanto ao *Tipo de Método de Solução*, Fanslau e Bortfeld [41] categorizam da seguinte forma os procedimentos que resolvem Problemas de Carregamento de Contêiners:

1. **Meta-heurísticas**: Assim como previamente descrito, tem-se como meta-heurísticas mais conhecidas: **(a)** Busca Tabu (TS - *Tabu Search*) de Bortfeldt [65], **(b)** *Simulated Annealing* (SA) de Mack et al. [64], **(c)** Algoritmos Genéticos (GA - *Genetic Algorithms*) de Gehring e Bortfeldt [24] e [66], e de Bortfeldt e Gehring [67], e **(d)** *Greedy randomized adaptive search procedures* (GRASP), de Moura e Oliveira [68] e Parreño et al. [39].
2. **Métodos de Busca em Árvore ou Métodos de Busca em Grafos**: São algoritmos que dividem o problema inicial em problemas menores, excluindo restrições de integralidade, até a resolução completa do problema tratado. A aplicação destes métodos nos PCC pode ser encontrada nos trabalhos de Pisinger [27], Morabito e Arenales [33], Fanslau e Bortfeldt [41], Hifi [51] e Eley [63].
3. **Heurísticas Convencionais que incorporam métodos de construção e melhoramento**: Também descritas previamente, algumas heurísticas aplicadas aos problemas de carregamento de contêiners podem ser encontradas nos trabalhos de

Bischoff et al. [38], Bischoff e Ratcliff [61] e de Lim et al. [69].

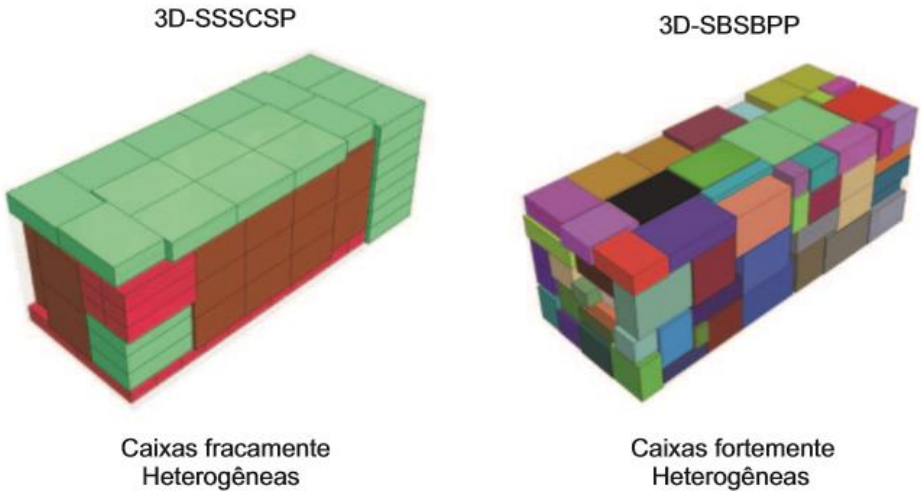
2.4 DEFINIÇÃO DETALHADA DOS PROBLEMAS ESTUDADOS

No presente trabalho, propõe-se uma nova meta-heurística, o Algoritmo Genético Evolucionário Especializado (AGE-E), como método de solução para resolver a categoria (2.4) dos Problemas de Carregamento de Contêiners, dentre as 14 listadas na subseção 2.1.2: o *3D-SBSBPP - Problema da Embalagem em Objetos de um Único Tamanho* (*Single Bin Size Bin Packing Problem*), com caixas fortemente heterogêneas e considerando, em particular, o caso tridimensional.

O objetivo neste problema é utilizar o menor número de contêiners idênticos possíveis, de dimensões fixas e iguais a (D,W,H), para carregar todas as n caixas fortemente heterogêneas do conjunto $N = \{1,2, \ldots,n\}$. Cada caixa $i \in N$ possui dimensões variadas (d_i,w_i,h_i), $i = 1,...,n$, e seu volume é dado por v_i. Na Figura 2.7, ilustram-se dois problemas semelhantes, o 3D-SSSCSP e o 3D-SBSBPP, os quais se diferenciam apenas de acordo com o tipo de sortimento das caixas: fracamente heterogêneo ou fortemente heterogêneo, respectivamente. Em muitos trabalhos, esta diferenciação é ignorada, e ambos os problemas são nomeados apenas por 3D-BPP (Problema da Embalagem Tridimensional). A partir deste momento, este trabalho irá considerar nomear apenas como 3D-BPP o problema objeto de estudo, 3D-SBSBPP, pois entende-se que problemas do tipo SSSCSP estão contidos dentro de problemas do tipo SBSBPP.

Adicionalmente, considera-se que as caixas podem ser carregadas em sua posição original, com os lados paralelos aos lados do contêiner, ou podem ser rotacionadas em 90 graus, em qualquer uma das outras cinco possibilidades (Restrição de Orientação dos Itens - Subseção 2.2.2, Caso 5). Considera-se também que devem existir restrições quanto à estabilidade da carga (Subseção 2.2.5), de tal forma que o canto inferior esquerdo de caixas carregadas devem estar diretamente alocados sobre o chão do contêiner ou sobre outras caixas já carregadas. Sem perda de generalidade, admite-se que os dados possuem valores inteiros positivos, tais que $d_i < D$, $w_i < W$ e $h_i < H$, para $i = 1, ..., n$.

Figura 2.7 - Contêiners carregados com caixas fraca/fortemente heterogêneas.



Fonte: Adaptado de Resende e Gonçalves (2002) [17].

2.5 PRINCIPAIS MÉTODOS DE SOLUÇÃO PARA O 3D-BPP

Trabalhos recentes da literatura têm apresentado o desenvolvimento de métodos de solução específicos para o 3D-BPP, nos quais as soluções obtidas demonstram superioridade nos resultados encontrados, tanto no que concerne à qualidade das soluções, quanto com relação ao tempo computacional. Dentre estes estudos, destacam-se os artigos de Martello et al. [9] e os de Resende e Gonçalves, [17] e [12]. O trabalho de Martello et al. [9], por exemplo, apresenta um método exato, baseado em um procedimeto *Branch and Bound* de dois níveis, enquanto que os de Resende e Gonçalves, [17] e [12], apresentam algoritmos genéticos modificados com uma aleatoriedade menos arbitrária (*viciada*) para a evolução das populações.

Também é possível citar outros trabalhos, tais como os de Fekete e Schepers, [54] e [70], que definem uma representação implícita para o carregamento através de *Intervalos de Grafos* (IG), a qual foi chamada de *Representação Clássica de Carregamento*. Nesta representação, os autores consideram a posição relativa das caixas em um carregamento factível e, da projeção dos itens em cada eixo ortogonal, definem um grafo associado, descrevendo então a sobreposição de itens dentro dos contêiners. Para a proposta de resolução exata, os autores utilizam uma nova classe de limitantes inferiores, propostos em [54], e fazem uma extensão do uso de funções factíveis duais, estas, utilizadas pela primeira vez no trabalho de Johnson [71], em 1973.

Faroe et al. [72] propõem uma heurística que foi denominada por *Busca Local Guiada* (GLS, do inglês, *Guied Local Search*), e Lodi *et al.*, em seus trabalhos [73] e [74], propuseram algoritmos baseados em *Busca Tabu* e em um novo procedimento construtivo para exemplos em

duas e três dimensões. Posteriormente, em [75], Lodi et al desenvolveram um código de *Busca Tabu* unificado, em que problemas gerais e multidimensionais de carregamento de contêiners foram então considerados. Mais recentemente, em 2009, Crainic et al. [76] desenvolveram um algoritmo de *Busca Tabu* de dois níveis em que o primeiro nível visava reduzir o número de contêiners e o segundo, otimizava o carregamento de caixas dentro deles, utilizando a representação IG proposta por Fekete e Shepers [70] e Fekete et al. [77].

Para o caso específico do 2D-BPP, Boschetti e Mingozzi [78] desenvolveram uma heurística construtiva eficaz, no qual admite-se um parâmetro valorado a cada caixa, fazendo uma ordenação decrescente das caixas de acordo com os valores associados, e atualizando tais parâmetros de acordo com algum critério particular do problema em questão. Foi estabelecido como critério de parada ou quando a solução ótima é encontrada, ou se um número máximo de iterações é atingido. Monaci e Toth [79] também propuseram uma heurística para o caso 2D, baseada em um *Conjunto de Cobertura* (do inglês, *Set-Covering*), que é composta de duas fases: na primeira, um grande número de colunas é gerado por um procedimento heurístico e pelo algoritmo exato proposto em Martello e Vigo [80], com um tempo limite especificado; na segunda fase, estas colunas são utilizadas para resolver o problema *Conjunto de Cobertura*, no qual resulta na solução para o problema bidimensional original.

No que concerne ao 3D-BPP, um trabalho importante na literatura é o de Parreño et al. [81], que desenvolveram, em 2010, um novo algoritmo híbrido GRASP/VND. Neste, a fase construtiva é baseada na *Heurística de Espaços Maximais Vazios*, desenvolvida por Lai e Chan [20], enquanto que a fase de melhoramento é marcada por várias novas modificações que são realizadas e combinadas em uma estrutura VND. O algoritmo apresentado também pode ser reduzido a problemas bidimensionais.

A referência de estudo inicial deste trabalho teve como ponto de partida as pesquisas desenvolvidas por Resende e Gonçalves, [17] e [12], supracitados no início desta seção, por apresentarem as melhores soluções para o 3D-BPP de que se tem conhecimento. Além disso, é importante ressaltar que o interesse também foi proveniente do fato de que os autores ainda aplicaram o mesmo algoritmo genético *viciado* (adaptado) a outros problemas da literatura, não somente a Problemas de Carregamento de Contêiners, obtendo resultados de excelente qualidade.

3 ALGORITMOS GENÉTICOS

Neste capítulo, apresentam-se o Algoritmo Genético e algumas de suas variações que serão utilizadas posteriormente como fundamentação teórica para a continuidade deste trabalho. Após uma descrição inicial sobre algoritmos genéticos em formato padrão, apresenta-se o *Algoritmo Genético de Chaves Aleatórias Viciadas*, que pode ser aplicado a vários tipos de problema das engenharias, porém, que será aqui desenvolvido sob a ótica de Problemas de Carregamento. Também desenvolve-se as principais estratégias do *Algoritmo Genético de Chu-Beasley*, que considera o trabalho com infactibilidades e que foi aplicado especificamente a Problemas de Atribuição Generalizada, obtendo excelentes resultados.

2.3 O ALGORITMO GENÉTICO

Algoritmos Genéticos (AG) são meta-heurísticas que foram inspiradas na teoria da evolução natural das espécies, de Charles Darwin, contendo mecanismos nomeados tal qual esta teoria, tais como: *genética* (problema de otimização), *cromossomo* (solução), *gene* (variável), evolução de populações por *ciclos geracionais* (conjunto de soluções modificadas no decorrer das iterações), além dos processos de *Seleção*, *Recombinação* e *Mutação*, que representam os operadores matemáticos comuns a todo Algoritmo Genético.

Propostos pela primeira vez por Holland [82], em 1975, os algoritmos genéticos são métodos que, assim como qualquer meta-heurística, realizam um processo de busca através de *transições*. Em uma meta-heurística, tais transições podem ocorrer a partir de um único ponto ou de um conjunto de pontos, percorrendo o *Espaço de Busca*. Nos AG, considera-se sempre um conjunto de pontos (*população*), e a estratégia de busca depende do problema específico a que ele é aplicado, e do conceito de *vizinhança* considerado. Os indivíduos da população (pontos) são soluções para o problema original, e estão em um formato codificado, i.e., cuja formatação necessita passar por um procedimento de decodificação para que, posteriormente, seja transformado em uma solução para o problema original, com um valor de *fitness* associado. Cada solução codificada é nomeada como *indivíduo/cromossomo* da população, contendo em sua estrutura *genes*, que são as componentes que estruturam a solução (em geral, as componentes de um vetor ou de uma matriz).

Operadores matemáticos de *Seleção*, *Recombinação* e *Mutação* são aplicados nas populações para que um conjunto de soluções evolua para populações contendo indivíduos mais diversificados geneticamente, capazes de gerar, no mínimo, soluções de igual qualidade, ou

melhor.

De acordo com Chu e Beasley [21], um algoritmo genético (AG) pode ser definido como "um algoritmo de busca probabilístico 'inteligente', o qual simula o processo de evolução ao tomar uma população de soluções e ao aplicar operadores genéticos em cada reprodução. Cada solução na população é calculada de acordo com alguma medida de *fitness*".

Nos algoritmos genéticos tradicionais, a população inicial é gerada de forma aleatória, porém, muitas variações utilizam-se de populações inicialmente geradas através de alguma heurística ou com estratégias aleatórias controladas, visando a diminuição do tempo computacional. Mas a falta de aleatoriedade, apesar de, em geral, melhorar a eficiência do tempo, pode fazer com que a qualidade da solução obtida fique restrita e correlacionada à qualidade do formato da heurística ou à estratégia que formou a população inicial, e que muitas vezes, é impossível de ser determinada. Assim, há uma preponderância na literatura em considerar-se outros mecanismos de geração da população inicial, que não seja aleatório, apenas para problemas que possuem soluções infactíveis na formação de suas populações.

Problemas de otimização com infactibilidades é outro tópico comumente abordado pelos algoritmos genéticos, que, em geral, realizam uma penalização ou eliminação dos indivíduos que violam alguma das restrições do problema tratado, contidos na população corrente e que poderão evoluir para a próxima população. O estudo de Venkatraman e Yen [83] apresenta de maneira detalhada formas de se trabalhar com infactibilidades tal que a calibragem dos parâmetros de penalização possam ser escolhidos da melhor maneira possível. Muitas vezes, dependendo de como estes parâmetros são definidos, soluções com pequenos graus de infactibilidade podem gerar soluções de excelente qualidade. No AG, é comum que esse trabalho com infactibilidades seja incluído na forma de se calcular a *fitness*, ou seja, na forma de se medir a função objetivo, explicitando-as através de condições de penalidade dentro da função objetivo.

O trabalho de Mitchell [84] descreve o algoritmo de um AG simples como a seguir (adaptado):

1. Iniciar com uma população de *n* cromossomos, gerada aleatoriamente (soluções candidatas ao problema).

2. Calcular a *fitness f* (*x*) de cada cromossomo *x* na população.

3. Repetir os passos a seguir até que *n* descendentes tenham sido criados.

a) Selecionar um par de cromossomos pais da população corrente, com probabilidade de seleção sendo uma função crescente da *fitness*. O mesmo cromossomo pai pode ser selecionado

mais que uma vez.

b) Com probabilidade p_r (*probabilidade de recombinação* ou *razão de recombinação*), recombinar o par utilizando um ponto escolhido aleatoriamente (escolhendo com probabilidade uniforme) para formar dois descendentes. (Observa-se aqui que a razão de recombinação é definida como sendo a probabilidade de dois pais efetuarem a recombinação em um único ponto. Existe também versões para a recombinação multi-pontos do AG, nos quais a razão de recombinação para um par de pais é um número de pontos maior que um).

c) Realizar a mutação de dois descendentes com probabilidade p_m (*probabilidade de mutação* ou *razão de mutação*), e colocar os cromossomos resultantes na nova população. Se *n* é ímpar, um destes dois descendentes mutantes pode ser descartado.

4. Substituir a população corrente pela nova população.

5. Voltar ao Passo 2.

Cada iteração deste processo é chamado *Geração* do Algoritmo Genético ou *Ciclo Geracional*. Este formato, em geral, segue uma padronização do tipo *Maximização da Saída*, e tal que os valores para a *fitness* sejam sempre não-negativos.

A seleção descrita no item 3(a) é conhecida na literatura como *Seleção Proporcional*, em que o número de descendentes é proporcional ao valor da *fitness*. A dificuldade em se utilizar a Seleção Proporcional, assim como sugerem Romero e Rider [85], é que muitas são as restrições que devem ser satisfeitas para que seja válida sua utilização. Primeiramente, o problema a ser resolvido deve estar padronizado. Além disso, o número de descendentes é determinado de acordo com uma relação de proporcionalidade, que gera um número de descendentes não-inteiros. Logo, esse número necessita ainda ser integralizado (o que em geral é feito utilizando-se um operador conhecido como *Roleta*). Estas exigências fazem com que a Seleção Proporcional não seja necessariamente um procedimento viável, uma vez que muitas dificuldades e novas condições decorrem de sua utilização.

A *Seleção por Torneio* é uma modificação da Seleção Proporcional, que também é um procedimento que gera descendentes, porém, sem restrições ou condições, e que é facilmente definido dentro de um código. Em geral, a Seleção por Torneio é definida pela realização de *n* jogos, para uma população de tamanho *n*, no qual cada jogo considera um número *k* de possíveis soluções pais para gerar um descendente. Escolhe-se a combinação de dois pais de acordo com a qualidade do filho, comparando os descendentes e escolhendo aquele que possui melhor valor *fitness*. O valor *k*

vem sendo utilizado pela literatura em valores fixados em $k = 2, 3, 4$.

A questão mais importante dos algoritmos genéticos é determinar um *Decodificador* eficaz, isto é, um procedimento que transforma (decodifica) os cromossomos (soluções codificadas) em uma configuração de solução para o problema original, e que fornece ainda um valor de *fitness* associado. A codificação mais simples de se representar é aquela que considera valores binários para os genes. Porém, variáveis inteiras e reais também podem ser consideradas, e, neste caso, deve-se definir os procedimentos e as especificidades da recombinação e mutação. "A recombinação e a mutação são operadores altamente dependentes do tipo de codificação usado"[85].

O trabalho de Romero e Rider [85] resume a implementação dos algoritmos genéticos, através dos seguintes passos:

1. Especificar os parâmetros de controle (tamanho da população, taxa de recombinação, taxa de mutação, etc).

2. Especificar as características genéricas do algoritmo tais como tipo de codificação, forma de montar a população inicial, manipulação de infactibilidades, tipo de seleção, necessidade e forma de padronização, etc.

3. Encontrar uma população inicial que se transforma na população corrente.

4. Encontrar o fitness da população corrente e atualizar a incumbente (melhor solução encontrada no processo), caso seja possível.

5. Se o critério de parada for satisfeito, parar. Caso contrário, continuar o processo.

6. Implementar a Seleção.

7. Implementar a Recombinação.

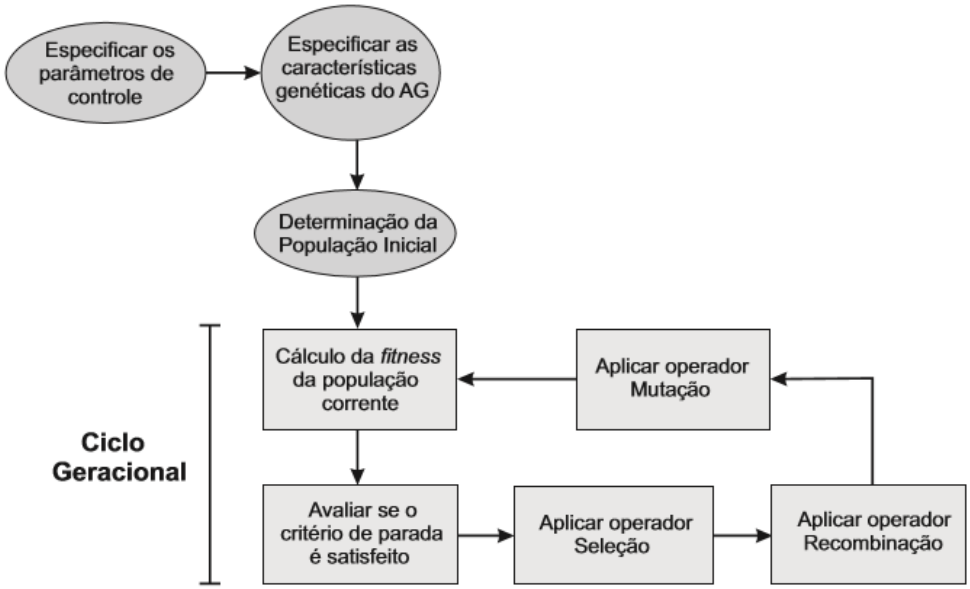
8. Implementar a Mutação, recompor a população corrente e voltar ao passo 4.

Na Figura 3.1 apresenta-se o esquema desta sequência.

O principal problema dos algoritmos genéticos refere-se à perda da diversidade nas populações, à medida que estas evoluem no decorrer das transições. As soluções de melhor qualidade tendem a se tornar repetidas ou dominantes, por terem maior probabilidade de repassarem seus genes. Segundo Romero e Rider [85], para contornar parcialmente este problema, duas estratégias podem ser utilizadas: (1) usar taxas de mutação cada vez mais elevadas; e (2) verificar e eliminar as soluções

repetidas e fazer a recomposição da população, em geral, através da geração aleatória de novas soluções. Porém, esta segunda estratégia geralmente exige um gasto computacional muito elevado, tornando-se inviável para a implementação, na maioria dos casos.

Figura 3.1 - Esquema da sequência de passos da implementação do AG.



Fonte: Elaboração do próprio autor.

2.4 ALGORITMO GENÉTICO DE CHAVES ALEATÓRIAS VICIADAS

Resende e Gonçalves têm publicado na literatura trabalhos recentes que envolvem a resolução de problemas clássicos da Pesquisa Operacional através de uma nova proposta de Algoritmo Genético, que eles chamaram de Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA, do inglês, *Biased Radom-Key Genetic Algorithm*). Os autores publicaram, inclusive, pesquisas voltadas especificamente à resolução dos problemas SSSCSP e SBSBPP (chamando-os apenas de 3D-BPP), em que apresentaram ainda um nova função objetivo que permite analisar com maior precisão a qualidade das soluções encontradas. O BRKGA aplicado aos Problemas de Carregamento de Contêiners, por Resende e Gonçalves, pode ser encontrado em [17], [12], [18] e, com restrições não lineares, por Resende e Silva, em [19].

Considerado como uma evolução do Algoritmo Genético tradicional, o BRKGA tem demonstrado possuir vários fatores positivos que o fizeram tornar-se o ponto de partida no desenvolvimento do presente trabalho de doutorado. Primeiramente, o BRKGA pode ser aplicado não apenas aos PCC, mas também a vários outros tipos e categorias de problemas da PO. Grande

parte da implementação pode ser aproveitada, e independe do problema tratado, sendo apenas modificados os procedimentos de codificação e decodificação. Além disso, em todos eles, o BRKGA tem demonstrado superioridade na eficiência e qualidade das soluções obtidas, se comparado aos melhores algoritmos encontrados na literatura. Outro ponto posivito importante é que o BRKGA é um algoritmo recente que tem potencial para apresentar-se como uma referência em um novo horizonte de estudos, na busca pela melhoria de estratégias que otimizem os resultados.

Para entender o Algoritmo Genético de Chaves Aleatórias Viciadas, é necessário inicialmente entender como se desenvolve o Algoritmo Genético de Chaves Aleatórias (RKGA, do inglês, *Radom-Key Genetic Algorithm*), proposto pela primeira vez em 1994, por Bean [86], e cujas ideias principais estão descritas a seguir. O desenvolvimento desta subseção tem como orientação os trabalhos de Resende e Gonçalves, [12] e [18].

3.2.1 O algoritmo genético de chaves aleatórias - RKGA

As principais diretrizes para o RKGA podem ser resumidas da seguinte forma:

- *Cromossomo* é um termo utilizado para nomear um vetor de componentes reais, cujos números foram gerados aleatoriamente no intervalo [0,1]. Um conjunto de *cromossomos*, formado por vetores de chaves (genes) geradas randomicamente por valores de zero a um, constituem uma *população*, que é formada em cada *geração* (iteração);

- Evoluindo através de gerações, a população inicial principia de forma aleatória, contendo p vetores que correspondem a n-uplas de chaves aleatórias;

- O RKGA possui um algoritmo determinístico, o *Decodificador*, que toma como entrada um vetor solução qualquer e associa a ele uma solução do problema de otimização original, cujo valor da função objetivo pode ser calculado, e é dado como valor de saída.

- A estratégia do RKGA consiste em calcular o valor da função objetivo de cada indivíduo da geração k através do Decodificador e, em seguida, particionar a população em dois grupos disjuntos: 1) p_e: pequeno grupo de indivíduos elite, que são os cromossomos que apresentaram as melhores *fitness* para o problema inicial; e, 2) $p-p_e$: grupo restante de indivíduos não-elite, composto pelos demais cromossomos da população que não fazem parte do conjunto elite. Para evoluir para a geração $k+1$, primeiramente, o algoritmo copia todos os indivíduos elite p_e, sem modificações. Ou seja, não há o processo de Seleção

convencional de um Algoritmo Genético (AG) tradicional, há apenas o operador elitismo, em que as melhores soluções são preservadas e a incumbente é mantida monotomicamente decrescente. Depois, o RKGA introduz o procedimento de Mutação, em que p_m mutantes são gerados dentro da população $k+1$, cuja representação é dada através de vetores de chaves aleatórias que foram gerados da mesma forma que um elemento da população inicial. Com p_e indivíduos e p_m mutantes introduzidos na nova geração $k+1$, ainda são agregados $p-p_e-p_m$ indivíduos para completar os p indivíduos da nova geração mediante operações em um processo conhecido como *Recombinação*, que seleciona dois pais aleatoriamente a partir de toda a população da geração atual k para gerar um único indivíduo descendente para a geração seguinte $k+1$.

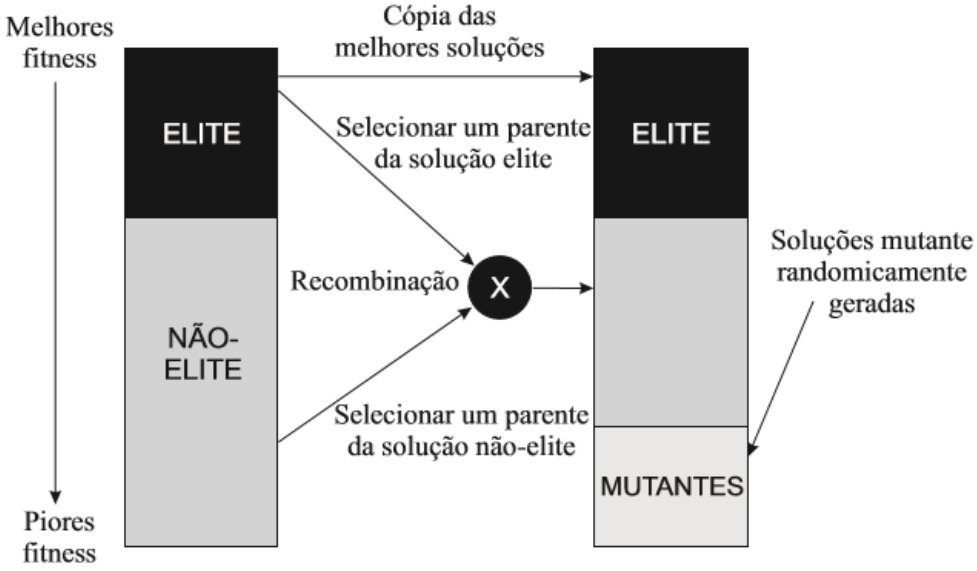
3.2.2 O BRKGA como uma evolução do RKGA

O BRKGA diferencia-se do RKGA apenas em como os pais são selecionados para gerar a nova prole de $p-p_e-p_m$ indivíduos no processo de Recombinação. No Algoritmo Genético de Chaves Aleatórias Viciadas, seleciona-se aleatoriamente um indivíduo pai da partição elite e o segundo individuo pai da partição não-elite (ver Figura 3.2). É claro que como a partição elite possui menos indivíduos que a partição não-elite, um indivíduo elite tem maior probabilidade de produzir mais que uma prole e passar suas características para gerações futuras, daí a incorporação da palavra "viciada"(do inglês, "biased"), no nome do RKGA. Ou seja, a Recombinação não é feita de maneira puramente aleatória, dado que uma solução geradora é necessariamente do conjunto elite (viciada), diferentemente do RKGA, em que os pais são selecionados de todo conjunto da população corrente, incluindo indivíduos elite e não-elite.

Dado que no BRKGA uma solução geradora de um indivíduo da próxima geração deve ser do grupo elite, existe uma segunda forma de selecionar a outra solução geradora, além de selecioná-la do grupo não-elite de $p-p_e$ elementos: escolhendo um indivíduo que esteja contido na população corrente de p elementos. Ou seja, para gerar um único indíviduo para a população da próxima geração $k+1$, é possível que uma das duas estratégias sejam seguidas para o processo de Recombinação no BRKGA:

- **Estratégia 1**: $\text{indivíduo}_{k+1} = \text{elite}_k + \text{nao-elite}_k$;
- **Estratégia 2**: $\text{indivíduo}_{k+1} = \text{elite}_k + \text{indivíduo}_k$.

Figura 3.2 - O operador Recombinação no processo de transição do BRKGA.



Fonte: Elaboração do próprio autor.

Além disso e do fato de que $p_e < p - p_e$ influencia com maior probabilidade uma solução geradora elite produzir mais descendentes, um outro fator que contribui de forma "viciada" para que o grupo elite tenha maior preponderância sobre a geração seguinte é o tipo de recombinação realizada no BRKGA, chamada *Cruzamento Uniforme Parametrizado*, ou *Recombinação PUC* (Do inglês, *Parametrized Uniform Crossover*), introduzido por Spears e DeJong, em seu trabalho [87], para gerar um único descendente através de uma recombinação uniforme. A Recombinação PUC é iniciada escolhendo-se um parâmetro $\rho_e > 0{,}5$, que representa a probabilidade de uma prole herdar a componente (alelo) do vetor de seu pai elite A. Se B for o segundo vetor pai, determinado através de uma das duas estratégias acima, então para gerar o descendente C da recombinação de A com B, é necessário determinar se sua componente $c(i)$, $i = 1, ..., n$, é copiada da i-ésima componente de A, $e(i)$, ou da i-ésima componente de B, $e(i)$. Assim, para cada um dos $p - p_e - p_m$ descendentes a serem gerados, um número aleatório ρ é gerado no intervalo $[0,1]$, n vezes, sendo que a cada i-ésima vez, $i = 1,...,n$, seu valor é comparado ao parâmetro ρ_e. Se $\rho \leq \rho_e$, então $c(i)$ é copiada do vetor elite, A. Caso contrário, $c(i)$ é copiada do vetor B. Logo, para cada $i = 1, ..., n$, tem-se que:

$$c(i) = \begin{cases} e(i), & se\ \rho \leq \rho_e \\ \bar{e}(i), & caso\ contrário \end{cases} \tag{3.1}$$

A Figura 3.3 ilustra o processo de Recombinação PUC dentro do BRKGA, para a recombinação entre dois indivíduos, Cromossomo 1 (considerado o indivíduo elite) e o Cromossomo 2 (considerado o outro indivíduo que pode ser obtido assim como definido na Estratégia 1 ou na Estratégia 2). O número de componentes dos cromossomos é $n = 4$ e o parâmetro escolhido é $\rho_e = 0{,}7$, ou seja, a prole deve herdar o componente do pai elite com 0,7 de probabilidade, e deve herdar o componente do outro pai com uma probabilidade de 0,3. Assim, são gerados quatro valores para o número aleatório, ρ, e as coordenadas da prole são obtidas conforme a equação (3.1).

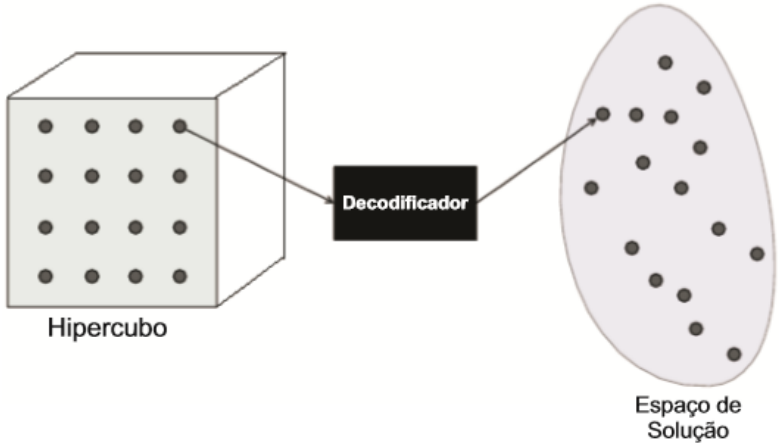
Figura 3.3 - Recombinação PUC do BRKGA.

Fonte: Adaptado de Resende e Silva (2013) [19].

Após construir toda a população de p indivíduos da geração $k+1$, um Decodificador transforma cada indivíduo de componentes no intervalo $[0, 1]$ em uma solução factível do problema inicial, calculando juntamente a isso, o valor de cada *fitness* das novas propostas de solução. E, assim, a nova população é particionada em dois grupos, elite e não-elite, iniciando-se uma nova geração.

É importante ressaltar que a busca das soluções em problemas de otimização combinatória pelo algoritmo BRKGA é realizada de forma indireta, pois não é feita sobre o espaço de busca do problema original, diretamente. A busca é realizada sobre um hipercubo unitário n-dimensional e, depois, através do Decodificador, é mapeada sobre o espaço de busca original (Ver esquema da Figura 3.4).

Figura 3.4 - Espaço de busca indireto mapeado pelo Decodificador.



Fonte: Adaptado de Resende e Gonçalves (2011) [18].

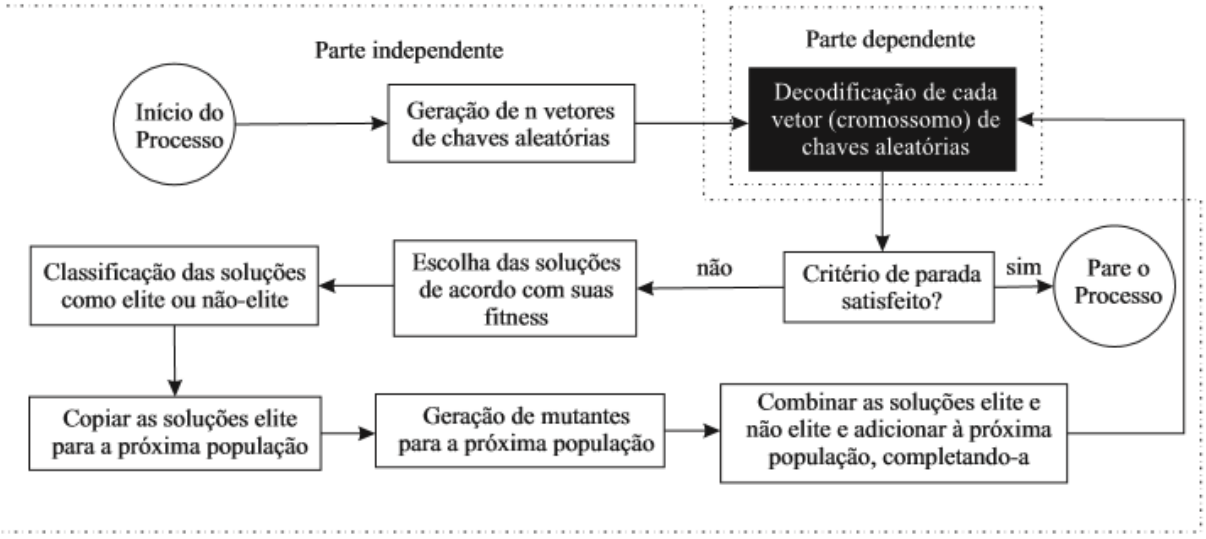
O BRKGA possui uma estrutura geral que permite resolver problemas de diversos tipos. Tal estrutura é dividida em duas partes bem distintas: uma delas é dependente do problema particular a ser resolvido, e a outra é totalmente independente ao problema. A parte da estrutura independente, não depende do problema que está sendo resolvido e refere-se apenas à busca no espaço definido pelo hipercubo unitário n-dimensional. A parte dependente relaciona-se com a conexão estabelecida pelo decodificador que recebe as propostas de solução codificadas (n-uplas de componentes no intervalo $[0, 1]$) e, a partir delas, transforma-as em soluções factíveis do Espaço de Busca do problema original. Na Figura 3.5 ilustram-se as partes dependentes e independentes, dentro de um esquema geral que descreve o BRKGA.

Assim, a questão mais importante na implementação do BRKGA é definir como se dá o funcionamento da codificação de uma solução na forma de chaves aleatórias e, principalmente, como decodificá-las através do operador Decodificador, que depende necessariamente do problema a ser resolvido. Esta parte dependente do BRKGA configura-se como a questão que pode trazer maior dificuldade para a formação das estratégias, porém, em contrapartida, a parte independente pode ser aproveitada para os vários tipos de problemas da PO.

3.2.3 Decodificador do BRKGA aplicado ao 3D-BPP: visão geral

No BRKGA aplicado ao Problema de Carregamento de Contêiners, e, em particular, ao 3D-BPP, o Decodificador é um algoritmo heurístico construtivo que determina a forma de carregamento, uma caixa por vez, dentro de um contêiner. As posições factíveis são identificadas pelo Decodificador, que executa o carregamento de novas caixas dentro do procedimento sequencial de carregamento de caixas.

Figura 3.5 - Estrutura Geral do BRKGA.



Fonte: Elaboração do próprio autor.

O Decodificador é chamado *Procedimento de Carregamento* nos trabalhos de Gonçalves e Resende, e sua estratégia é baseada na constituição de uma lista de *Espaços Maximais Vazios* (EMS, do inglês, *Empty Maximal Spaces*), que determina as possibilidades de carregamento de caixas. Esse Decodificador foi proposto, pela primeira vez, por Lai e Chan [20], em 1997, e será explicado em detalhes no decorrer desta subseção.

O trabalho de Resende e Gonçalves [12], propõe uma codificação tal que cada cromossomo contém 3*n* chaves aleatórias, e representa uma proposta de solução de carregamento para *n* caixas. Os valores das chaves (ou também comumente nomeadas por *genes*) de um cromossomo são tidos como os valores de entrada do Decodificador, onde cada um dos três conjuntos de *n* genes, representa as seguintes informações, respectivamente: (1) a sequência de carregamento das caixas - BPS (do inglês, *box packing sequence*); (2) o vetor da heurística de carregamento - VPH (do inglês, *vector of placement heuristic*); e (3) o vetor de orientação das caixas - VBO (do inglês, *vector of box orientation*). O decodificador *Procedimento de Carregamento* pode ser dividido em etapas, em que decodifica um cromossomo, transforma-o em uma solução para o PCC original, e retorna ainda como valor de saída uma *fitness* associada ao carregamento realizado, da seguinte forma:

1. *Etapa 1: Decodificação da sequência das caixas* - decodificação da primeira parte do cromossomo no que concerne à sequência com que as caixas são carregadas dentro do contêiner (**BPS**);

2. *Etapa 2: Decodificação da Heurística de Carregamento* - decodificação da segunda parte do cromossomo, no que concerne ao vetor de heurística de carregamento (**VPH**) para

ser usado no *Procedimento de Carregamento*;

3. *Etapa 3: Decodificação da orientação das caixas* - decodificação do vetor de orientação das caixas (**VBO**), para ser usado no *Procedimento de Carregamento*;

4. *Etapa 4: Estratégia de Carregamento* - utilização do BPS, VPH e VBO, definidos nas etapas 1-3, para construir um carregamento de caixas dentro dos contêiners;

5. *Etapa 5: Cálculo da Fitness* - cálculo do valor da *fitness* (que mede a qualidade do carregamento de caixas no container), que melhora significativamente a forma de se analisar e compreender os resultados.

Na Figura 3.6, ilustram-se a sequência anterior de cinco etapas que definem o Decodificador *Procedimento de Carregamento*, e seu processo de decodificação de um cromossomo (proposta de solução codificada), aplicado especificamente para o 3D-BPP. Vale ressaltar novamente que o Decodificador é o único procedimento dentro do BRKGA que depende do problema a ser resolvido, ou seja, o *Procedimento de Carregamento* é a *parte dependente* do Algoritmo BRKGA para o 3D-BPP.

3.2.4 Detalhamento das etapas de decodificação

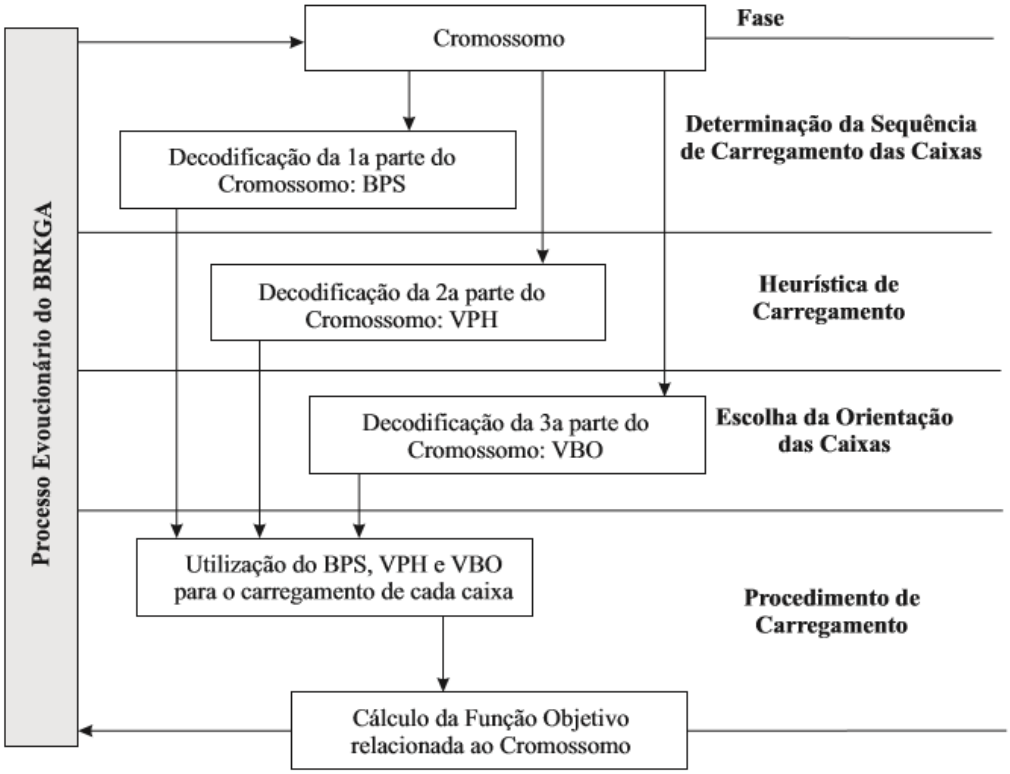
Para entender o decodificador *Procedimento de Carregamento*, é necessário inicialmente compreender como funciona o cromossomo com informações codificadas. Cada cromossomo possui $3n$ genes com valores no intervalo $[0, 1]$, em que n é número total de caixas a serem carregadas dentro de contêiners. Considera-se:

- **Do gene 1 ao gene n**: parâmetros utilizados para obter o BPS (sequência de carregamento das caixas);

- **Do gene n+1 ao gene 2n**: valores usados para obter o vetor VPH, que é o vetor da Heurística de Carregamento;

- **Do gene 2n+1 ao gene 3n**: parâmetros usados para obter o vetor de orientação das caixas, VBO.

Figura 3.6 - Detalhamento do Decodificador dentro do BRKGA - Parte dependente.



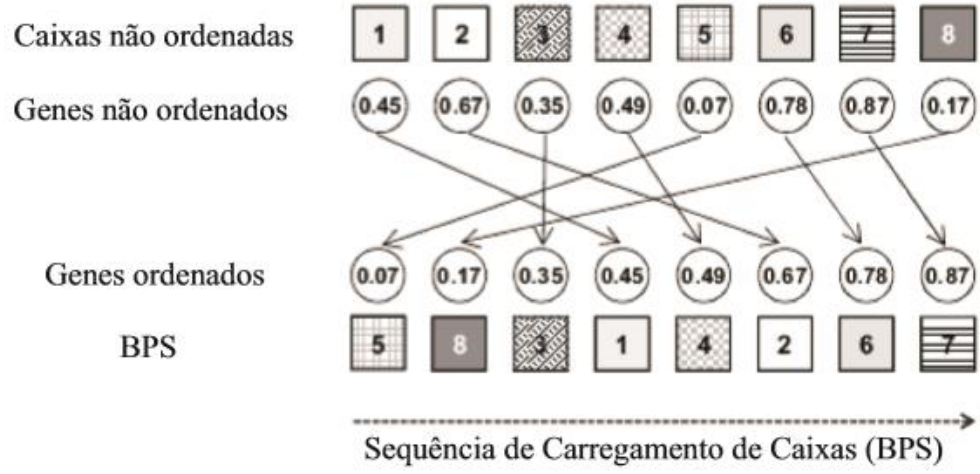
Fonte: Adaptado de Resende e Gonçalves (2013) [12].

Etapa 1: Decodificação do BPS

Na primeira etapa, mapeiam-se os primeiros *n* genes do cromossomo, considerando-se as *n* caixas ordenadas de forma crescente e associando-se a elas cada um destes genes. Isto quer dizer que a caixa 1 é associada ao gene 1, caixa 2 é associada ao gene 2, e assim sucessivamente até que a caixa *n* seja associada ao gene *n*. Em seguida, reordenam-se de forma crescente os n primeiros genes do cromossomo, de tal forma que as caixas associadas a eles preservem a ordenação de seus genes associados. Essa nova ordenação das caixas forma o vetor BPS.

Na Figura 3.7 apresenta-se um exemplo que ilustra a estratégia aplicada aos primeiros n=8 genes da sequência de 3n=24 chaves aleatórias de um cromossomo contendo os valores de entrada do decodificador *Procedimento de Carregamento*. Estes n=8 primeiros elementos do cromossomo foram gerados randomicamente com valores iguais a 0,45; 0,67; 0,35; 0,49; 0,07; 0,78; 0,87 e 0,17. Então, considerando as caixas com menores valores de cromossomo como prioritários, é contruído o vetor BPS= (5,8,3,1,4,2,6,7), contendo a ordenação das caixas a serem carregadas. Assim, o vetor BPS mostra a sequência em que as caixas devem ser carregadas.

Figura 3.7 - Decodificação do BPS.



Fonte: Adaptado de Resende e Gonçalves (2013) [12].

Etapa 2: Decodificação do VPH

Na segunda etapa, para obter o vetor VPH a ser utilizado no *Procedimento de Carregamento*, duas heurísticas nomeadas em [12] por *Heurísticas de Carregamento* são consideradas: a *Heurística BBL* (do inglês, *Back-Bottom-Left*, canto anterior esquerdo) e a *Heurística BLB* (do inglês, *Back-Left-Bottom*, canto esquerdo anterior). A decodificação do vetor de heurística de carregamento VPH, é realizada componente a componente, VPH_i, $i = 1, ..., n$, avaliando-se através da seguinte expressão:

$$VPH_i = \begin{cases} BBL, & se \quad gene_{n+1} \leq 0{,}5 \\ BLB, & se \quad gene_{n+1} > 0{,}5 \end{cases} \tag{3.2}$$

com i = 1, ..., n.

Dentro do decodificador *Procedimento de Carregamento*, o VPH é usado para selecionar qual heurística de carregamento aplicar em cada caixa a ser alocada dentro do contêiner. Ou seja, dentre as possíveis posições para se carregar uma caixa i, o $V\,P\,H_i$ indica qual delas deve ser selecionada.

A explicação completa sobre o funcionamento do vetor VPH só é possível ser realizada após o detalhamento da Etapa 4 do Decodificador, e está indicada em forma de observação, ao final da subseção 3.2.4.

Etapa 3: Decodiftcação do VBO

Na terceira etapa, a decodificação do terceiro conjunto de *n* valores no intervalo [0,1] contidos em um cromossomo é realizada de forma direta na composição do vetor VBO, ou seja, cada componente do vetor VBO é obtido pela expressão:

$$VBO_i = gene_{2n+i}$$

com $i = 1, ..., n$.

A escolha da orientação da caixa dentre as k possíveis orientações ($1 \leq k \leq 6$ para o 3D-BPP e $1 \leq k \leq 2$ para o 2D-BPP, $k \in Z$) é escolhida aleatoriamente através da relação $\lfloor VBO_i \times nBO \rfloor$, onde *nBO* é o número de orientações possíveis.

Etapa 4: A Estratégia de Carregamento

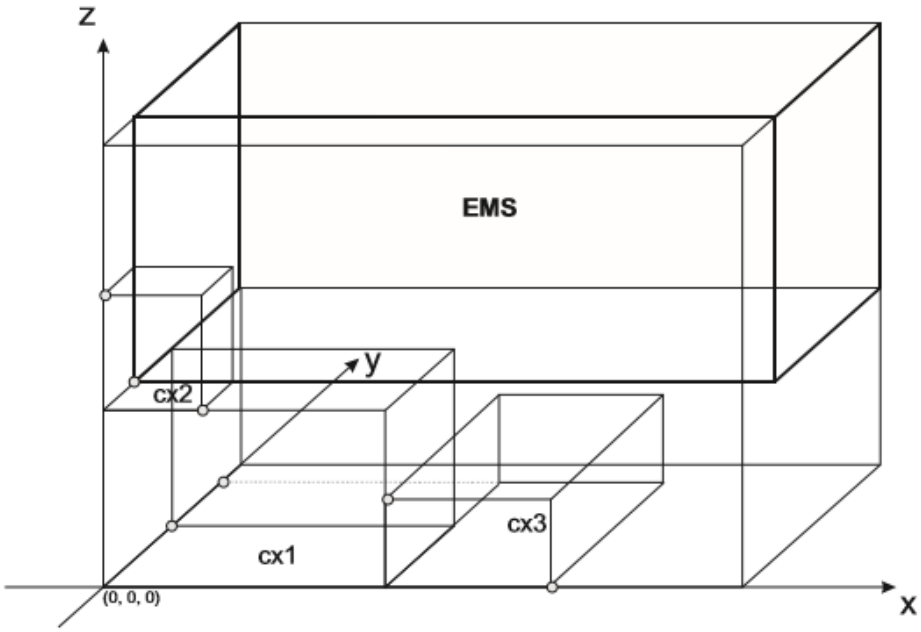
A Etapa 4 é o passo mais importante do Decodificador, e a etapa que exige maior complexidade e tempo computacional, considerando-se todo o algoritmo BRKGA. É neste passo em que, de fato, o carregamento é executado. A *Estratégia de Carregamento*, proveniente do trabalho de Lai e Chan [20], foi utilizada por Resende e Gonçalves [12] para que as informações obtidas nas Etapas 1 a 3, codificadas, fossem transformadas, de fato, em um carregamento de caixas para o problema original, com *fitness* associada.

Lai e Chan [20] definem um espaço vazio em formato de paralelepípedo (para o caso 3D), ou de retângulo (para o caso 2D), como *maximal* quando ele não está contido em nenhum outro espaço de mesma forma dentro de um contêiner. Cada *Espaço Maximal Vazio* (EMS) é representado por $[(x_i, y_i, z_i), (X_i, Y_i, Z_i)]$, no qual as 3-uplas, (x_i, y_i, z_i) e (X_i, Y_i, Z_i), denotam as coordenadas mínimas e máximas de cada EMS, respectivamente. Este caso pode ser particularizado para problemas bidimensionais, ao considerar as coordenadas em *z* iguais a zero. Para efetuar o carregamento de uma caixa qualquer, o procedimento *Estratégia de Carregamento* realiza uma busca ordenada pelas coordenadas mínimas (x_i, y_i, z_i) de cada EMS disponível, escolhendo o primeiro espaço maximal vazio que seja compatível com o carregamento da caixa. Essa busca segue a ordenação de uma lista (um conjunto) S_b, que é criada para armazenar todos os espaços maximais vazios disponíveis em um contêiner *b*. Novos contêiners são abertos, sempre que os disponíveis não conseguem acomodar toda a demanda.

Na Figura 3.8 mostra-se o carregamento de três caixas dentro de um contêiner, cujos pontos cinza em destaque são as coordenadas mínimas dos sete EMS possíveis gerados após a configuração de carregamento. Um dos EMS está em negrito, ilustrando o formato de

paralelepípedo vazio maximal, i.e., tal que não existe outro paralelepípedo vazio que contenha o primeiro. É importante ressaltar ainda que a primeira caixa sempre é carregada na posição (0, 0, 0), dado que, no início do processo de carregamento, o próprio contêiner forma o primeiro espaço (paralelepípedo) maximal vazio $[(0,0,0),(D,W,H)]$.

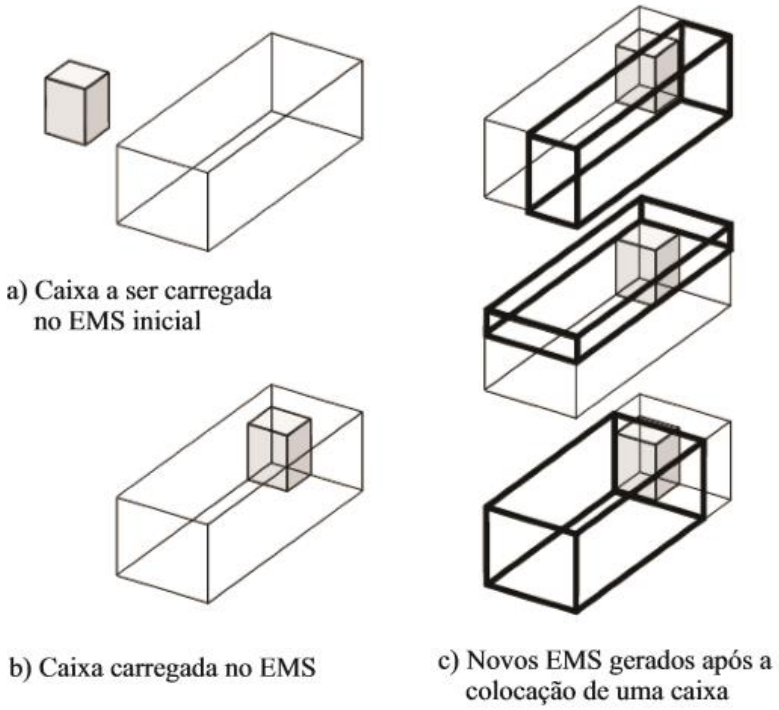
Figura 3.8 - Espaços Maximais Vazios para o caso tridimensional.



Fonte: Elaboração do próprio autor.

A geração de EMS é feita por um processo que Lai e Chan chamaram de *DP* (do inglês, *Difference Process*). Tal processo é gerado uma ou mais vezes, conforme a necessidade, sempre que uma nova caixa é carregada. A Figura 3.9 apresenta a representação geométrica da aplicação do processo DP para a geração dos EMS de um problema tridimensional. No exemplo, inicialmente existe um contêiner vazio (parte a), e, portanto, ele próprio é o único EMS contido na lista S_b, para algum contêiner *b*. Logo, a primeira caixa foi carregada na origem (0,0,0), que são as coordenadas mínimas do contêiner (parte b). Em seguida, na parte c, a lista S_b de EMS foi atualizada, contendo três EMS possíveis, e portanto, três possibilidades de carregamento de caixas (em suas coordenadas mínimas) para a próxima iteração, ou seja, em que a caixa seguinte deve ser carregada. Assim, sempre que uma caixa é colocada dentro do contêiner, a lista S_b deve ser atualizada pelo processo DP para acusar quais as possibilidades de carregamento para a próxima caixa. Em síntese, o *Processo DP* é o gerador de EMS, cujas coordenadas mínimas fornecem as possíveis posições para o carregamento da caixa seguinte.

Figura 3.9 - Representação geométrica do Processo Diferença (DP).



Fonte: Adaptado de Resende e Gonçalves (2013) [12].

Em termos algébricos, o DP é baseado em um procedimento de projeções entre caixa carregada e o EMS selecionado para sua alocação, devendo ser utilizado em duas ocasiões: 1) quando uma caixa foi carregada em um EMS selecionado; ou 2) sempre que a caixa carregada sobrepôs um EMS não selecionado. Se $[(x_1, y_1, z_1), (x_2, y_2, z_2)]$ representam as coordenadas do EMS selecionado ou do EMS não selecionado que foi sobreposto, e que $[(x_3, y_3, z_3), (x_4, y_4, z_4)]$ denotam as coordenadas da caixa carregada, tem-se como pressupostos iniciais que:

$$x_1 \leq x_3 \leq x_4 \leq x_2, \qquad y_1 \leq y_3 \leq y_4 \leq y_2 \qquad \text{e} \qquad z_1 \leq z_3 \leq z_4 \leq z_2,$$

Na sequência, o *Processo DP* constitui-se de duas fases, a primeira, gera os possíveis EMS através das projeções e, a segunda, realiza testes que eliminam EMS infactíveis.

<u>Fase 1 do Processo DP</u>: Geração de EMS

Sempre que uma caixa é carregada em um EMS, a caixa sobrepõe o espaço do próprio EMS escolhido, e pode também sobrepor parcialmente outros EMS. O processo DP realiza uma subtração entre o espaço do EMS (que é completa ou parcialmente sobreposto) e o espaço ocupado pela caixa, que está em comum com o espaço do EMS. Para isso, a estratégia DP promove o cálculo

de todas as possibilidades de projeções, o que, para o caso tridimensional, configura-se como seis subespaços candidatos a serem espaços maximais vazios para a próxima iteração. Testes de eliminação posteriormente excluem espaços gerados pelo processo DP que não tem capacidade de promover o carregamento das caixas remanescentes. Os espaços não eliminados são classificados como EMS para a próxima iteração e são incluídos na lista de EMS do contêiner b em que houve o DP. A expressão que determina os seis espaços provenientes do processo DP foi desenvolvida por Lai e Chan [20], e é apresentada a seguir:

$$
\begin{aligned}
DP &= [(x_1, y_1, z_1), (x_2, y_2, z_2)] - [(x_3, y_3, z_3), (x_4, y_4, z_4)] \\
&= \{[(x_1, y_1, z_1)], [(x_3, y_2, z_2)], \\
&\quad [(x_4,y_1,z_1)],[(x_2,y_2,z_2)], \\
&\quad [(x_1,y_1,z_1)],[(x_2,y_3,z_2)], \\
&\quad [(x_1,y_4,z_1)],[(x_2,y_2,z_2)], \\
&\quad [(x_1,y_1,z_1)],[(x_2,y_2,z_3)], \\
&\quad [(x_1, y_1, z_4)], [(x_2, y_2, z_2)]\}
\end{aligned}
$$

Para o caso particular bidimensional, pode-se desconsiderar as duas últimas projeções, uma vez que as coordenadas em z serão mantidas, porém, consideradas iguais a zero.

Após a geração do conjunto DP acima, atualiza-se a lista S_b do contêiner aberto *b* em que foi realizado o carregamento e efetuado o Processo DP.

<u>Fase 2 do Processo DP</u>: Testes de Eliminação de EMS

Após a Fase 1, iniciam-se os procedimentos de eliminação de EMS que são considerados infactíveis ou redundantes, como por exemplo: EMS que possuem larguras ínfimas (muito estreitos), ou EMS que estão totalmente inscritos por outros EMS. Para a identificação de EMS inutilizáveis, Lai e Chan [20] apresentam dois processos de eliminação que estão descritos abaixo, e doravante nomeados por *Teste1* e *Teste2*.

- **Teste 1 - Eliminação de EMS inscritos:** Compare cada EMS de coordenadas $[(x_1, y_1, z_1), (x_2,y_2, z_2)]$ com todos os outros EMS disponíveis na lista de EMS de um contêiner aberto, S_b. Elimine $[(x_1, y_1, z_1),(x_2, y_2,z_2)]$ ao compará-lo com um EMS = $[(x_1', y_1',z_1'), (x_2', y_2', z_2')]$ se:

$$(x_1 \geq x_1') \text{ e } (y_1 \geq y_1') \text{ e } (z_1 \geq z_1') \text{ e } (x_2 \leq x_2') \text{ e } (y_2 \leq y_2') \text{ e } (z_2 \leq z_2')$$

- **Teste 2 - Eliminação de EMS de larguras ínftmas:** Elimine um EMS de coordenadas $[(x_1,y_1,z_1),(x_2,y_2,z_2)]$ das listas de EMS se:

$$(x_1 = x_2) \text{ ou } (y_1 = y_2) \text{ ou } (z_1 = z_2) \text{ ou } (x_1' = x_2') \text{ ou } (y_1' = y_2') \text{ ou } (z_1' = z_2')$$

De acordo com Resende et al. [12], os Testes 1 e 2 representam o processo que mais demanda tempo computacional dentro da *Estratégia de Carregamento* do decodificador *Procedimento de Carregamento*. Para reduzir este tempo, os autores propuseram outras duas novas estratégias prévias que fazem reduzir em 60% o tempo total demandado:

- **Teste1-Prévio**: Se o volume do novo EMS criado é menor que o volume de cada caixa remanescente a ser carregada, não o adicione à lista S_b (EMS de larguras ínfimas são automaticamente removidas por essa estratégia).

- **Teste2-prévio:** Se a menor dimensão do novo EMS criado é menor que a menor dimensão de cada caixa remanescente a ser carregada, não o adicione à lista S_b (Esta estratégia é importante para eliminar EMS em que as caixas remanescentes não são passíveis de carregamento, e que não foram removidas pela estratégia anterior, por não terem ínfimas larguras).

Observação

Após a compreensão acerca do funcionamento da *Estratégia de Carregamento*, é possível complementar a explicação da Etapa 2, no sentido de entender as heurísticas BBL e BLB. Estas heurísticas orientam quanto à determinaçao da posição de carregamento das caixas. Assim como descrito anteriormente, estas posições disponíveis para a alocação de itens são justamente as coordenadas mínimas (x_i,y_i,z_i) de cada espaço maximal vazio, EMS_i. Por isso, a importância em se ordenar os EMS dentro das listas S_b, para todo contêiner aberto b, a fim de estabelecer-se a ordem de prioridade com que eles devem ser selecionados.

Basicamente, na Etapa 2, se o VPH indicar a utilização da heurística BBL dentro do decodificador *Procedimento de Carregamento* (assim como apresentado na subseção 3.2.4), então ordena-se a utilização de EMS em um contêiner aberto, dentro de sua lista S_b, de forma crescente, tal que tem-se $EMS_i < EMS_j$, se uma das três condições for verificada:

$$x_i < x_j \quad \text{ou} \quad x_i = x_j \text{ e } z_i < z_j \quad \text{ou} \quad x_i = x_j,\ z_i = z_j \text{ e } y_i < y_j$$

Em seguida, o *Procedimento de Carregamento* escolhe o primeiro EMS da ordenação

crescente, tal que a caixa a ser carregada caiba dentro dele com qualquer orientação possível.

Entretanto, assim como observado por Liu e Teng [88], algumas soluções ótimas não eram possíveis de serem geradas utilizando a estratégia BBL. Para contornar este problema, no trabalho [12], os autores combinaram o BBL com uma nova heurística, chamada de Heurística BLB (como mencionado *a priori*), em que a ordenação de EMS em um contêiner acontece de tal forma que $EMS_i < EMS_j$, se uma das três condições for verificada:

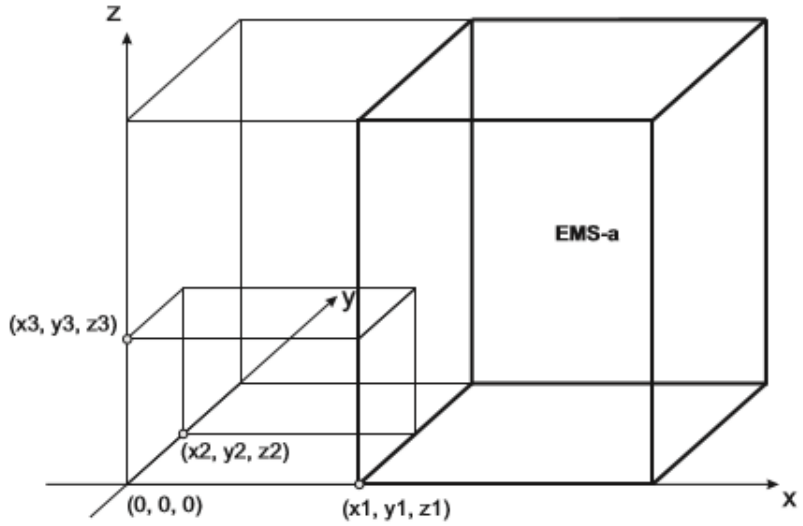
$$x_i < x_j \qquad \text{ou} \qquad x_i = x_j \text{ e } y_i < y_j \qquad \text{ou} \qquad x_i = x_j,\ y_i = y_j \text{ e } z_i < z_j$$

Novamente após a ordenação, escolhe-se o primeiro EMS, no qual a caixa a ser carregada caiba dentro dele (usando qualquer orientação possível).

Ou seja, a *Estratégia de Carregamento* utiliza duas heurísticas combinadas, a BBL e a BLB, para construir o carregamento das caixas, selecionando em qual EMS deve ser realizado o carregamento da caixa atual. O vetor VPH, descrito anteriormente, apenas será o fator aleatório que decidirá quando uma caixa deve ser carregada utilizando a heurística BBL ou a heurística BLB.

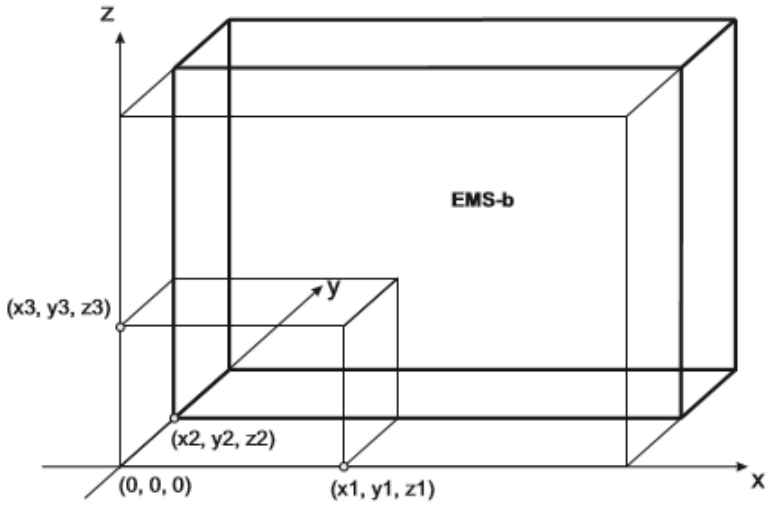
Para ilustrar o funcionamento de VPH, considere as Figuras 3.10, 3.11 e 3.12, que representam um contêiner contendo uma caixa carregada e as três configurações de EMS possíveis: EMS-a, EMS-b e EMS-c. Suponha que o Decodificador tenha de efetuar o carregamento da próxima caixa que se encontra na posição *i* do vetor BPS, e que tem um valor $gene_{n+i} \in [0, 1]$, dentro deste contêiner.

Figura 3.10 - Ilustração do Funcionamento do Vetor VPH - Parte 1.



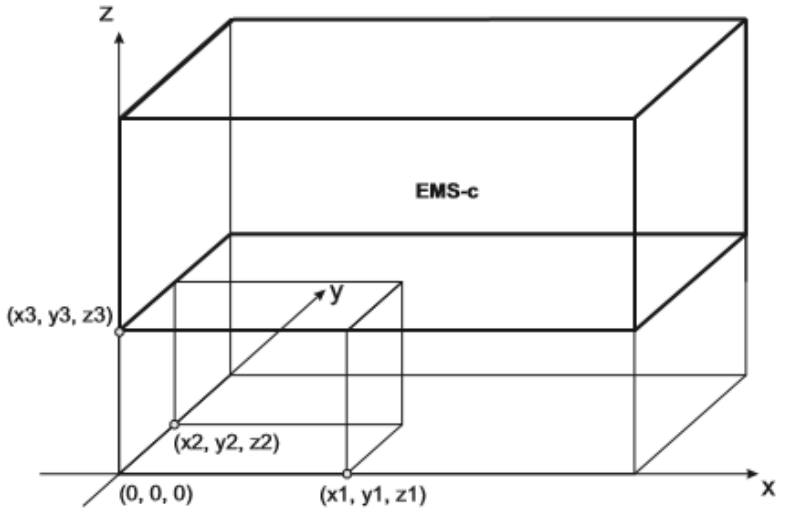
Fonte: Elaboração do próprio autor.

Figura 3.11 - Ilustração do Funcionamento do Vetor VPH - Parte 2.



Fonte: Elaboração do próprio autor.

Figura 3.12 - Ilustração do Funcionamento do Vetor VPH - Parte 3.



Fonte: Elaboração do próprio autor.

Então, a ordem de testes para o carregamento da caixa *i* irá seguir uma lista ordenada de espaços maximais vazios possíveis, no caso, três, cuja ordenação irá ser efetuada segundo o critério VPH:

- **Se gene**$_{n+i} \leq 0,5$**: Usar o critério BBL**. Isto equivale a comparar os valores dos cantos anteriores esquerdos de cada um dos três EMS projetados no eixo *x*: $x2 = x3 = 0 < x1$, isto é, o EMS-a é o último na listagem de EMS. A prioridade são os EMS que possuem posições com menores valores para x (mais ao fundo do contêiner, pensando-se, por exemplo, no carregamento de um caminhão). Daí o conceito de "canto anterior". Em

seguida, para EMS com mesmos valores de x, analisam-se os valores projetados no eixo z: para os pontos $(x2,y2,z2)$ e $(x3,y3,z3)$, que possuem $x2 = x3 = 0$, tem-se que $z2 < z3$, fazendo com que a prioridade neste subconjunto de EMS seja aquele que possui menor altura, no caso, o EMS-b (carregando caixas, por exemplo, que estão ao fundo de um contêiner, formando uma base estável, para depois pensar em carregar-se para cima). Seguindo esta lógica, na figura, o decodificador do BRKGA tenta carregar a caixa i, em ordem, nos seguintes EMS e em todas as rotações possíveis para a caixa i:

EMS-b < EMS-c < EMS-a.

- **Se gene$_{n+i} > 0,5$: Usar o critério BLB**. Novamente, a prioridade é comparar os cantos que estão mais ao fundo do contêiner, através da coordenada x: $x2 = x3 = 0 < x1$, logo, o contêiner EMS-a continua sendo o último da listagem de EMS. Na sequência, e diferentemente do BBL, a heurística BLB compara os valores projetados no eixo y: para os pontos $(x2,y2,z2)$ e $(x3,y3,z3)$, que possuem $x2 = x3 = 0$, tem-se que $y2 < y3$, fazendo com que a prioridade neste subconjunto de EMS seja aquele que está mais à esquerda do contêiner, no caso, o EMS-c (e não o de menor altura). Logo, na figura, o decodificador do BRKGA tenta carregar a caixa i, em ordem, nos seguintes EMS e em todas as rotações possíveis para a caixa i:

  EMS-c < EMS-b < EMS-a.

Etapa 5: Cálculo da *Fitness*

Na Etapa 5, o novo cálculo da função objetivo, proposto por Resende e Gonçalves [12], visa quantificar a qualidade do carregamento realizado, dado que vários carregamentos podem possuir a mesma quantidade NB (do inglês, *number of bins*) de contêiners, porém com qualidades diferentes de carregamento. Essa qualidade é mensurada de acordo com o carregamento que possui um contêiner com o maior espaço vazio encontrado, uma vez que, se um contêiner possui o maior espaço não utilizado, então será aquele que terá maior possibilidade de ser preenchido com novas caixas em uma eventual necessidade. Assim, com essa nova perspectiva, adicionou-se à função objetivo (que retornava apenas o valor NB) o que se chamou de *qualidade de medida*, fazendo surgir uma nova função objetivo, que foi nomeada pelos autores por *Número Ajustado de Contêiners* (do inglês, *Ajusted Number of Bins*): αNB.

A nova *fitness* αNB combina então o NB (número de contêiners) com uma nova *qualidade de medida* (que está contida no intervalo]0, 1[) que é dada através de uma razão entre dois

valores: o *LeastLoad* e o *BinCap*. O *LeastLoad* é um valor variável, com variação 0 < *LeastLoad* <*BinCap*, o qual indica o volume do carregamento do contêiner de carregamento mínimo (dentre todos os contêiners utilizados), e *BinCap* é um valor fixo que representa a capacidade de cada contêiner: *BinCap*=*W* × *H* × *D*, para o caso tridimensional e *BinCap*=*W* × *H*, para o caso bidimensional. Logo, o *Número Ajustado de Contêiners* é dado por:

$$\alpha NB = NB + \frac{LeasLoad}{BinCap}$$

Observação: A razão $\frac{LeasLoad}{BinCap}$ indica o percentual de ocupação do contêiner com menor volume utilizado (de carregamento mínimo), e que faz com que, dentre duas possibilidades de solução que utilizam o mesmo número de contêiners, escolha-se aquele que obtém menor razão, ou seja, menor ocupação. Dado que *BinCap* é um valor fixo, pode-se considerar ainda que a solução que obtiver um menor *LeastLoad*, será aquela considerada com maior potencial de melhoramento, já que terá maior espaço vazio para carregar mais caixas, em um possível recarregamento.

3.2.5 Decodificador Procedimento de Carregamento ± Visão Geral

Em síntese, dado um cromossomo de 3*n* chaves aleatórias, com valores contidos no intervalo [0, 1], o decodificador *Procedimento de Carregamento*, dentro do algoritmo BRKGA, transforma este cromossomo em um carregamento factível para o problema original, com *fitness* associada. Combinando os vetores BPS, VPH e VBO definidos pelo BRKGA, as listas S_b de espaços-maximais vazios, para cada container *b* aberto, e as heurísticas de carregamento *BBL* e *BLB*, cada estágio de desenvolvimento inclui necessariamente os seis passos a seguir:

1. Seleção de caixas;
2. Seleção da heurística de carregamento;
3. Seleção do espaço-maximal vazio e do contêiner;
4. Seleção da orientação da caixa;
5. Carregamento da caixa;
6. Atualização das informações.

A *Seleção de caixas* é realizada através da seleção da i-ésima posição do BPS. A *Seleção da heurística de carregamento* é proveniente da estratégia de seleção das coordenadas do VPH. A *Seleção do espaço-maximal vazio e do contêiner* é realizada para um espaço maximal em que a caixa associada ao BPS_i caiba dentro dele (usando a heurística já previamente selecionada pelo VPH e incluindo qualquer orientação possível). O processo de busca por contêiners (em que a caixa BPS_i caiba dentro dele) pára quando um contêiner é encontrado. Se nenhum contêiner é encontrado, então um novo contêiner é aberto. Após um contêiner e um espaço maximal serem selecionados, então a *Seleção da orientação da caixa* é realizada através do vetor VBO. A escolha da orientação da caixa dentre as *k* possíveis orientações ($1 \leq k \leq 6$ para o 3D-BPP e $1 \leq k \leq 2$ para o 2D-BPP, $k \in Z$) é escolhida aleatoriamente. O *Carregamento da caixa* consiste, portanto, em carregar uma caixa BPS_i dentro de um espaço-maximal vazio e contêiner que foram selecionados. Finalmente, os dados são atualizados pelo passo 6, *Atualização das informações*, no que concerne à lista de espaços-maximais vazios dentro do contêiner utilizado para carregar a última caixa (utilizando o *Processo DP*).

3.2.6 Algumas considerações do decodificador do BRKGA para o caso bidimensional

Para iniciar a implementação computacional do BRKGA aplicado ao 3D-BPP, foi necessário, primeiramente, compreender o problema bidimensional, 2D-BPP. Para isto, algumas adaptações foram realizadas para concretizar o estudo, porém, é importante ressaltar que as estruturas tridimensionais foram mantidas desde o início, mantendo-se a terceira coordenada igual a zero, com vistas à futura implementação para o caso tridimensional.

Decodificação do vetor heurística de carregamento ± VPH

Dado que as coordenadas em z dos EMS são sempre iguais a zero, não existe a necessidade de se utilizar duas heurísticas distintas: BBL ou BLB. Para ordenar os EMS, basta considerar a seguinte comparação: $EMS_i < EMS_j$, se uma das duas condições for verificada:

$$x_i < x_j \qquad \text{ou} \qquad x_i = x_j \ \text{ e } \ y_i < y_j$$

Ou seja, o vetor VPH deixa de ter finalidade para o caso bidimensional, pois a heurística que determina a ordenação de EMS é única. Assim, o segundo conjunto de *n* coordenadas do cromossomo poderia ser desconsiderado, e os cromossomos poderiam ter $2n$ chaves aleatórias, a

primeira, correnpondendo ao BPS e, a segunda, ao VBO. Porém, foi mantida a mesma estrutura tripla dos cromossomos para a adaptação futura para o caso tridimensional.

Decodificação do vetor orientação da caixa ± VBO

Em seguida à ordenação e seleção de EMS, o decodificador *Procedimento de Carregamento* escolhe o primeiro EMS da ordenação crescente, tal que a caixa a ser carregada caiba em pelo menos uma das orientações. Neste momento, é realizada a decodificação do VBO, para escolher a primeira orientação da caixa a ser carregada naquele EMS escolhido. Considerando como orientação 1 a orientação original da caixa, e como orientação 2, a caixa rotacionada em 90 graus, então, para:

$$VBO_i = gene_{2n+i}$$

com $i = 1,...,n$, a escolha da primeira tentativa de orientação da caixa é realizada aleatoriamente através da relação $|VBO_i \times nBO|$, onde $nBO = 2$ é o número de orientações possíveis. Se $gene_{2n+i} \leq 0,5$, escolhe-se a orientação 1, e, se $gene_{2n+i} > 0,5$, então, a primeira tentativa de carregamento deve ser realizada com orientação 2.

Estratégia de Carregamento

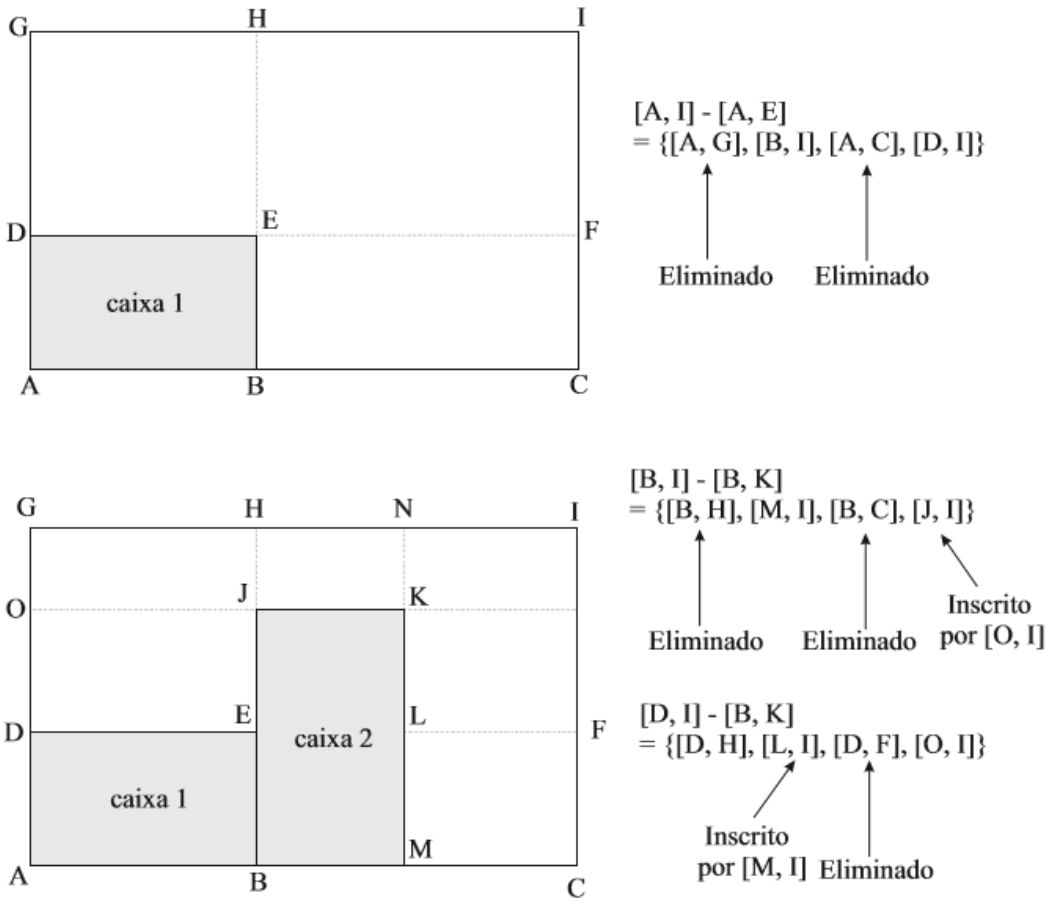
Para o 2D-BPP, excluem-se os dois últimos EMS do conjunto DP de seis EMS gerados pelo *Processo DP*. A Figura 3.13, adaptada do trabalho de Lai e Chan [20], ilustra o DP para o caso bidimensional, incluindo os Testes de Eliminação 1 e 2. Os EMS da figura no qual constam a palavra "Inscrito", foram excluídos de acordo com o Teste 1, e, onde destaca-se a palavra "Eliminado", são EMS que foram exluídos da lista de acordo com o Teste 2. Os pontos mínimos e máximos dos EMS são apresentados em termos de letras maiúsculas, ao invés de suas coordenadas.

Na primeira parte do exemplo, uma caixa foi carregada no contêiner bidimensional, em formato de retângulo, na origem. Devido ao fato de que o próprio contêiner era o único EMS disponível, ele foi o selecionado, implicando na necessidade de se aplicar o *Processo DP*, $[A, I] - [A, E]$ (EMS selecionado menos caixa carregada). O resultado da Fase 1 forneceu a geração dos quatro EMS: $[A, G]$, $[B, I]$, $[A, C]$ e $[D, I]$. O primeiro e terceiro EMS gerados algebricamente pelas projeções foram eliminados pelo Teste 2, na Fase 2 do *Processo DP*, por possuírem larguras ínfimas.

Na segunda parte do exemplo, houve a seleção do EMS $[B, I]$ para a alocação da segunda

caixa, [*B, K*]. Como a caixa sobrepôs o EMS não selecionado, [*D, I*], então, aplicou-se o *Processo DP* para excluir o espaço da caixa em ambos os espaços maximais vazios, no selecionado [*B, I*], e no não selecionado sobreposto, [*D, I*]: [*B, I*] − [*B, K*] e [*D, I*] − [*B, K*]. Foram gerados oito novos EMS, respectivamente a cada um dos processos, exluindo-se da lista anterior os dois EMS que tiveram seus espaços alterados pela colocação da segunda caixa. Após a aplicação dos testes de eliminação, apenas três EMS permaneceram na lista S_b relacionada a este contêiner: [*M,I*],[*D, H*] e [*O,I*]. Se houvesse outros EMS provenientes da lista S_b anterior, além dos dois existentes na primeira parte do exemplo, que não tivessem sido sobrepostos pela colocação da segunda caixa, então, eles seriam copiados juntamente a estes três novos EMS gerados.

Figura 3.13 - O *Processo DP* aplicado ao carregamento bidimensional.



Fonte: Adaptado de Lai e Chan (1997) [20].

Cálculo da Fitness

Para o caso bidimensional, o valor constante *BinCap* adaptado refere-se à área de cada contêiner, ou seja, $BinCap = W \times H$. O valor variável, *LeastLoad*, será representado pelo somatório

das áreas das caixas carregadas no contêiner de carregamento mínimo.

3.3 O ALGORITMO GENÉTICO DE CHU-BEASLEY

O algoritmo genético de Chu-Beasley, *Heurística AG*, foi proposto por Chu e Beasley em 1996, em seu trabalho [21], para resolver Problemas de Atribuição Generalizada (GAP, do inglês, *Generalised Assigment Problem*).

O GAP também é demonstrado na literatura como sendo do tipo NP-difícil, sendo, portanto, considerado como pertencente à categoria dos problemas mais complexos para se apresentar soluções ótimas através de algoritmos exatos. Dessa forma, a garantia de otimalidade através de procedimentos exatos são indicados apenas para a solução de alguns problemas específicos, e, em geral, de pequeno porte. Assim como para o PCC, heurísticas e meta-heurísticas vêm sendo desenvolvidas no decorrer das últimas décadas com a finalidade de apresentar boas soluções aproximadas para estes problemas.

Dentre estes estudos, a meta-heurística AG é a que desperta maior interesse para o presente trabalho, pois demonstrou superioridade nos resultados computacionais apresentados para o GAP, se comparada a outras heurísticas, e, principalmente, por ser um algoritmo genético que contém estratégias que ainda não foram aplicadas nos Problemas de Carregamento de Contêiners. Assim, o objetivo do estudo desta seção foi entender os mecanismos implementados no desenvolvimento do Algorimo Genético de Chu-Beasley, com o objetivo de assimilar novas e distintas estratégias de se trabalhar com algoritmos genéticos aplicados ao 3D-BPP.

3.3.1 O problema de atribuição generalizada

O Problema de Atribuição Generalizada objetiva minimizar o custo de se atribuir n tarefas a m agentes, no qual, cada tarefa j, $j=1, ..., n$, possui um custo c_{ij}, se for executada pelo agente i, $i=1, ..., m$, e tal que pode ser executada apenas por um único agente. Cada agente i demanda r_{ij} recursos para executar uma tarefa j, e uma quantidade máxima de b_i recursos pode ser designada a um agente i, para a execução de uma ou mais tarefas.

O GAP padrão, contendo os pressupostos supradescritos, pode ser modelado pela formulação matemática (3.3) a (3.6).

A *fitness* (3.3) minimiza o somatório dos custos de se atribuir tarefas à agentes, dado que as variáveis de decisão x_{ij} são binárias e iguais a 1, caso a tarefa j seja atribuída ao agente i, e iguais a zero, caso contrário. O conjunto de restrições (3.4) garante que cada tarefa j será realizada por

apenas um agente *i*. As restrições (3.5) impõem a condição de que, para um determinado agente *i*, o somatório dos recursos r_{ij} a ele atribuídos para realizar tarefas distintas, não seja superior à sua capacidade máxima b_i.

$$\text{minimizar} \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{3.3}$$

sujeito a:

$$\sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J \tag{3.4}$$

$$\sum_{i \in I} r_{ij} x_{ij} \leq b_{ij} \qquad \forall i \in I \tag{3.5}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in I, \ \ \forall j \in J \tag{3.6}$$

3.3.2 A heurística AG de chu-beasley

O Algoritmo Genético de Chu-Beasley conta com a estrutura básica dos algoritmos genéticos, considerando cada cromossomo como um vetor *n*-dimensional. A codificação deste vetor baseia-se na representação dos *m* agentes e *n* tarefas, através dos conjuntos $I = \{1,2,...,m\}$ e $J = \{1,2,...,n\}$, respectivamente, da seguinte forma: cada posição j ($j \in J$) do cromossomo representa a tarefa que deve ser executada, e o valor i ($i \in I$) é atribuído a cada uma destas componentes, indicando qual agente *i* executará a tarefa j. Na Tabela 3.1 apresenta-se um exemplo de cromossomo codificado em que a tarefa 1 foi atribuída ao agente 2, a tarefa 2, ao agente 1, a tarefa 3, ao agente 3, e assim, sucessivamente.

Tabela 3.1 - Cromossomo codificado da Heurística AG

tarefa	1	2	3	4	5	...	n-1	n
agente	2	1	3	m	3	...	1	2

Fonte: Adaptado de Chu e Beasleu (1997) [21].

O Algoritmo AG de Chu-Beasley, portanto, possui cromossomos com *n* genes inteiros, que é igual ao número de tarefas, e o qual é dividido em 5 etapas, segundo a descrição do trabalho [21], adaptado como a seguir:

1. **Gerar uma população inicial de *N* soluções randomicamente constituídas.** Cada uma

das soluções iniciais é gerada de acordo com uma atribuição randômica de um agente a cada tarefa. Como as soluções iniciais podem violar as restrições de capacidade em (3.5), soluções iniciais podem ser infactíveis. Seja s_{kj} um índice que representa o agente atribuído à tarefa j ($j = 1,...,n$) na solução k ($k = 1,...,N$).

2. **Decodificar a solução codiftcada para obter o valor *fitness* associado.** O valor *fitness* é calculado para cada proposta de solução conhecida. Como as soluções podem violar as restrições de capacidade (3.5), soluções iniciais podem ser infactíveis. Logo, deve-se levar em consideração não apenas os custos de uma solução, mas também o seu grau de infactibilidade. Ao contrário de outras propostas existentes na literatura sobre algoritmos genéticos, a proposta de Chu-Beasley armazena a informação de infactibilidade em um vetor independente chamado de *unfitness*. Os autores utilizaram a heurística proposta por eles, no trabalho anterior [89], para a decodificação dos cromossomos. A *fitness* f_k da solução k é igual ao valor da função objetivo e que é calculada observando-se apenas o somatórios dos custos:

$$f_k = \sum_{j \in J} c_{s_{kj} j}$$

A *unfitness* u_k é uma medida de infactibilidade e é calculada por:

$$u_k = \sum_{i \in I} \max[0, (\sum_{j \in J, s_{kj} = i} r_{ij}) - b_i]$$

Note que $u_k = 0$ se e somente se k é factível.

3. **Selecionar duas soluções pais para a reprodução.** Os autores utilizaram o método binário de seleção por torneio, em que dois indivíduos são escolhidos aleatoriamente da população. O indivíduo que possui melhor *fitness* (o menor valor para a função objetivo), dentre os dois escolhidos, será o pai selecionado para a *Recombinação*. Para gerar um descendente, dois torneios binários são realizados. Cada um deles produz um dos pais. Note que o critério de seleção não envolve o valor *unfitness* do pai.

4. **Gerar uma solução ftlho aplicando primeiramente um operador *Recombinação* para os pais selecionados.** Foi considerado o operador *Recombinação* simples de *um-ponto*, no qual um ponto da recombinação $p \in J$ (uma tarefa, uma componente do vetor solução k) é randomicamente selecionado e a solução filho será constituída dos primeiros p

genes do primeiro pai, e os genes $n-p$ genes remanescentes, do segundo pai, ou vice-versa, com iguais probabilidades. O procedimento de *Recombinação* tem na sequência um procedimento de *Mutação*. Esse procedimento de *Mutação* consiste em modificar elementos em dois genes aleatoriamente selecionados (i.e., modificando a atribuição de agentes a duas tarefas selecionadas randomicamente). A experiência computacional dos autores, que é descrita no decorrer do artigo, mostrou que a Heurística AG é eficiente na geração de soluções de boa qualidade, ao considerar apenas os operadores de *Recombinação* e de *Mutação*. Mas o algoritmo pôde ser ainda melhorado utilizando-se um novo operador, que os autores chamaram de *Heurística de Problema-Específico*, e que envolve a melhoria local através de dois passos, como descritos abaixo.

(a) Seja T_j o agente atribuído à tarefa j na solução filho. Para cada agente $i \in I$, se a capacidade de recurso do agente i é excedida, ou seja, se:

$$\sum_{j \in J, T_j = i} r_{ij} > b_i$$

então uma simples tarefa q selecionada randomicamente e com $T_q = i$ é reatribuída do agente i para o próximo agente (na seguinte ordem: $i+1, ..., m, 1, 2, ..., i-1$) que tem recurso suficiente para executá-la (não extrapola sua capacidade máxima de recurso), se ele puder ser encontrado.

(b) Para cada tarefa $j \in J$, procurar pelo menor custo de recursos de se executar cada tarefa j, $min_{i, i \in I} c_{ij}$, considerando-se todos os agentes $i \in I$, que satisfaçam ambas as condições: $c_{ij} < c_{T_j j}$ e:

$$\left(\sum_{q \in J, T_q = i} r_{iq} \right) + r_{ij} \leq b_i,$$

se i puder ser determinado, e então reatribuir a tarefa j do agente T_j para o agente i. A heurística (a) é uma tentativa de melhoramento da factibilidade dos descendentes, ao reatribuir tarefas de agentes que tem sua capacidade máxima violada, para agentes que ainda possuem a disponibilidade de a ela serem designadas tarefas. A heurística (b) tenta ainda reduzir os custos dos filhos gerados, ao designar tarefas a agentes que possuem menores custos.

5. **Substituir um indivíduo da população por uma solução ftlho.** Na heurística GA de Chu-Beasley, a evolução da população anterior para a próxima geração consiste na

substituição do indivíduo da população com o maior valor *unfitness* (i.e., a solução mais infactível) pelo descendente gerado. Se a população possui apenas soluções factíveis ($u_k = 0, \forall k$), então, substitui-se pelo filho, a solução com maior *fitness*. Este esquema de substituição de soluções na evolução entre populações promove a eliminação de soluções infactíveis. Além disso, descendentes em duplicidade são impedidos de entrar na população atual, para evitar que uma população seja constituída apenas de soluções idênticas, por exemplo, e limitando a capacidade do algoritmo de gerar novas soluções.

6. **Repetir os passos 3-5 até que M ftlhos não duplicados sejam gerados sem que haja o melhoramento da melhor solução até então encontrada.**

3.3.3 Análise entre BRKGA e heurística AG

O objetivo desta subseção é apontar algumas diferenças entre as estruturas dos algoritmos genéticos BRKGA aplicado ao PCC e o de Chu-Beasley aplicado ao GAP, sem ter a intenção de compará-los, uma vez que são aplicados a problemas distintos. Este apontamento visa apresentar a diferença entre os algoritmos posteriormente implementados, e como as diferentes estratégias, mescladas, puderam constituir um novo algoritmo.

- **Soluções infactíveis**

Na Heurística AG de Chu-Beasley, o valor *unfitness* u_k de uma solução infactível é um valor positivo, que fornece o somatório de todos os recursos necessários utilizados pelo agente i ($s_{kj} = i$) ao executar tarefas j que a ele foram atribuídas e que estão dispostas em uma solução k. Esse somatório de recursos é maior do que a capacidade máxima b_i de recursos que podem ser dispendidos ao agente i, tornando a diferença, portanto, positiva. Porém, caso a solução k seja factível, então esta diferença é negativa, e $u_k = 0$. Desde que soluções infactíveis podem aparecer no decorrer da heurística, o algoritmo diferencia soluções factíveis de *fitness*, f_k, das não factíveis *unfitness* u_k, gerando uma penalização de acordo com o grau de infactibilidade da solução gerada. Essa preocupação, entretanto, não é necessária no BRKGA aplicada ao PCC, uma vez que o decodificador apresentado transforma qualquer solução gerada em uma solução factível ao problema original.

- **A seleção das soluções geradoras de novas soluções**

Na Heurística AG de Chu-Beasley, os pais são selecionados de forma randômica dentre os elementos de toda a população atual. Não há uma probabilidade maior de se escolher os pais que

tendem a produzir melhores filhos. Também não há uma separação na população, classificando-a em população elite, não-elite ou mutante, como acontece no BRKGA, e nem estratégias para a seleção dos pais. Quatro possíveis soluções pais são selecionadas ao acaso e, dois a dois, são realizados dois torneios; após, escolhe-se então os vencedores de cada um destes, que serão as soluções pais geradoras de soluções para a próxima população. Essa aleatoriedade tem como consequência a perda da informação de boas soluções que já foram obtidas, e que podem estar próximas à melhor solução possível de ser determinada, porém, evitam um grande problema enfrentado pelos algoritmos genéticos: a geração de boas soluções cada vez mais repetidas ou com pouco grau de variação.

- **O operador Recombinação**

A *Recombinação* (*crossover*) no algoritmo AG de Chu-Beasley é completamente diferente do que acontece no BRKGA. Na Heurística AG, cria-se um descendente através de uma composição de duas sequências de genes provenientes das soluções pais, uma sequência de cada pai. Cada sequência é composta exatamente com a mesma sequência existente na solução pai; no caso do primeiro pai, são copiados para o descendente filho seus p primeiros genes, e no caso do segundo pai, são copiados ao filho, seus $n-p$ genes finais, i.e., do gene $p+1$ ao gene n. O ponto de troca p da cópia das sequências do primeiro para o segundo pai, é um número randomicamente gerado no intervalo $[1,n]$. Já no BRKGA, considera-se um parâmetro ρ (em geral igual a $0,7$), que representa a probabilidade viciada de se copiar os genes, um a um (e não dois fragmentos de sequências), do descendente elite (70%) ou do descendente não-elite (30%).

- **O processo de Mutação**

A *Mutação* também acontece de maneira muito distinta em ambos os algoritmos considerados. Para Chu-Beasley, em cada solução filho gerado, escolhe-se dois genes de maneira aleatória, e troca-se o valor das componentes segundo uma estratégia, i.e., trocando a atribuição de um agente a duas tarefas específicas. No BRKGA, a mutação consiste em gerar uma subpopulação completamente nova, de forma randômica, e que permite o aparecimento de novas soluções.

- **A evolução da população**

As transições entre ciclos geracionais também se mostram muito diferentes nos dois algoritmos genéticos, pois, na Heurística AG, a evolução é feita substituindo-se o indivíduo de pior qualidade da população corrente, pelo descendente gerado, realizando a troca de apenas uma solução por iteração. No BRKGA, toda uma nova população é gerada a cada iteração, copiando-se apenas os

indivíduos elite, após a ordenação dentre as melhores soluções.

É ainda importante ressaltar que o operador *Heurística de Problema-Específico* é uma melhoria importante para problemas que lidam com o tratamento de infactibilidades, o que não é o caso de algoritmos genéticos que resolvem Problemas de Carregamento de Contêiners. Além disso, o decodificador da Heurística AG não foi apresentado, uma vez que sua estrutura depende do problema original e não do algoritmo em si, e o presente trabalho não objetiva trabalhar com problemas GAP.

4 IMPLEMENTAÇÃO E RESULTADOS DO BRKGA

4.1 INTRODUÇÃO

Neste capítulo, apresenta-se o desenvolvimento do estudo aprofundado do BRKGA, principalmente no que se refere ao decodificador *Procedimento de Carregamento*, e sua implementação computacional. Primeiramente, é realizado o estudo do algoritmo contido nos experimentos computacionais dos artigos de Resende e Gonçalves, [12] e [17]. Após, apresenta-se o problema bidimensional da literatura que norteou o entendimento do decodificador *Procedimento de Carregamento*, antes da implementação propriamente dita. Testes computacionais iniciais foram então realizados para a comprovação do funcionamento do BRKGA implementado, para o caso bidimensional. Para finalizar o capítulo, foi realizada a implementação para problemas do tipo 3D-BPP de grande porte, e os resultados foram comparados aos obtidos por Resende e Gonçalves [17], utilizando-se os mesmos exemplos desta referência.

4.2 CONSIDERAÇÕES INICIAIS

De acordo com Resende e Gonçalves [12], a diferença na forma de seleção das soluções geradoras entre o RKGA e o BRKGA tem produzido consideráveis melhorias nos resultados computacionais, inclusive quando se trata da aplicação destes algoritmos para o 3D-BPP. Isto porque no RKGA, há grandes possibilidades de que os pais geradores não sejam aqueles que obtiveram as melhores soluções na geração anterior. O BRKGA impõe de maneira desequilibrada (viciada) que as melhores soluções da geração anterior tenham maiores chances de passar suas características para seus descendentes, por elitismo, por Recombinação PUC e através da condição de que pelo menos um dos pais geradores seja do grupo elite.

Além disso, uma vantagem muito particular na implementação do BRKGA, é que devido à sua estrutura estar dividida em partes dependente e independente, a implementação também é realizada em duas partes bem distintas, separadamente. Uma delas, que se refere à implementação computacional da parte independente, pode ser utilizada em quaisquer tipos de problema, pois a busca no espaço definido pelo hipercubo unitário n-dimensional não faz relação com a estrutura do problema inicial. Assim, uma vez implementada, pode ser reaproveitada para outros problemas de otimização combinatória e a implementação computacional restante estará resumida à implementação da parte dependente: a do Decodificador.

A implementação computacional do BRKGA pode ser realizada em cinco passos, além da

fase inicial de definição dos parâmentros, nomeada pelo presente trabalho de "Passo 0". O detalhamento está descrito a seguir, considerando-se problemas de otimização combinatória do tipo minimização e é uma síntese das ideias apresentadas no trabalho [12].

4.2.1 Implementação computacional do BRKGA ± sem incluir a descrição da parte dependente (decodificador)

- *Passo 0 (Definição dos Parâmetros Iniciais)*: Definem-se os parâmetros iniciais: n, p, p_e, p_m e ρ_e. Uma população inicial de vetores de coordenadas com valores em $[0,1]$ deve ser gerada e armazenada na matriz A_{pop} de dimensão $p \times 3n$, em que a i-ésima linha da matriz corresponde ao i-ésimo indivíduo da população, para todo $i = 1,...,p$. Em seguida, utiliza-se o Decodificador (que é particular a cada tipo de problema de otimização) para decodificar cada indivíduo contido em cada uma das linhas da matriz A_{pop} (em formato codificado) como uma solução factível do problema original, obtendo, também do Decodificador, os valores de *fitness* associados a cada proposta de solução e armazenando-os em um vetor $f_{fitness}$. Verifica-se se o critério de parada foi satisfeito. Caso não o tenha sido, inicia-se o processo geracional contido nos Passos 1 a 5, descritos a seguir.

- *Passo 1 (Reordenação)*: Reordenam-se as componentes do vetor *fitness* de forma crescente e as linhas de A_{pop} de acordo com a ordenação dos *fitness*. Observação: Não é necessário remover as linhas de A_{pop} para a reordenação, apenas um rearranjo de suas posições.

- *Passo 2 (Recombinação)*: Geram-se $p - p_e - p_m$ novas soluções por Recombinação através de $p - p_e - p_m$ pares de pais geradores. O gerador elite de cada solução obtida terá um índice, que é um número aleatório inteiro no intervalo $[1, p_e]$. O outro gerador (que pode ser elite ou não), terá também índice inteiro e aleatório, no intervalo $[p_e+1, n]$ (Estratégia 1) ou no intervalo $[1, n]$ (Estratégia 2). O i-ésimo descendente gerado é armazenado na i-ésima linha da matriz temporária B_{temp}, de dimensão $(p-p_e-p_m) \times n$. Para gerar um descendente, que serão armazenados em B_{temp}, primeiramente, geram-se os números aleatórios $r_1, r_2, ..., r_n$, no intervalo [0, 1]. Para cada $j = 1, ..., n$, se:

$$r_j \leq \rho_e,$$

então copia-se para a coordenada j do descendente, a coordenada j do pai elite. Caso contrário, copia-se a coordenada j da outra solução geradora.

- *Passo 3 (Mutação)*: Geram-se p_m novas soluções, chamadas soluções mutantes, de dimensão n. Cada solução mutante i é armazenada na linha $p - p_m + i$ da matriz A_{pop}, para $i = 1, ..., p_m$.

- *Passo 4:* Esta etapa completa a população $k + 1$. Para tal, copiam-se os $(p - p_e - p_m) \times n$ elementos da matriz B_{temp} nas linhas $p_e + 1, ..., p - p_m$ da matriz A_{pop}.

- *Passo 5:* Calcula-se a função objetivo de cada uma das novas propostas de solução armazenadas nas linhas $p_e + 1, ..., p$ de A_{pop}, atualizando as posições $p_e + 1, ..., p$ do vetor $f_{fitness}$ pelos valores obtidos nesta etapa.

Observação 1: O processo de elitismo está implícito no *Passo 1*, uma vez que as primeiras linhas da matriz A_{pop}, que é a população que será atualizada, não modifica suas primeiras linhas, cujos elementos são os que fornecem as melhores soluções (que fornecem *fitness* de menor valor, para problemas de minimização).

Observação 2: No *Passo 5*, não foi necessário calcular os valores de função objetivo das soluções das linhas $1, ..., p_e$ de A_{pop}, pois o cálculo foi realizado na geração anterior, já que são as soluções do grupo elite, copiadas da iteração passada.

Observação 3: Este ciclo de cinco passos (*Ciclo Geracional*) é repetido até que o critério de parada seja satisfeito. Os critérios de parada mais comuns são: parar o processo depois de atingido um número máximo de iterações previamente especificado, parar após a solução incumbente não melhorar depois de um número fixo de iterações, parar o processo após atingir-se um tempo limite máximo de processamento, ou após atingir uma solução de qualidade melhor que um valor previamente estipulado.

4.2.2 A escolha dos parâmetros iniciais no passo 0 da implementação do BRKGA

Assim como em toda meta-heurística, a calibragem dos parâmetros iniciais do *Passo 0* do BRKGA é realizada de forma empírica, através da análise dos dados e resultados obtidos após variações dos valores iniciais. Testes computacionais envolvendo vários tipos de problemas fizeram com que Resende e Gonçalves, em seu trabalho [12], recomendassem as sugestões de parâmetros contidas na Tabela 4.1.

Tabela 4.1 - Valores recomendados para os parâmetros do BRKGA

Parâmetro	Especificação	Valores recomendados
p	Tamanho da população	$p = an$, em que $1 \leq a \in \mathrm{R}$ é uma constante que depende do comprimento de uma proposta de solução codificada pelo vetor de chaves aleatórias
p_e	Número de soluções do grupo elite	$0,10p \leq p_e \leq 0,25p$
p_m	Número de soluções mutantes	$0,10p \leq p_m \leq 0,30p$
ρ_e	Probabilidade de herdar a informação da solução geradora elite	$0,5 \leq \rho_e \leq 0,8$

Fonte: Adaptado de Resende e Gonçalves (2013) [12].

4.2.3 Formação da população inicial no BRKGA

A forma tradicional dos Algoritmos Genéticos de se gerar uma população inicial também é aplicada ao BRKGA: puramente randômica. Os valores aleatórios gerados são armazenados na matriz A_{pop}, do *Passo 0* da implementação computacional. No caso particular do BRKGA aplicado ao 3D-BPP, cada elemento da matriz A_{pop} é um valor aleatório contido do intervalo $[0,1]$. Entretanto, alguns algoritmos genéticos especializados geram pelo menos uma parcela de sua população inicial utilizando procedimentos heurísticos, o que pode potencializar e agilizar o processo de obtenção de uma solução próxima ou igual à ótima. Um exemplo dessa ideia está contida no trabalho de Buriol *et al.* [90], que utiliza o BRKGA para resolver problemas de roteamento. Com relação ao 3D-BPP, Resende e Gonçalves [18] propõem que uma única solução seja gerada utilizando um algoritmo heurístico, e demais soluções que completam a populaçao inicial, geradas de forma aleatória. Porém, a tentativa de estimar a população inicial de uma forma inteligente pode provocar uma complexidade computacional que supere os ganhos desta relação custo-benefício. Recomenda-se uma análise prévia que verifique estas questões advindas da possível inserção de uma heurística que interfira na aleatoriedade da formação da população inicial.

4.3 IMPLEMENTAÇÃO COMPUTACIONAL DO BRKGA PARA O CASO BIDIMENSIONAL

4.3.1 O problema-teste bidimensional

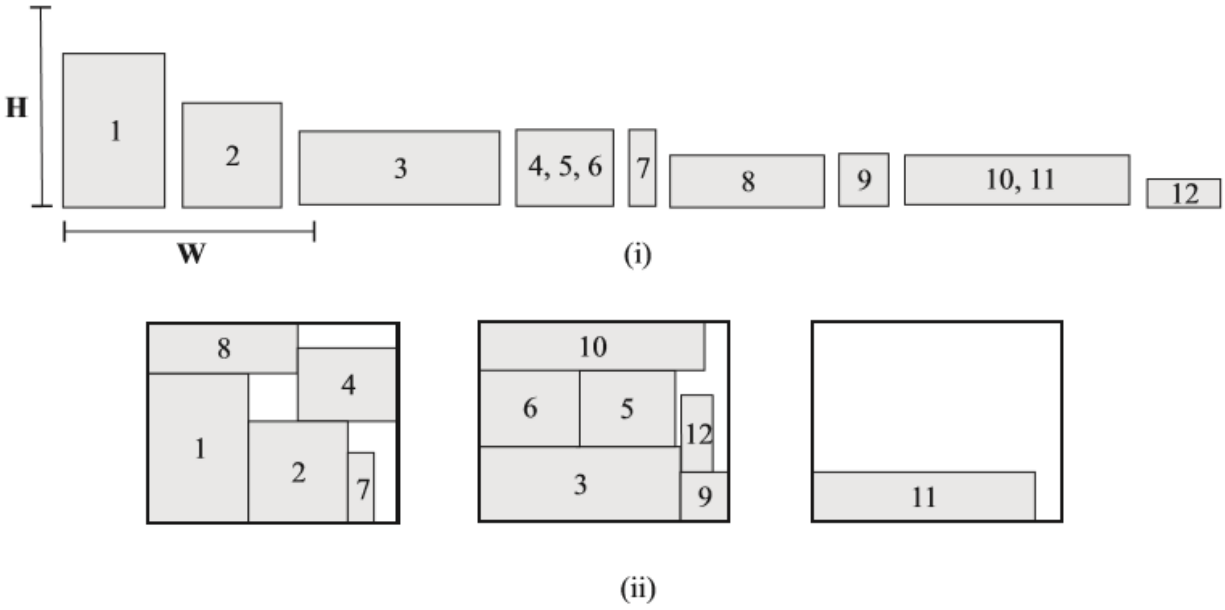
Para iniciar com os testes computacionais, considerou-se o exemplo do 2D-BPP de se

carregar dentro de contêiners de dimensões idênticas a $(W,H) = (10,8)$, 12 caixas de dimensões (w_i, h_i), $i = 1, ...,12$, dadas por:

Caixa	1	2	3	4, 5, 6	7	8	9	10, 11	12
Dimensões	(4,6)	(4,4)	(8,3)	(4,3)	(1,3)	(6,2)	(2,2)	(9,2)	(3,1)

Na Figura 4.1 apresentam-se em (i) os formatos e orientações originais das caixas a serem carregadas, e em (ii) um exemplo de carregamento.

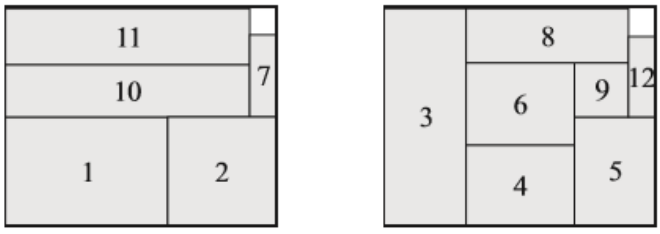
Figura 4.1 - Dados e exemplo de carregamento para o *Problema-Teste.*



Fonte: Adaptado de Lodi et al. (2012) [74].

Uma solução ótima para o exemplo é determinada pela seguinte configuração:

Figura 4.2 - Carregamento ótimo para o *Problema-Teste*.



Fonte: Elaborado pelo próprio autor.

Dado que os dois contêiners utilizados possuem área não útilizada igual a 1 u.a., então a *fitness* ótima procurada é igual a:

$$\alpha NB = 2 + \frac{1}{10 \times 8} = 2{,}0125$$

4.3.2 Entendendo o decodificador

O objetivo desta subseção é apresentar o funcionamento do decodificador *Procedimento de Carregamento* de maneira detalhada, que foi realizado como o primeiro estudo que antecedeu a construção do código do BRKGA. Para isso, foi desenvolvido o início da estratégia do algoritmo de Lai e Chan [20] (parte dependente do BRKGA) aplicada ao *Problema-Teste*, considerando-se o carregamento das cinco primeiras caixas. Da maneira como é exposto, a seguir, é possível compreender como um cromossomo de chaves aleatórias que adentra o Decodificador transforma-se em um carregamento real para o problema original. Assim como especificado anteriormente, apesar de tratar-se do caso bidimensional, as coordenadas em R^3 foram mantidas, com as coordenadas em z iguais a zero, visando à posterior generalização da programação para três dimensões.

Iteração 1:

Alguns valores iniciais foram fixados, para iniciar a primeira iteração. O BPS foi inicializado como (1,3,10,11,2,4,5,6,8,9,7,12), o VPH foi desconsiderado por se tratar de um caso bidimensional, e o VBO foi fixado com valores menores que 0.5 (o algoritmo irá promover o carregamento, primeiramente, sempre com as caixas em sua posição original). Apenas um contêiner inicial está aberto, com a lista de EMS contendo um único EMS: $S_1 = \{[(0, 0, 0), (10, 8, 0)]\}$. Ou seja, o primeiro EMS é o próprio contêiner. A primeira caixa, caixa 1, deve ser carregada no único EMS da lista S_1 (Figura 4.3), na posição $(0,0,0)$. Como o EMS foi escolhido para se efetuar o carregamento, nele é realizado o *Processo DP*, que irá gerar outros novos EMS:

$DP = [(0, 0, 0), (10, 8, 0)] - [(0, 0, 0), (6, 4, 0)]$

$= \{[(0, 0, 0), (0, 10, 0)] \Rightarrow$ Eliminado, pois $x_1 = x_2$ (Teste 2)

$= [(6, 0, 0), (10, 8, 0)]$

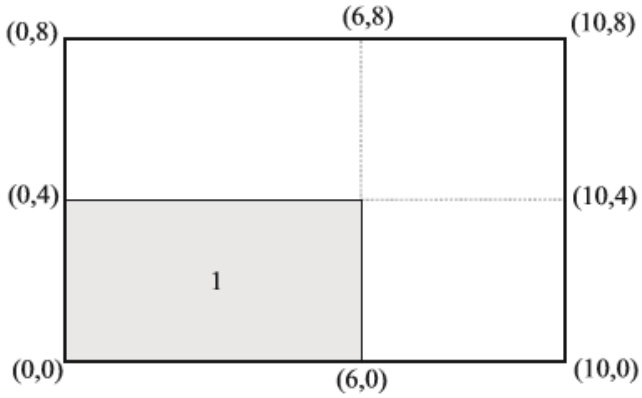
$= [(0, 0, 0), (10, 0, 0)] \Rightarrow$ Eliminado, pois $y_1 = y_2$ (Teste 2)

$= [(0, 4, 0), (10, 8, 0)]$

$= [(0, 0, 0), (10, 8, 0)] \Rightarrow$ EMS desconsiderado para o caso bidimensional

$= [(0, 0, 0), (10, 8, 0)]\} \Rightarrow$ EMS desconsiderado para o caso bidimensional

Figura 4.3 - Iteração 1 - primeiro carregamento da caixa 1 na posição original.



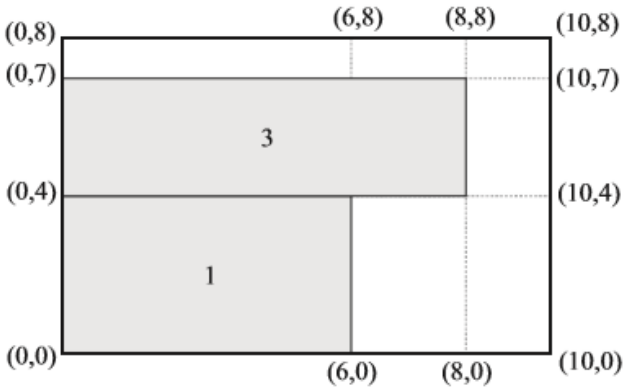
Fonte: Elaboração do próprio autor.

Ordenam-se os EMS remanescentes dos Testes de Eliminação, de acordo com a descrição do subitem 3.2.6, e atualiza-se S_1 = {[(0,4,0),(10,8,0)],[(6,0,0),(10,8,0)]}, contendo dois EMS, que são os maiores retângulos vazios visíveis na Figura 4.3: do ponto (0, 4) a (10, 8) e de (6, 0) a (10,8).

Iteração 2:

Verifica-se a possibilidade de carregamento da caixa 3 (próxima caixa a ser carregada) no primeiro EMS da lista S_1, com a orientação original. Não é possível. Rotacionando-se a caixa (invertendo dimensões da caixa), é possível realizar o carregamento da caixa 3 no EMS [(0, 4, 0), (10, 8, 0)] (Figura 4.4).

Figura 4.4 - Iteração 2 - segundo carregamento da caixa 3 rotacionada.



Fonte: Elaboração do próprio autor.

Deve-se realizar o processo DP tanto no EMS selecionado para o carregamento, [(0,4,0), (10,8,0)], quanto no EMS [(6,0,0),(10,8,0)], pois a caixa 3 o sobrepõe:

DP_1 = [(0, 4, 0), (10, 8, 0)] − [(0, 4, 0), (8, 7, 0)]
= {[(0, 4, 0), (0, 8, 0)] ⇒ Eliminado, pois $x_1 = x_2$
= [(8, 4, 0), (10, 8, 0)] ⇒ Eliminado, pela comparação com (*) (Teste 1)
= [(0, 4, 0), (10, 4, 0)] ⇒ Eliminado, pois $y_1 = y_2$
= [(0, 7, 0), (10, 8, 0)]}

DP_2 = [(6, 0, 0), (10, 8, 0)] − [(6, 4, 0), (8, 7, 0)]
= {[(6, 0, 0), (6, 8, 0)] ⇒ Eliminado, pois $x_1 = x_2$
= [(8, 0, 0), (10, 8, 0)] (∗)
= [(6, 0, 0), (10, 4, 0)]
= [(0, 7, 0), (10, 8, 0)]}

Ao final, atualiza-se S_1 de forma ordenada e contendo os três EMS restantes:

S_1 = {[(0, 7, 0), (10, 8, 0)], [(6, 0, 0), (10, 4, 0)], [(8, 0, 0), (10, 8, 0)]}

Iteração 3:

Verifica-se se há a possibilidade de carregamento da caixa 10 (próxima caixa a ser carregada) no primeiro EMS da lista S_1, [(0,7,0),(10,8,0)], com orientação original. Não é possível. Rotacionando-se a caixa, ainda assim não é possível efetuar o carregamento, o que também acontece para os outros dois EMS da lista S_1. Então, é necessário que um novo contêiner seja aberto, inicializado com S_2 = [(0,0,0),(10,8,0)], lista contendo um único EMS, que é o próprio contêiner 2, e no qual deve-se realizar o carregamento da caixa 10. (Figura 4.5).

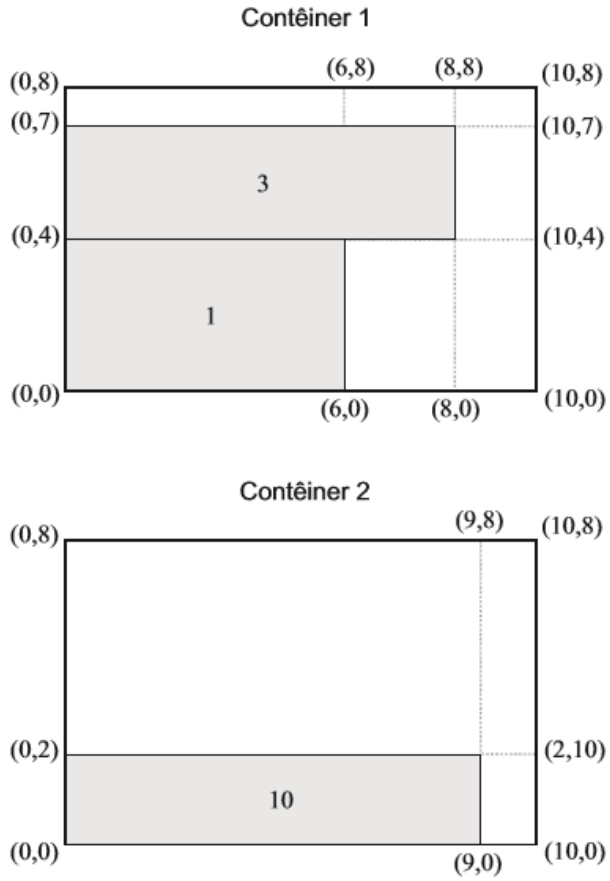
Deve-se realizar o processo DP apenas no EMS de carregamento [(0, 0, 0), (10, 8, 0)] do segundo contêiner e não atualizar S_1, uma vez que o contêiner 1 não foi modificado com o carregamento da terceira caixa. Logo:

DP = [(0, 0, 0), (10, 8, 0)] − [(0, 0, 0), (9, 2, 0)]
= {[(0, 0, 0), (0, 8, 0)] ⇒ Eliminado, pois $x_1 = x_2$
= [(9, 0, 0), (10, 8, 0)]

= [(0, 0, 0), (10, 0, 0)] ⇒ Eliminado, pois $y_1 = y_2$

= [(0, 2, 0), (10, 8, 0)]}

Figura 4.5 - Iteração 3 - terceiro carregamento da caixa 10 na posição original.



Fonte: Elaboração do próprio autor.

Finalizando a iteração 3, atualiza-se S_2 e copia-se S_1:

$$S_1 = \{[(0,7,0),(10,8,0)],[(6,0,0),(10,4,0)],[(8,0,0),(10,8,0)]\} \quad \text{e}$$

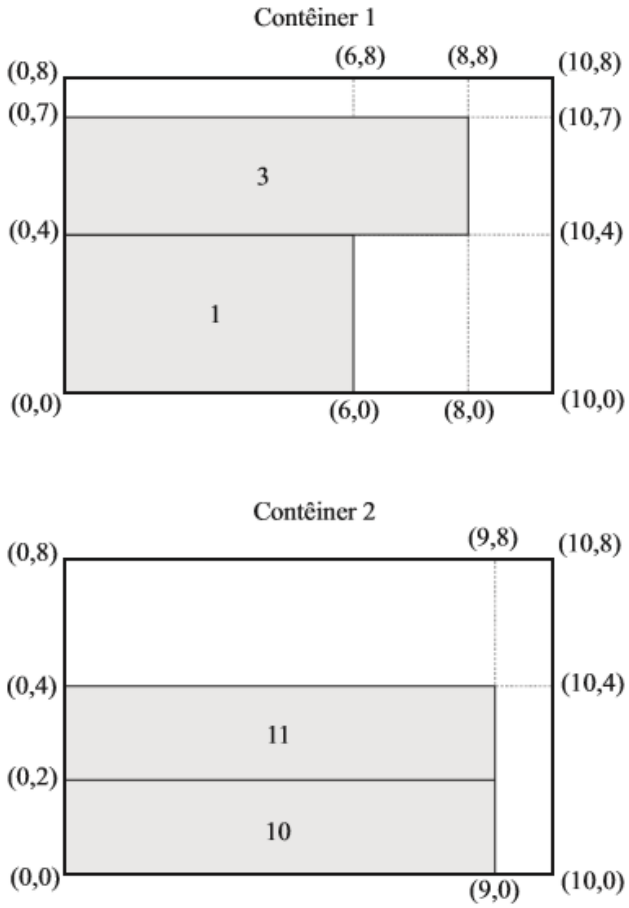
$$S_2 = \{[(0, 2, 0), (10, 8, 0)], [(9, 0, 0), (10, 8, 0)]\}$$

Iteração 4:

Verifica-se se há a possibilidade de carregamento da caixa 11 nos EMS da lista S_1. Entretanto, como a caixa 11 possui dimensões idênticas às da caixa 10, já é sabido que o carregamento não poderá acontecer no primeiro contêiner. Já no segundo contêiner, é possível realizar o carregamento da caixa 11 no primeiro EMS de S_2, [(0,2,0),(10,8,0)], com a primeira

orientação (Figura 4.6).

Figura 4.6 - Iteração 4 - quarto carregamento da caixa 11 na posição original.



Fonte: Elaboração do próprio autor.

Deve-se realizar o *Processo DP* apenas no EMS de carregamento $[(0,2,0),(10,8,0)]$ do segundo contêiner, já que o segundo EMS de S_2 não é sobreposto. Ou seja, o EMS remanescente de S_2, [(9,0,0),(10,8,0)] (∗∗), será apenas copiado para S_2 da próxima iteração. Novamente, S_1 não será atualizado. Logo:

$$DP = [(0, 2, 0), (10, 8, 0)] - [(0, 2, 0), (9, 4, 0)]$$
$$= \{[(0, 2, 0), (0, 8, 0)] \Rightarrow \text{Eliminado, pois } x_1 = x_2$$
$$= [(9, 2, 0), (10, 8, 0)] \Rightarrow \text{Eliminado, pela comparação com (**)}$$
$$= [(0, 2, 0), (10, 2, 0)] \Rightarrow \text{Eliminado, pois } y_1 = y_2$$
$$= [(0, 4, 0), (10, 8, 0)]\}$$

Dessa forma, tem-se:

$S_1 = \{[(0,7,0),(10,8,0)],[(6,0,0),(10,4,0)],[(8,0,0),(10,8,0)]\}$ e

$S_2 = \{[(0, 4, 0), (10, 8, 0)], [(9, 0, 0), (10, 8, 0)]\}$

Iteração 5:

Inicialmente, verifica-se a possibilidade de carregamento da caixa 2 no contêiner 1, o que é possível no segundo EMS de S_1, [(6,0,0),(10,4,0)], necessitando realizar o *Processo DP* deste EMS escolhido com a quarta caixa (caixa 2) carregada. Além disso, ao realizar tal carregamento, a caixa 2 sobrepõe o terceiro EMS de S_1, [(8,0,0),(10,8,0)], e, portanto, deve-se realizar um segundo *Processo DP* para lidar com as projeções geradas. Na Figura 4.7 ilustra-se este carregamento. O EMS [(0,7,0),(10,8,0)], em que a caixa 2 não o sobrepõe, deve ser apenas mantido na lista S_1.

Assim:

$DP_1 = [(6, 0, 0), (10, 4, 0)] - [(6, 0, 0), (10, 4, 0)]$

$= \{[(6, 0, 0), (6, 4, 0)] \Rightarrow$ Eliminado, pois $x_1 = x_2$

$= [(10, 0, 0), (10, 4, 0)] \Rightarrow$ Eliminado, pois $x_1 = x_2$

$= [(6, 0, 0), (10, 0, 0)] \Rightarrow$ Eliminado, pois $y_1 = y_2$

$= [(6, 4, 0), (10, 4, 0)] \Rightarrow$ Eliminado, pois $y_1 = y_2\}$

$DP_2 = [(8, 0, 0), (10, 8, 0)] - [(8, 0, 0), (10, 4, 0)]$

$= \{[(8, 0, 0), (8, 8, 0)] \Rightarrow$ Eliminado, pois $x_1 = x_2$

$= [(10, 0, 0), (10, 8, 0)] \Rightarrow$ Eliminado, pois $x_1 = x_2$

$= [(8, 0, 0), (10, 0, 0)] \Rightarrow$ Eliminado, pois $y_1 = y_2$

$= [(8, 4, 0), (10, 8, 0)]\}$

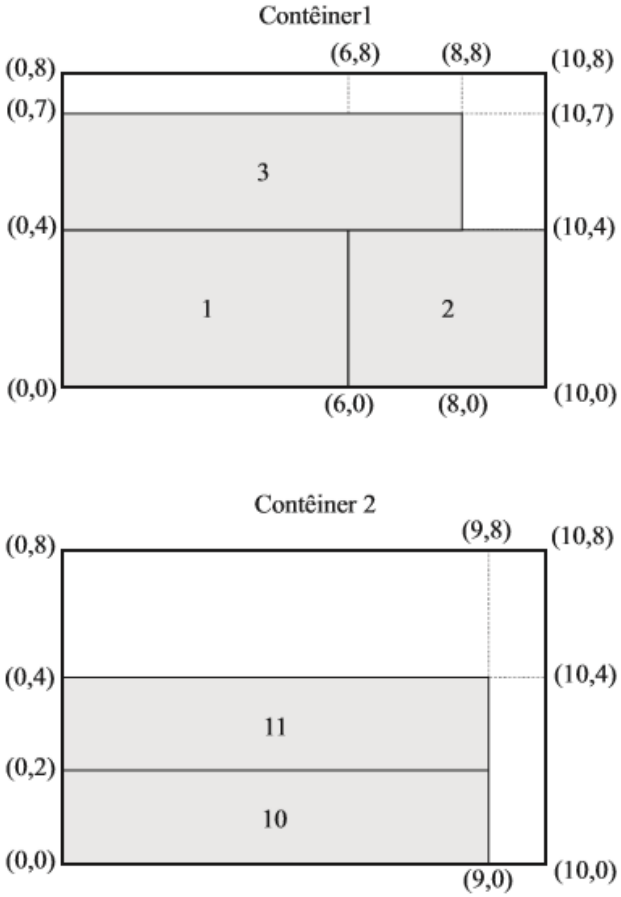
E as listas de EMS podem então ser atualizadas:

$S_1 = \{[(0,7,0),(10,8,0)],[(8,4,0),(10,8,0)]\}$ e

$S_2 = \{[(0, 4, 0), (10, 8, 0)], [(9, 0, 0), (10, 8, 0)]\}$

Este procedimento de carregamento de caixas continua sucessivamente, dentro do Decodificador, até que todas as caixas sejam carregadas.

Figura 4.7 - Iteração 5 - quinto carregamento da caixa 2 na posição original.



Fonte: Elaboração do próprio autor.

Observações Importantes

É importante ressaltar algumas considerações importantes com relação à *Estratégia de Carregamento*, que resumem o funcionamento do Decodificador:

1. O *Processo DP* deve ser realizado para a geração de novos EMS naquele EMS em que houve o carregamento da caixa atual. Neste caso, novos EMS irão substitui-lo na lista de EMS daquele contêiner. É o caso da parte 1 da Figura 3.13, em que houve o carregamento de uma caixa.

2. Se a caixa carregada não sobrepôs algum EMS da lista de EMS do contêiner, então este deve ser mantido na lista, sem alterações.

3. Se a caixa carregada sobrepôs outro EMS da lista de EMS, além daquele escolhido para

que a caixa seja carregada, então também deve ser aplicado o *Processo DP*. É o caso da parte 2 da Figura 3.13: A caixa [B, K] foi carregada no EMS [B, I]. Logo, em [B, I] deve ser realizado o *Processo DP*. Em seguida, o único EMS remanescente (excluindo aquele que já houve o carregamento) que havia na lista de EMS daquele contêiner era o EMS [D, I]. Se este não tivesse sido sobreposto, ele deveria ser mantido na lista de EMS do contêiner, assim como descrito no item 2. Entretanto, em [D, I] houve a sobreposição pela caixa [B, K], havendo então a necessidade de substituir esse EMS por outros, obtidos por um segundo *Processo DP*.

4. Após a realização de processos *DP* e cópia de EMS não sobrepostos, todos os EMS devem ser colocados em uma mesma lista de EMS, para realização dos Testes de Eliminação.

4.3.3 Métodos utilizados na implementação computacional

A implementação computacional do BRKGA foi realizada utilizando-se a linguagem C por meio da ferramenta de desenvolimento Open Source DevC++, compilador GCC 4.8.1. Os resultados computacionais foram obtidos utilizando um computador com CPU Intel Core i3-M380 @2.53 GHz, com o Sistema Operacional Microsoft Windows 7 Professional (32 bits). Os métodos utilizados estão explicitados a seguir:

1. GerarPopulacaoInicial(): Gera (uma única vez por tentativa de melhor solução) os valores iniciais entre 0 e 1 e salva na matriz *PopulacaoInicial*.

2. AlimentarValores(): Define os parâmetros de inicialização do algoritmo, tais como as dimensões das caixas, o número *n* de contêiners e a *fitness* máxima inicial para cada cromossomo. Além disso, cria a estrutura inicial de cada contêiner, contendo: uma lista de EMS (nomeada simplesmente por *EMS*) de até *LimiteEMS* =1000, uma variável booleana para indicar se o contêiner já foi aberto e uma variável que atualiza a área/volume utilizada(o) dentro do contêiner. O método também inicializa as matrizes auxiliares que serão utilizadas no decorrer do algoritmo: *EmsReaproveitados* (EMS que não foram sobrepostos e devem ser copiados), *EmsSobrepostos*, *EmsUtilizados* (EMS que sobram depois das eliminações por testes). Todo contêiner inicializa contendo um EMS útil (indicado sua disponibilidade através de uma variável booleana) de tamanho máximo, que é o próprio contêiner. E cada cromossomo possuirá *n* contêiners disponíveis para efetuar o carregamento, sendo o primeiro contêiner inicializado com a variável booleana igual a um, indicando, portanto, que está aberto.

3. Bps(): Cria uma matriz nomeada por *PopulacaoBps*, de dimensão *cromossomo*×*n*, que é estruturada por um registro de dados (*struct*), onde cada registro refere-se a uma caixa, e que contém as variáveis *ValorBps* e *IndiceCaixa*. O *ValorBps* recebe os primeiros *n* componentes extraídos da *PopulacaoInicial*, e o *IndiceCaixa* indica a qual caixa que o valor Bps está vinculado (para que a informação não seja perdida após a reordenação crescente). Além disso, é o momento onde é criada a matriz *Carregamento*, de dimensão *cromossomo* × *n*, onde cada posição da matriz contém um registro de dados (nomeado *CaixaCarregamento*) composto por 10 campos: o primeiro é o índice da caixa de acordo com o *Bps* (abaixo). Em seguida, existem as seis coordenadas de carregamento (x_1,y_1,z_1,x_2,y_2 e z_2), uma variável, *Orientacao*, que recebe a orientação a partir da posição original de cada caixa, respectivamente, uma variável booleana (*Carregada*) que informa se a caixa já foi carregada e a variável *Conteiner*, que recebe a informação sobre qual contêiner a caixa deverá ser carregada. Neste momento, apenas o índice será salvo na matriz *Carregamento*, definindo, então, a sequência de caixas a serem carregadas.
4. Vbo(): Cria uma matriz nomeada por *PopulacaoVbo*, de dimensão *cromossomo*×*n*, que irá receber os *n* últimos valores aleatórios das coordenadas da matriz *PopulacaoInicial*, *cromossomo* − *n*+ 1 até *cromossomo*, para cada linha.
5. CarregarCaixa(): Verifica qual o primeiro contêiner aberto, para cada cromossomo. Em seguida, verifica qual o primeiro EMS útil (disponível) de cada contêiner aberto (já considerando que a lista de EMS está testada e ordenada) em que há a possibilidade de se carregar a caixa atual *i*, em uma das orientações possíveis, descrita na matriz *Carregamento*, que é então atualizada a cada iteração. Esse método será repetido para todas as caixas e para todos os cromossomos.
6. VerificarSobreposicao(): Verifica se a caixa carregada sobrepõe EMS úteis dentro do contêiner utilizado. Se verdadeiro, o EMS sobreposto será subtraído da parcela da caixa sobreposta pelo *Processo DP* dentro do método *CalcularEMSsobrepostos* (abaixo). Caso contrário, o EMS não sobreposto, será armazenado na matriz auxiliar *EMSReaproveitados*.
7. CalcularEMSsobrepostos(): Calcula o processo DP para os EMS que foram sobrepostos pela caixa atual carregada, não incluindo o próprio EMS selecionado para realizar o carregamento. Os EMS gerados são armazenados na matriz *EMSsobrepostos*.
8. TestarEMSsobrepostos(): Verifica os quatro testes de eliminação de EMS, em todas as listas de EMS de cada contêiner que foram sobrepostos pela caixa atual. É um teste

prévio, visando à diminuição da lista final de EMS do método *calcularEMS*.

9. CalcularEMS(): Calcula o *Processo DP* apenas no EMS em que a caixa *i* foi carregada, utilizando as ideias de Lai e Chan [20]. São gerados seis EMS que são armazenados na lista de EMS do contêiner utilizado. Nesta mesma lista, também são incluídos os EMS contidos nas matrizes *EMSsobrepostos* e *EMSreaproveitados*.
10. TestarEMS(): Verifica os quatro testes de eliminação de EMS, na lista de EMS do contêiner utilizado.
11. OrdenarEMS(): Ordena a lista de EMS do contêiner utilizado, segundo as heurísticas BBL e BLB do vetor VPH.
12. CalcularFitness(): Calcula área/volume não utilizada(o) de todos os contêiners abertos e calcula o valor da *fitness*.
13. GerarPopulacaoElite(): Ordena a população inicial de acordo com os valores *fitness* obtidos pelo método *CalcularFitness*. Em seguida, são extraídos os p_e melhores valores (em que $p_e = |0{,}2 \times cromossomo|$) e copiados para a matriz *PopulacaoElite*.
14. GerarPopulacaoRecombinada(): Cria uma matriz auxiliar, de dimensão $(p_e + 1) \times (cromossomo - p_m)$, com $p_m = \rho_e$, que recebe valores aleatórios entre 0 e 1. Escolhe aleatoriamente um indivíduo da população elite e um indivíduo da população não-elite, faz a Recombinação PUC e armazena os novos indivíduos gerados em uma matriz chamada *PopulacaoRecombinada*.
15. GerarPopulacaoMutante(): Gera uma matriz *PopulacaoMutante* de $\rho_m \times componente$, em que componente=3n (três vezes o número de caixas), de forma aleatória, entre 0 e 1.
16. GerarNovaPopulacaoInicial(): Atualiza a matriz *PopulacaoInicial* com os valores obtidos pelas matrizes *PopulacaoElite*, *PopulacaoRecombinada* e *PopulacaoMutante*.
17. ListarMelhorCarregamento(): Apresenta a melhor solução encontrada em *k* execuções do algoritmo (consecutivas e distintas) realizadas.
18. Main(): É o método principal do programa que descreve toda a lógica do BRKGA. Unindo todas as informações, GerarPopulacaoInicial(), AlimentarValores(), Bps(), Vbo(), CarregarCaixa() CalcularFitness() e ListaMelhorCarregamento(), apresenta a melhor solução encontrada.

Observação: É importante destacar novamente que o segundo conjunto de *n* componentes, dentro de cada cromossomo, possui relevância apenas para os problemas tridimensionais. Entretanto, a mesma estrutura para os cromossomos foi mantida na implementação, contendo sempre 3*n* chaves

aleatórias, para que o mesmo algoritmo considerasse ambos os casos, bi e tridimensional.

4.3.4 Implementação computacional realizada para o BRKGA ± sem incluir a descrição do decodificador

- *Passo 0*: O método *GerarPopulacaoInicial* inicia com os parâmetros iniciais: n, p, *cromossomo* e ρ_e. (Para o Exemplo-Teste, $n = 12$, $p = cromossomo = 4 \times n = 48$, e $\rho_e = 0,7$). Uma população inicial de vetores com coordenadas no intervalo $[0,1]$ é gerada e armazenada na matriz *PopulacaoInicial* de dimensão $(cromossomo \times componente) + 1$, em que $componente = 3 \times n$ (que é igual a $3 \times 12 = 36$ para o Exemplo-Teste). A coordenada extra (no exemplo, a coordenada 37) existe para o armazenamento dos valores *fitness* de cada cromossomo, que inicializa com o valor TotaldeConteiner + AreaNaoUtilizada/AreaDoConteiner. No caso do exemplo para testes, este valor é igual a $12 + 80/80 = 13$; e, no caso tridimensional, entende-se a palavra "Área" como o cálculo dos volumes associados. Em seguida, o método *AlimentarValores* cria a estrutura inicial do carregamento e as matrizes auxiliares. Após, o Decodificador é utilizado para decodificar cada indivíduo contido em cada uma das linhas da matriz *PopulacaoInicial* como uma solução factível do problema original. Ao final, obtém-se os valores das *fitness* de cada cromossomo. Verifica-se se o critério de parada foi satisfeito, que, no caso, foi considerado quando uma melhor solução foi obtida, repetidamente, por um número k de iterações. Caso não seja satisfeito, inicia-se o processo geracional contido nos Passos 1 a 5.

- *Passo 1 (Reordenação)*: O método *GerarPopulacaoElite* reordena as linhas (cromossomos) da matriz *PopulacaoInicial* de acordo com uma ordenação crescente da coordenada $(cromossomo \times componente) + 1$ da matriz (37^a coordenada, no exemplo), que refere-se aos valores *fitness*. Então, as primeiras p_e linhas são armazenadas na matriz *PopulacaoElite*.

- *Passo 2 (Recombinação)*: O método *GerarPopulacaoRecombinada* gera o número $(cromossomo - p_e - p_m)$ de novas soluções por Recombinação através do mesmo valor $(cromossomo - p_e - p_m)$ de pares de pais geradores. O gerador elite de cada solução gerada terá um índice, que é um número inteiro aleatório no intervalo $[1, p_e]$. O outro gerador não-elite, terá também índice inteiro e aleatório, no intervalo $[p_e + 1, cromossomo]$. O i-ésimo descendente gerado é armazenado na i-ésima linha da matriz *PopulacaoRecombinada*, de dimensão $(cromossomo - p_e - p_m) \times componente$. Para isso, geram-se números aleatórios $crossover_{ij}$, no intervalo $i = 1,...,(cromossomo - p_e - p_m)$ e $j = 1, ...,componente$, em que é realizada a seguinte comparação:

$$crossover_{ij} \leq \rho_e.$$

Caso verdadeiro, copia-se para a coordenada *ij* do descendente, a coordenada *ij* do pai elite. Caso contrário, copia-se a coordenada *ij* da outra solução geradora.

- *Passo 3 (Mutação)*: O método *GerarPopulacaoMutante* gera p_m novas soluções, chamadas soluções mutantes, a serem armazenadas na matriz *PopulacaoMutante*, de dimensão $p_m \times componente$.

- *Passo 4:* O método *GerarNovaPopulacaoInicial* atualiza a matriz *PopulacaoInicial*, substituindo, respectivamente, as linhas pelos valores gerados nas matrizes *PopulacaoElite*, *PopulacaoRecombinada* e *PopulacaoMutante*.

- *Passo 5:* Utiliza-se o Decodificador para decodificar cada indivíduo contido em cada uma das linhas da matriz *PopulacaoInicial* atualizada e para obter os valores das *fitness* de cada cromossomo. Verifica-se o critério de parada, e caso tenha sido satisfeito, o método *ListarMelhorCarregamento* apresenta a melhor solução encontrada. Em caso contrário, inicia-se novamente o processo geracional contido nos Passos 1 a 5.

Observação: Foi também implementado um critério de parada global, externo, determinado pelo método *Main*, no qual realiza-se um número fixo *t* de execuções do Algoritmo BRKGA. Isto quer dizer que, após a obtenção de repetidas *k* melhores soluções idênticas, o BRKGA finaliza, porém, inicializa novamente, de maneira completamente independente da execução anterior. E estas reinicializações acontecem de forma a totalizar *t* execuções do Algoritmo. Foi chamado de *Critério de Parada Interno* (CPI) e *Critério de Parada Externo* (CPE) ambos os critérios de parada. Ao final, o Algoritmo implementado apresenta como solução final a melhor dentre as *t* soluções obtidas.

4.3.5 A implementação computacional do decodificador

A implementação computacional do decodificador do BRKGA, *Procedimento de Carregamento*, foi realizada de acordo com o apresentado no pseudocódigo da Figura 4.8. Neste, o número total de contêiners inicializado é igual ao número de caixas, ou seja, no pior das hipóteses, uma única caixa é carregada em cada contêiner. O Decodificador inicializa a

transformação de cada cromossomo da matriz *Carregamento* em uma *fitness* para o problema original, aplicando os métodos *Bps*() e *Vbo*(). Em seguida, para cada cromossomo, em cada contêiner aberto, e para o primeiro EMS útil possível, é realizado o carregamento da caixa atual.

Figura 4.8 - Pseudocódigo do decodificador *Procedimento de Carregamento* do BRKGA Implementado.

```
// Inicialização
TotalContainer = n;     // n = número de caixas
Bps()   // A matriz carregamento recebe a ordem de
          carregamento das caixas.
Vbo()   // Fornece a orientação inicial das caixas, ie,
          se a cada caixa irá ser carregada, tentando
          inicialmente pela posição original ou pela
          posição rotacionada.

// Processo de Carregamento de Caixas de acordo
  com a matriz carregamento
Para i = 1 até cromossomo       // cromossomo = 4n
Início
    Para b = 1 até TotalContainer
    Início
        Se container[b] está aberto então
        Início
            Para e = 1 até limiteEMS
            Início
                Se EMS[e] do container [b] é últil então
                Início
                    CarregarCaixa();
                    VerificarSobreposição();
                    CalcularEMSSobrepostos();
                    TestarEMSSobrepostos();
                    CalcularEMS();
                    TestarEMS();
                    OrdenarEMS();
                Fim
            Fim
        Fim
    Fim
Fim

// Finalização
CalcularFitness()
```

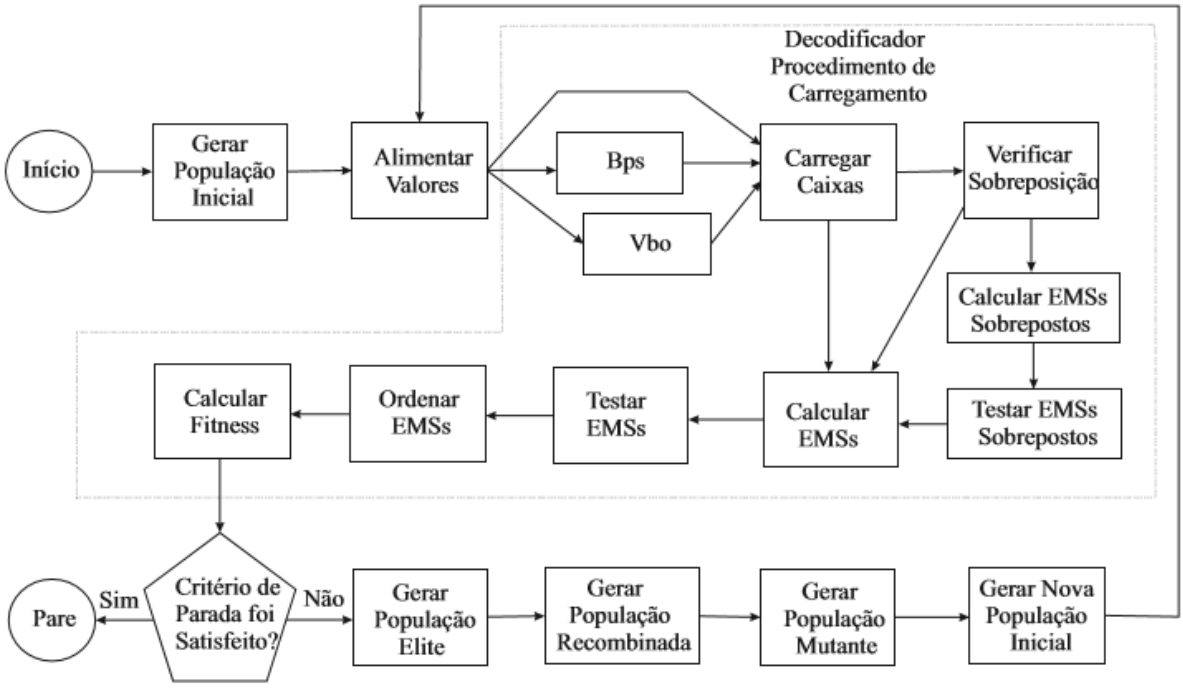
Fonte: Elaboração do próprio autor.

4.3.6 Visão global da implementação

Todos os passos do algoritmo BRKGA estão dispostos na Figura 4.9, incluindo a estrutura

geral do algoritmo, e um detalhamento das etapas de implementação do Decodificador.

Figura 4.9 - Estrutura Completa do Algoritmo BRKGA Implementado.



Fonte: Elaboração do próprio autor.

4.3.7 Resultados computacionais parciais

Primeiramente, como teste inicial, o algoritmo BRKGA implementado foi verificado para o Exemplo-Teste bidimensional, pois, além da solução ótima ser conhecida, objetivou-se validar os resultados através de valores simples e, no qual, a solução ótima poderia ser facilmente determinada. Esperava-se que o fato de ter-se mantido a mesma estrutura base do algoritmo para ambos os casos, bi e tridimensional, pudesse tornar mais fácil a implementação computacional do caso tridimensional.

Além da validação da implementação do BRKGA para o caso bidimensional piloto, os testes tiveram como objetivo adjacente a verificação do impacto das reinicializações externas (CPE) no desempenho do algoritmo. Para isso, foram realizados um total de 60 testes computacionais introdutórios, divididos em 4 grupos de 15 conjuntos de dados, e que investigaram estas questões acerca do CPE e da qualidade das soluções obtidas considerando também o CPI. Os resultados estão apresentados na Tabela 4.2.

Para os dois primeiros grupos, Grupo 1A e Grupo 1B, foi mantido o número de

inicializações igual a três, gerando-se uma nova população inicial de forma aleatória a cada reinicialização externa. O CPI estabelecido para o Grupo 1A foi considerado como a interrupção da execução após 3 repetições da melhor solução encontrada, e, para o Grupo 1B, um CPI sendo igual à repetição de 20 melhores soluções iguais. Para o Grupo 2A e 2B, estratégia análoga foi adotada, mantendo-se o CPE igual a 75 reinicializações, e variando-se o número de repetições internas, CPI, de 8 no Grupo 2A, para 20, no Grupo 2B.

Tabela 4.2 - Resultados Computacionais do Exemplo-Teste.

Código do Grupo do Problema	Número de execuções - Reinicializações (CPE)	Número de repetições da melhor solução encontrada (CPI)	Melhores soluções encontradas	Tempo total de execução (média)
Grupo 1A	3	8	3,40; 3,40; 3,40	0h24
			3,40; 3,40; 3,35	0h33
			3,35; 3,40; 3,35	0h32
			3,40; 3,40; 3,40	0h18
			3,40; 3,40; 3,40	0h33
Média			**3,39**	**0h28**
Grupo 1B	3	20	3,35; 3,40; 3,40	1h51
			3,40; 3,35; 3,40	1h46
			3,35; 3,40; 3,35	1h59
			3,35; 3,40; 3,40	1h55
			3,40; 3,40; 3,35	1h51
Média			**3,38**	**1h52**
Grupo 2A	75	8	3,35; 3,35; 3,40	28h39
			2,01; 2,01; 3,35	25h41
			2,01; 3,35; 3,35	19h17
			3,35; 2,01; 3,35	24h35
			2,01; 2,01; 3,35	25h39
Média			**2,82**	**24h46**
Grupo 2B	75	20	2,01; 2,01; 3,40	57h50
			2,01; 2,01; 2,01	45h59
			3,35; 2,01; 3,35	55h12
			2,01; 2,01; 3,35	56h17
			3,35; 2,01; 2,01	57h45
Média			**2,46**	**54h37**

Fonte: Elaboração do próprio autor.

A opção por tomar-se valores extremos, principalmente no que concerne ao CPE, teve como objetivo potencializar a percepção proveniente dos resultados obtidos que possui natureza empírica. Valores maiores que 20 para as repetições no CPI também foram inicialmente supostos, porém, a

exequibilidade foi comprometida, culminando, portanto, em uma menor variação para o CPI.

A quarta coluna da Tabela 4.2 apresenta as quinze melhores soluções encontradas para cada grupo, que foram agrupadas de três em três, em cada linha da tabela. Ressalta-se que a opção de se agrupar em trios não faz correlação com as três execuções sequenciais externas do CPE, do Grupo 1. Cada um destes quinze melhores valores, tiveram suas 3 reinicializações externas e suas 8 repetições de soluções internas, por exemplo, no que concerne ao Grupo 1A. Procedimento semelhante é aplicada aos demais grupos, enfatizando que a implementação retorna como valor final, independentemente do número CPE ou CPI, apenas um único valor final *fitness*, que é a melhor aproximação encontrada para a solução do problema real. A única intenção foi apresentar apenas uma contagem de 5 tempos, ao invés de 15 tempos, para detalhar com maior precisão a média final de tempo demandado, no lugar de apenas apresentar uma média de tempo por grupo.

Os 60 valores computacionais da tabela indicam que um maior número de reinicializações para o CPE influencia com maior impacto no melhoramento da qualidade das soluções obtidas do que o aumento do CPI. A solução ótima 2.0125 do *Problema-Teste* (Subseção 4.3.1) não foi obtida em nenhuma das 30 iterações do Grupo 1, com um número pequeno para o CPE, mesmo para um caso simples bidimensional.

Mesmo utilizando o critério de parada interno igual a uma repetição de apenas 8 aproximações iguais, com 75 reinicializações, o BRKGA-2DBPP conseguiu em 80% dos casos encontrar a solução ótima, já previamente conhecida (ver Subseção 4.3.1). Em contrapartida, aumentando-se para 20 as repetições internas e considerando apenas 3 reinicializações globais, o Algoritmo não produziu a solução ótima, em nenhuma das vezes testadas.

Além disso, pode-se constatar o fato de que as duas configurações de carregamento obtidas pelas ocorrências de solução ótima do grupo 2 são diferentes daquelas em que se almejava determinar, e explicitada na Seção 4.3.1, porém com mesma *fitness*, demonstrando a eficácia da implementação do Algoritmo BRKGA. São elas:

Configuração de Carregamento 1:

Contêiner 1:

Caixa 5: [(0, 0, 0), (3, 4, 0)] com orientação 2.

Caixa 11: [(0, 4, 0), (9, 6, 0)] com orientação 1.

Caixa 6: [(3, 0, 0), (6, 4, 0)] com orientação 2.

Caixa 2: [(6, 0, 0), (10, 4, 0)] com orientação 1.

Caixa 10: [(0, 6, 0), (9, 8, 0)] com orientação 1.

Caixa 12: [(9, 4, 0), (10, 7, 0)] com orientação 2.

Contêiner 2:

Caixa 4: [(0, 0, 0), (3, 4, 0)] com orientação 2.

Caixa 7: [(0, 4, 0), (1, 7, 0)] com orientação 1.

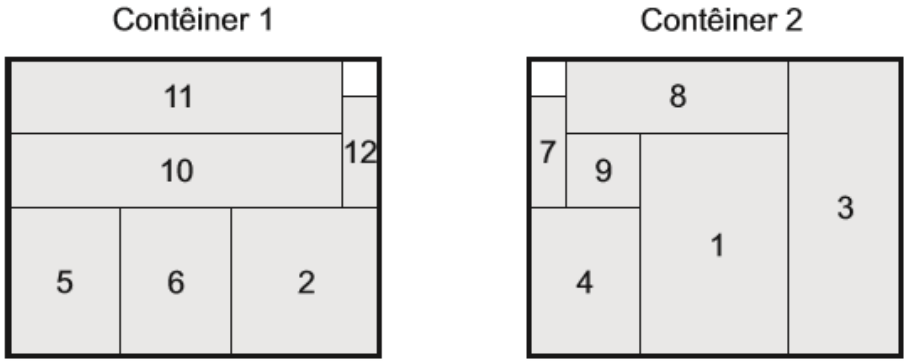
Caixa 9: [(1, 4, 0), (3, 6, 0)] com orientação 2.

Caixa 8: [(1, 6, 0), (7, 8, 0)] com orientação 1.

Caixa 3: [(7, 0, 0), (10, 8, 0)] com orientação 2.

Caixa 1: [(3, 0, 0), (7, 6, 0)] com orientação 1.

Figura 4.10 - Configuração de Carregamento 1.



Fonte: Elaboração do próprio autor.

Configuração de Carregamento 2:

Contêiner 1:

Caixa 2: [(0, 0, 0), (4, 4, 0)] com orientação 2.

Caixa 11: [(0, 4, 0), (9, 6, 0)] com orientação 1.

Caixa 4: [(4, 0, 0), (7, 4, 0)] com orientação 2.

Caixa 6: [(7, 0, 0), (10, 4, 0)] com orientação 2.

Caixa 10: [(0, 6, 0), (9, 8, 0)] com orientação 1.

Caixa 12: [(9, 4, 0), (10, 7, 0)] com orientação 2.

Contêiner 2:

Caixa 1: [(0, 0, 0), (4, 6, 0)] com orientação 1.

Caixa 5: [(4, 0, 0), (7, 4, 0)] com orientação 2.

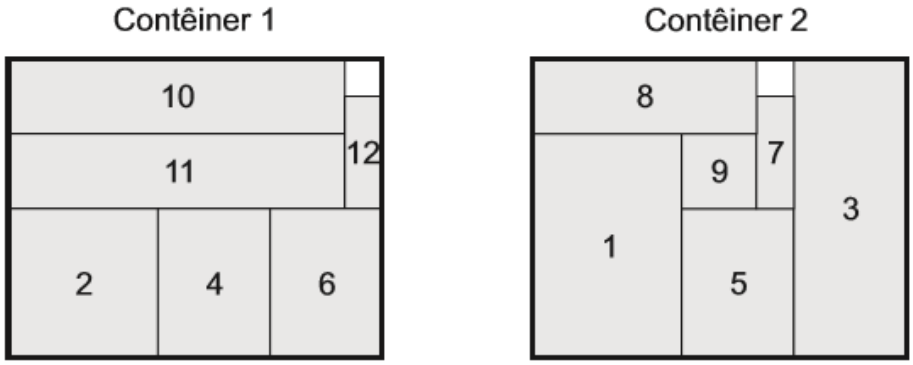
Caixa 3: [(7, 0, 0), (10, 8, 0)] com orientação 2.

Caixa 8: [(0, 6, 0), (6, 8, 0)] com orientação 1.

Caixa 9: [(4, 4, 0), (6, 6, 0)] com orientação 1.

Caixa 7: [(6, 4, 0), (7, 7, 0)] com orientação 1.

Figura 4.11 - Configuração de Carregamento 2.



Fonte: Elaboração do próprio autor.

4.4 IMPLEMENTAÇÃO COMPUTACIONAL DO BRKGA PARA O CASO TRIDIMENSIONAL

A estrutura da implementação do BRKGA para o caso tridimensional foi realizada, praticamente, de maneira idêntica à que foi desenvolvida para o caso bidimensional, com algumas exceções. Primeiramente, as terceiras componentes das caixas, contêiners, e pontos que descrevem os posicionamentos das caixas dentro dos contêiners, deixaram de ter valores apenas iguais a zero. Isso incidiu diretamente no fato de que, agora, o segundo conjunto de *n* componentes dos cromossomos, que descreve o vetor VPH, foi ativado, e que o vetor VBO começou a considerar as seis possibilidades de rotacionamento para cada caixa. Dentro do *Processo DP*, também, os últimos EMS gerados (5^o e 6^o EMS, dentre os 6 novos possíveis) passaram a fazer sentido e, portanto, a influenciar na determinação de espaços maximais vazios.

Para iniciar os testes computacionais, o algoritmo foi implementado para o que se chamou

Problema-Teste Tridimensional, cuja solução ótima era conhecida e trivial, e no qual objetivou-se, apenas, a validação da estrutura do algoritmo BRKGA, também para o caso tridimensional. A estrutura deste exemplo considera um único contêiner e contém poucas caixas, estas, fortemente heterogêneas.

Na sequência, foram realizados testes computacionais para os primeiros exemplos de grande porte, para problemas do tipo 3D-BPP, envolvendo 50 caixas e pelo menos 10 contêiners. A escolha destes problemas de grande porte é proveniente do fato de que, além de serem exemplos complexos para se resolver, são os problemas encontrados na bibliografia base deste trabalho, apresentados no Capítulo 1.4. Dessa forma, a ideia foi estabelecer-se um critério de comparação entre os resultados obtidos e verificar-se a coerência da implementação efetuada. A expectativa era de que os resultados estivessem próximos ou iguais aos determinados pelo trabalho de Resende e Gonçalves [12].

Esta subseção final cumpre com o terceiro objetivo específico determinado no início deste trabalho.

4.4.1 O problema-teste tridimensional

Os dados do exemplo inicial se encontram no trabalho de Dyckhoff [13], e o objetivo foi verificar a validade do algoritmo BRKGA, implementado para o caso tridimensional, e com as alterações supracitadas e descritas como *exceções*. Os dados do problema estão na Figura 4.12, que apresenta as dimensões de seis caixas que devem ser carregadas em um único contêiner, com utilização de 50% de seu volume total. Logo, todo carregamento será ótimo, e espera-se que a *fitness* sempre retorne o mesmo valor αNB.

Como $NB = 1$, $LeastLoad = 210$ (somatório do volume das seis caixas) e $BinCap = 6 \times 10 \times 7 = 420$, então a *fitness* esperada, para qualquer carregamento, deve ser igual a:

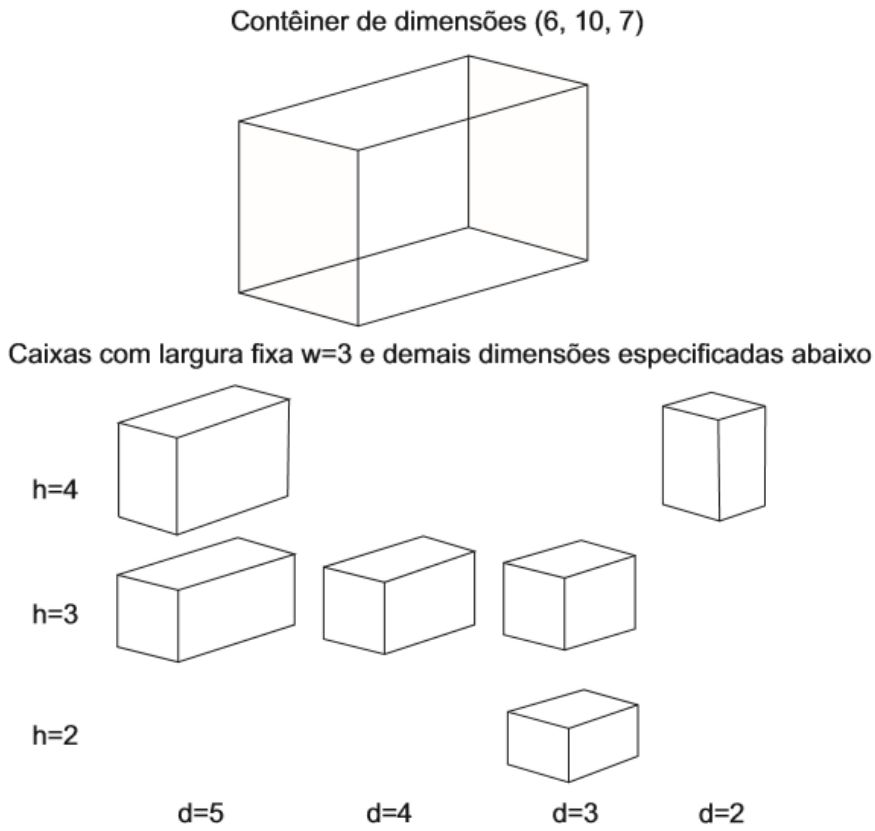
$$\alpha NB = 1 + \frac{210}{420} = 1{,}5$$

A Figura 4.13 ilustra uma das várias possibilidades de carregamento.

Foram realizadas duas execuções do Problema-Teste tridimensioal. Em ambas as vezes, considerou-se oito execuções externas e três execuções internas, CPE = 8 e CPI = 3, de acordo com as observações realizadas para o caso bidimensional. A solução ótima, em ambos os casos, foi encontrada de maneira correta, e devia ser a melhor solução dentre as oito soluções internas

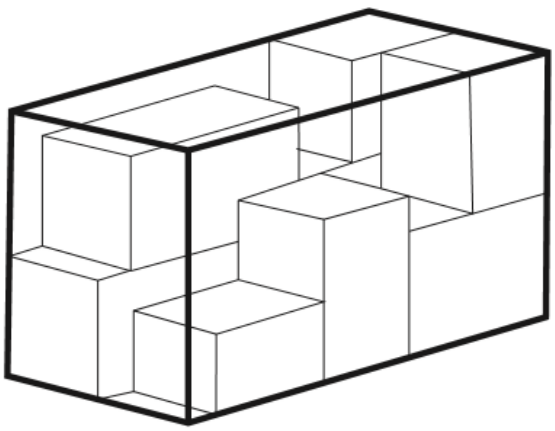
obtidas pelas reinicializações do algoritmo. Em todos os casos, obteve-se $\alpha N\ B = 1, 5$, em um tempo médio de 32 minutos e 40 segundos, mesmo considerando-se os resultados das 8 reinicializações dentro de cada uma das duas execuções.

Figura 4.12 - Dados do Exemplo-Teste tridimensional.



Fonte: Adaptado de Dyckhoff (1990) [13].

Figura 4.13 - Solução ótima do Exemplo-Teste.



Fonte: Adaptado de Dyckhoff (1990) [13].

Os resultados computacionais sobre as localizações das caixas foram retornados como a seguir. Na Figura 4.14 ilustram-se as duas soluções finais, que foram rotacionadas para melhor visualização das não sobreposições. Para auxiliar nesta visualização e ajustes no decorrer da implementação, foi utilizado o software Wolfram Mathematica, versão 9, para identificar possíveis problemas de sobreposições entre caixas.

Configuração de Carregamento 1:

Contêiner Único

Caixa 4: [(0, 0, 0), (3, 4, 3)] com orientação 4.

Caixa 1: [(0, 0, 3), (3, 5, 7)] com orientação 6.

Caixa 5: [(3, 0, 0), (6, 3, 3)] com orientação 1.

Caixa 6: [(0, 4, 0), (3, 6, 3)] com orientação 5.

Caixa 2: [(3, 3, 0), (7, 6, 2)] com orientação 3.

Caixa 3: [(3, 3, 2), (8, 6, 5)] com orientação 2.

Configuração de Carregamento 2:

Contêiner Único

Caixa 2: [(0, 0, 0), (3, 2, 4)] com orientação 6.

Caixa 3: [(0, 2, 0), (5, 5, 3)] com orientação 1.

Caixa 1: [(0, 0, 4), (4, 5, 7)] com orientação 4.

Caixa 6: [(3, 0, 0), (6, 2, 3)] com orientação 2.

Caixa 4: [(4, 0, 3), (7, 3, 7)] com orientação 3.

Caixa 5: [(4, 3, 3), (7, 6, 6)] com orientação 5.

4.4.2 Os exemplos de grande porte

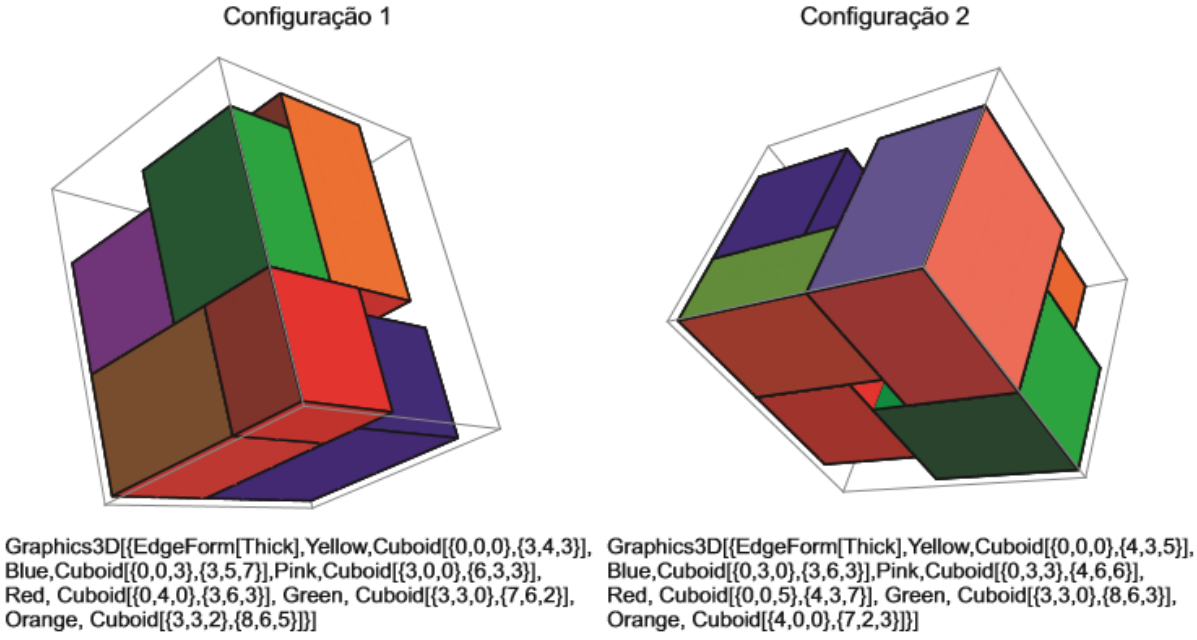
Os exemplos de grande porte usados nos testes computacionais foram extraídos do trabalho de Martello *et. al* [9], e são os mesmos utilizados por Resende e Gonçalves [12], uma das referências de estudo para a elaboração deste documento. Em ambos os estudos foram realizados testes em um conjunto amplo de problemas , porém, nesta Subseção, será considerada uma dentre as categorias

definidas pelo trabalho de Resende e Gonçalves [12], e que foi constituída a partir de um gerador de exemplos, proposto por Pisinger em [105]. Esta categoria foi nomeada *Classe 1* em [12], e envolve o tratamento de dez problemas do tipo 3D-BPP, logo, com caixas fortemente heterogêneas, e consideram um número de 50 caixas que devem ser carregadas em contêiners idênticos de dimensões $W = H = D = 100$. As caixas possuem dimensões w_j,h_j e d_j, que foram uniformemente geradas de acordo com cinco tipos de critérios:

Tipo 1	$w_j \in [1, \frac{1}{2}W]$,	$h_j \in [\frac{2}{3}H, H]$,	$d_j \in [\frac{2}{3}D, D]$
Tipo 2	$w_j \in [\frac{1}{2}W, \frac{2}{3}W]$,	$h_j \in [1, \frac{1}{2}H]$,	$d_j \in [\frac{2}{3}D, D]$
Tipo 3	$w_j \in [\frac{1}{2}W, \frac{2}{3}W]$,	$h_j \in [\frac{2}{3}H, H]$,	$d_j \in [1, \frac{1}{2}D]$
Tipo 4	$w_j \in [\frac{1}{2}W, W]$,	$h_j \in [\frac{1}{2}H, H]$,	$d_j \in [\frac{1}{2}D, D]$
Tipo 5	$w_j \in [1, \frac{1}{2}W]$,	$h_j \in [1, \frac{1}{2}H]$,	$d_j \in [1, \frac{1}{2}D]$

para todo $j = 1, ..., n$.

Figura 4.14 - Configurações de carregamento para o Exemplo-Teste 3D.



Fonte: Elaboração do próprio autor.

Em todos os 10 problemas da *Classe 1*, cada caixa possui 60% de probabilidade de ter suas dimensões geradas a partir da categoria Tipo 1, e 10% para os outros quatro tipos de critérios. No Apêndice A apresentam-se as dimensões de todas as caixas, para cada problema, nas Tabelas A.1

até A.5.

De acordo com Resende e Gonçalves [12], e assim como em toda meta-heurística, a definição dos parâmetros de entrada constitui-se como um problema a parte que depende muito mais da arte empírica de manipulação e verificação dos dados, e que requer grande experiência com este tipo de situação. Os autores supracitados possuem este passado ao estudar e testar as estratégias do BRKGA em vários trabalhos, tais como: Gonçalves e Almeida [91], Ericsson et al. [92], Resende e Gonçalves [93], Gonçalves et al. [94], Buriol et al. [90], Buriol et al. [95], Gonçalves [96], Gonçalves et al. [97], Fontes e Gonçalves [98], Festa et al. [99], Resende e Gonçalves [100], Resende e Gonçalves [101], Gonçalves e Souza [102], Resende e Gonçalves [103] e Resende e Gonçalves [17]. Após tantos anos de estudo, os autores puderam obter bons resultados aos considerar parâmetros iniciais contidos nos intervalos da Tabela 4.1. Para os exemplos de grande porte do trabalho [12], Resende e Gonçalves testaram todas as combinações entre os valores:

- p = 10, 20,30 e 40 vezes o número de itens do problema testado;
- $p_e \in \{0{,}10; 0{,}15; 0{,}20; 0{,}25\} \times p$
- $p_m \in \{0{,}15; 0{,}20; 0{,}25; 0{,}30\} \times p$
- $\rho_e \in \{0{,}70; 0{,}75; 0{,}80\}$

Os melhores valores foram encontrados utilizando-se os parâmetros dispostos na Tabela 4.1, na coluna *Valores Recomendados*. O critério de parada foi estabelecido em [12] como sendo o de 300 gerações, e os resultados computacionais obtidos foram comparados com outros quatro algoritmos (dispostos na Tabela 4.3), os melhores da bibliografia para o 3D-BPP, e que também realizaram testes com os mesmos problemas de grande porte. Resende e Gonçalves nomearam estes algoritmos como *Algoritmos de Teste de Desempenho* (do inglês, *Benchmark Algorithms*), cujas especificações são apresentadas na Tabela 4.3.

Tabela 4.3 - Algoritmos Eficientes para Teste de Desempenho.

Algoritmo	Trabalho de origem	Tipo de método
TS3	Lodi et al. [73] e Lodi et al. [74]	Busca Tabu
GLS	Faroe et al. [72]	Busca Local Guiada
TS^{2}PACK	Crainic et al. [76]	Busca Tabu Paralela
GVND	Parreño et al. [81]	GRASP/VND

Fonte: Adaptado de Resende e Gonçalves (2013) [12].

Entretanto, esses outros quatro algoritmos consideram apenas que o carregamento das caixas deve ser realizado em uma única orientação, a orientação inicial. Resende e Gonçalves [12], implementaram, portanto, o BRKGA, fixando-se apenas uma orientação, para fins de comparação entre os seus resultados e aqueles contidos na literatura, porém, também realizaram testes envolvendo as seis rotações para as caixas, esperando encontrar melhores resultados, e na tentativa de demonstrar o potencial do BRKGA.

O BRKGA implementado neste trabalho utilizou como parâmetros iniciais aqueles dispostos na coluna *Valores Utilizados* da Tabela 4.4, o qual demonstra que os parâmetros utilizados por Resende e Gonçalves [12] (Valores Recomendados) foram mantidos na implementação realizada, com exceção do número de cromossomos da população (p) e do critério de parada. A decisão sobre essas mudanças levou em consideração o fato de que o tempo computacional mostrava-se muito elevado em relação aos ganhos obtidos.

Tabela 4.4 - Valores recomendados para os parâmetros do BRKGA.

Parâmetros	Valores Recomendados	Valores Utilizados
p	$30 \times n$	$10 \times n$
p_e	$0,10 \times p$	$0,10 \times p$
p_m	$0,15 \times p$	$0,15 \times p$
ρ_e	$0,70$	$0,70$
Critério de Parada	300 gerações	CPI $=$ 3 e CPE $=$ 3

Fonte: Adaptado de Resende e Gonçalves (2013) [12].

Sobre o critério de parada, também se considerou os resultados dos testes computacionais realizados com o *Problema-Teste Bidimensional*, analisados a partir da Tabela 4.2, e descritos no subcapítulo 4.3.7. O número CPI = 3 foi escolhido, pois, nas experimentações, houve a percepção de que, uma vez que as soluções começavam a se repetir duas ou três vezes, ela continuaria a se repetir, independentemente do número escolhido, logo, é o menor número que tem significância. Esse comportamento era esperado, por caracterizar-se como o principal problema dos algoritmos genéticos: a perda da diversidade ocasionada pelas melhores soluções dominantes. O valor CPE=3 foi fixado, também após experimentações, por ser o maior valor possível que retornasse boas soluções em tempo exequível.

4.4.3 Resultados computacionais parciais

Como descrito anteriormente, a implementação computacional do BRKGA foi realizada utilizando-se a linguagem C por meio da ferramenta de desenvolvimento Open Source DevC++, compilador GCC 4.8.1 e os resultados computacionais foram obtidos utilizando um computador com CPU Intel Core i3-M380 @2.53 GHz, com o Sistema Operacional Microsoft Windows 7 Professional (32 bits).

Para a possibilidade de rotação das caixas em qualquer umas das 6 disposições, o BRKGA implementado considera a orientação inicial como sendo a orientação 1, e, para cada caixa, através de um número k gerado aleatoriamente do conjunto $\{1, 2, 3, 4, 5, 6\}$, inicia a tentativa de carregamento do item a partir deste número k, até o número 6, retornando a sequência ao número 1, que finaliza, no máximo, com o valor $k-1$. Por exemplo, se uma caixa é selecionada para ser carregada em um EMS, com orientação 4, então o algoritmo primeiramente tenta carregá-la no primeiro EMS da lista de EMS do primeiro contêiner, com esta orientação 4. Caso seja infactível, o algoritmo realiza a tentativa com a orientação 5, e depois, caso ainda haja infactibilidade, com a orientação 6, retornando, depois, para tentativas com a orientação 1, orientação 2 e, finalmente, caso nenhuma das possibilidades anteriores de rotação da caixa tenha sido possível realizar o carregamento no EMS escolhido, tenta-se a orientação 3. Caso nenhuma das possibilidades testadas seja possível de haver a alocaçao da caixa, então tenta-se carregá-la, com raciocínio análogo, no segundo EMS possível da lista de EMS, caso ele exista, e assim, sucessivamente, até que todos os EMS da lista do primeiro contêiner sejam testados. Só depois, o algoritmo passa ao segundo contêiner aberto, caso exista, e realiza tentativas até que todos os contêiners já abertos tenham sido testados para carregar a caixa selecionada. Se ainda assim, nenhuma tentativa for satisfeita, então o algoritmo considera a utilização de um novo contêiner e carrega a caixa em sua posição inicial, origem do espaço tridimensional. Todo esse procedimento de realização de testes dentro do Decodificador *Procedimento de Carregamento* (descrito no pseudocódigo da Figura 4.8) demanda um tempo computacional muito elevado, e é a parte que solicitou quase que todo o tempo dispendido para a realização dos carregamentos, no decorrer das gerações.

Na Tabela 4.5 apresentam-se os resultados computacionais provenientes da execução do BRKGA implementado aplicado aos problemas de grande porte. As três execuções externas (reinicializações) de cada problema forneceu as três melhores soluções dispostas na quarta coluna, *Soluções*. O valor *fitness*, αNB, retornado ao fim de cada execução global é o melhor resultado encontrado entre estas três melhores soluções, apresentado na segunda coluna da tabela. As colunas

tempo(min) e GE expõem, respectivamente, o tempo em minutos e o número de gerações necessárias para o término de cada execução externa. A coluna *Gerações* exibe a quantidade de gerações necessárias para a obtenção da *fitness* e a coluna t(min)/G, o tempo em minutos necessários para a execução de cada geração. A última linha apresenta ainda as médias de cada um destes valores determinados.

Tabela 4.5 - Resultados Computacionais Parciais do BRKGA.

Problemas	αNB	Gerações	Soluções	Tempo (min)	GE	t (min)/G
			13.4010	427	11	39
PROB01-050-01	13.2870	23	13.5504	167	5	33
			13.2870	234	7	33
			13.1995	1351	17	79
PROB01-050-02	13.1995	46	13.2139	563	15	38
			13.3960	1365	14	97
			12.5981	356	10	36
PROB01-050-03	12.5981	20	12.6081	205	6	34
			13.1190	136	4	34
			10.3905	238	8	30
PROB01-050-04	10.2078	24	11.1677	237	6	40
			10.2078	296	10	30
			10.3102	210	7	30
PROB01-050-05	10.3102	15	10.4756	122	4	31
			10.3469	121	4	30
			11.4023	124	4	31
PROB01-050-06	10.3988	33	10.4406	481	16	30
			10.3988	392	13	30
			14.2975	285	8	36
PROB01-050-07	14.1771	25	14.1771	389	11	35
			14.4709	212	6	35
			15.4351	179	5	36
PROB01-050-08	15.4351	15	15.4855	180	5	36
			15.4561	179	5	36
			10.4044	186	6	31
PROB01-050-09	10.2530	26	10.2530	433	14	31
			10.3225	185	6	31
			9.3384	413	14	30
PROB01-050-10	9.2622	46	9.2622	706	24	29
			10.1253	117	4	29
Média	11.9129	27	12.0847	350	9	37

Fonte: Elaboração do próprio autor.

Na Tabela 4.6 apresenta-se também uma comparação entre as médias das *fitness* determinadas por sete algoritmos distintos, (1) o BRKGA implementado no presente trabalho; e os demais resultados apresentados no trabalho de Gonçaves e Resende [12]: (2) BRKGA-6r [12], implementado por Resende e Gonçalves, considerando-se as seis rotações; (3) BRKGA [12], implementado por Resende e Gonçalves, considerando-se apenas uma única orientação fixa por caixa; (4) TS3; (5) GVND; (6) TS^{2}PACK; e (7) GLS. Com exceção do primeiro algoritmo, as médias das demais meta-heurísticas foi retirada do trabalho [12].

Tabela 4.6 - Melhores aproximações para o valor αNB.

Meta-heurística	*Fitness* (αNB)
BRKGA	11.9
BRKGA-6r [12]	11.5
BRKGA [12]	13.4
TS3	13.4
GVND	13.4
TS^{2}PACK	13.4
GLS	13.4

Fonte: Elaboração do próprio autor.

Os resultados computacionais de Resende e Gonçalves demonstraram que, considerando-se todas as classes de problemas, o BRKGA [12] produz soluções de igual ou melhor qualidade do que as outras quatro meta-heurísticas testadas no artigo. Para o caso particular dos problemas considerados nesta Subseção (*Classe 1* de [12]), o BRKGA [12] apresentou uma média de *fitness* de igual qualidade comparando-a às obtidas pelas metaheurísticas (4) a (7), $\alpha NB = 13.4$. Como esperado, a meta-heurística (2), por considerar todas as seis rotações de itens, conseguiu produzir melhores resultados do que os algoritmos (3) a (7), obtendo $\alpha N\ B = 11.5$. Como é possível observar na segunda linha da Tabela 4.6, o algoritmo (1), BRKGA implementado no desenvolvimento do presente trabalho, teve eficiência muito próxima ao obtido em [12], ambos ao considerarem as seis rotações de caixa, com uma média de *fitness* igual a 11.9. Isso demonstrou a coerência dos resultados obtidos e a validação da implementação do BRKGA. Os resultados também mostraram que, no BRKGA implementado, cada geração leva, em média, 37 minutos para ser finalizada, e que, cada execução global de três execuções externas, com repetição de três soluções internas, demanda, em torno de 27 gerações. Logo, a média de tempo destes problemas de grande porte, necessitou de aproximadamente 16,5 horas para ser executado. Os problemas 10, e,

principalmente 2, fizeram esta média subir consideravelmente, dado que a maioria dos problemas demandaram cerca de 9 a 11 horas.

É importante ressaltar que os tempos computacionais de [12] não foram disponibilizados para o BRKGA-6r, e, mesmo para o caso de orientação fixa de caixas, foi disponibilizado apenas o tempo médio da execução da geração que fornece as soluções ótimas. Logo, não há base de comparação para os tempos computacionais, que crescem exponencialmente à medida que se aumenta o número de caixas, todas elas com a tentativa de carregamento em todas as rotações, dentro de cada possível espaço maximal vazio. Acredita-se também que computadores mais potentes possam melhorar o desempenho do tempo computacional dispendido. É importante ressaltar também que não existem registros conhecidos acerca de meta-heurísticas que resolvem o 3D-BPP, considerando-se caixas fortemente heterogêneas, e que podem ser rotacionadas em quaisquer direções.

5 ALGORITMO GENÉTICO EVOLUCIONÁRIO ESPECIALIZADO: AGE-E

5.1 INTRODUÇÃO

O Algoritmo Genético Evolucionário Especializado (AGE-E) foi idealizado após o estudo de meta-heurísticas da bibliografia que apresentam soluções de boa qualidade, não necessariamente aplicadas ao 3D-BPP, e visando cumprir com os dois últimos objetivos específicos deste trabalho de doutorado. Os estudos concentraram-se na criação de um algoritmo que disponibilizasse boas soluções para uma das categorias mais complexas de exemplos do SBSBPP, considerando-se o problema em três dimensões, com caixas fortemente heterogêneas, e que permite a rotação de itens em qualquer orientação.

Uma nova proposta de formato de cromossomo e formas de decodificá-lo, a inserção de estratégias e heurísticas que não haviam interagido entre si, bem como a criação de novas heurísticas e adaptações, vieram por contemplar a busca de redução do tempo computacional, uma vez que o AGE-E resolve uma das categorias mais difíceis de Problemas de Carregamento de Contêiner.

Inicialmente, foi desenvolvido o Algoritmo Genético Evolucionário (AGE), e, na sequência, uma variação deste algoritmo, o Algoritmo Genético Evolucionário Especializado (AGE-E), que acrescentou uma melhoria ao final de cada Ciclo Geracional do AGE: uma estrutura inteligente de busca em vizinhança, que promove a melhoria local das soluções mutante.

Neste capítulo apresenta-se o desenvolvimento do AGE, e os resultados computacionais obtidos após sua implementação, bem como o desenvolvimento e resultados do AGE-E, após a inclusão da estrutura de busca em vizinhança ao final do Ciclo Geracional. Os dados obtidos com o AGE e AGE-E são comparados com os melhores resultados da bibliografia, utilizando-se os mesmos exemplos de grande porte, para finalizar o capítulo.

5.2 A META-HEURÍSTICA AGE

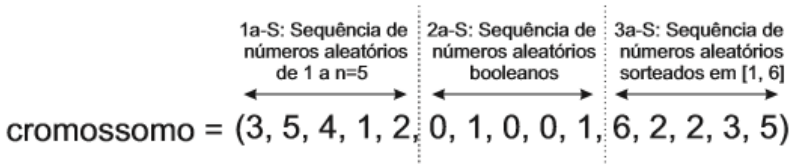
5.2.1 O Cromossomo

O termo *cromossomo* é utilizado no AGE para designar um vetor contendo um número 3*n* de genes, onde *n* é o número total de caixas, que é constituído por três sequências de *n* componentes geradas de forma aleatória ou parcialmente aleatória. A primeira sequência (1a-S) armazena números inteiros de 1 a *n*, que indicam a ordenação das caixas a serem carregadas. Essa

ordenação não é puramente aleatória, e leva em consideração a dificuldade de encaixe das caixas maiores. A segunda sequência (2a-S) é composta de números booleanos gerados aleatoriamente e preenchem os genes de posição $n + 1$ até $2n$. O valor gerado na posição $n + i$ indica qual dentre as heurísticas BBL e BLB será utilizada para promover o carregamento da caixa i. E a terceira sequência (3a-S) gera valores inteiros aleatórios no intervalo [1,6], sendo que a posição $2n+i$ indica a orientação com que a caixa deve ser carregada.

Na Figura 5.1 ilustra-se o formato dos cromossomos do AGE para um exemplo para carregar 5 caixas.

Figura 5.1 - Estrutura de Cromossomo do AGE.



Fonte: Elaboração do próprio autor.

5.2.2 A população inicial

Após a inicialização dos dados, o AGE realiza a formação de uma *População Inicial*, contendo um conjunto de p cromossomos, e tal que o encaixe das caixas mais difíceis de se arranjar, as caixas de maior volume, sejam consideradas no início do carregamento, para alguns cromossomos. Para isso, uma heurística denominada Heurística de Formação da População Inicial (HFPI) foi criada e aplicada na segunda metade dos cromossomos da População Inicial. Na primeira metade, considerou-se uma geração de cromossomos de maneira puramente aleatória, promovendo a caracterização dos conceitos sobre metaheurísticas e, em particular, sobre algoritmos genéticos. Ao final, e antes de iniciar o Ciclo Geracional do AGE, é realizado o cálculo da *fitness* de cada cromossomo da População Inicial, através de um Decodificador.

5.2.3 Formação do ciclo geracional

A estratégia de evolução do AGE foi inspirada no Algoritmo Genético de Chu-Beasley [21], e consiste na realização de dois torneios em cada iteração do *Ciclo Geracional*, os quais fornecem os dois ganhadores, os cromossomos pais para a geração de um único cromossomo descendente, o cromossomo filho. Cada torneio possui k cromossomos pais, $k = 2,3,4$, todos eles associados a uma

fitness, a qual irá possuir melhor qualidade à medida que tiver menor valor. A este processo é dado o nome *Seleção* (aleatória), que é realizado em todo início de *Ciclo Geracional*. Um operador *Recombinação* é posteriormente aplicado nos cromossomos pais, a fim de que as características dos bons carregamentos de ambos sejam herdadas para o cromossomo filho. Cada iteração do Ciclo Geracional forma um único cromossomo descendente, candidato a promover a atualização da população atual. Uma *Mutação* é realizada no descendente, antes dessa atualização, e há um cálculo de *fitness* associada a um decodificador (*Decodificador Carregamento*). Essa *fitness* é o parâmetro de comparação que faz com que haja a possibilidade de evolução da população corrente (por isso o nome *Evolucionário* atribuído ao Algoritmo Genético criado), através de um *Teste de Comparação* (TC): Se a *fitness* do descendente mutacionado é um valor melhor, se comparado à pior *fitness* da população atual, então a população é atualizada (população evoluída), excluindo-se a solução que corresponde à pior *fitness*, e incluindo-se o descendente mutacionado. Ao final do Ciclo Geracional, um *Critério de Parada (CP)* é verificado, e caso não seja satisfeito, retorna-se para o processo de Seleção de novos cromossomos pais, com a realização de dois novos torneios, e assim sucessivamente.

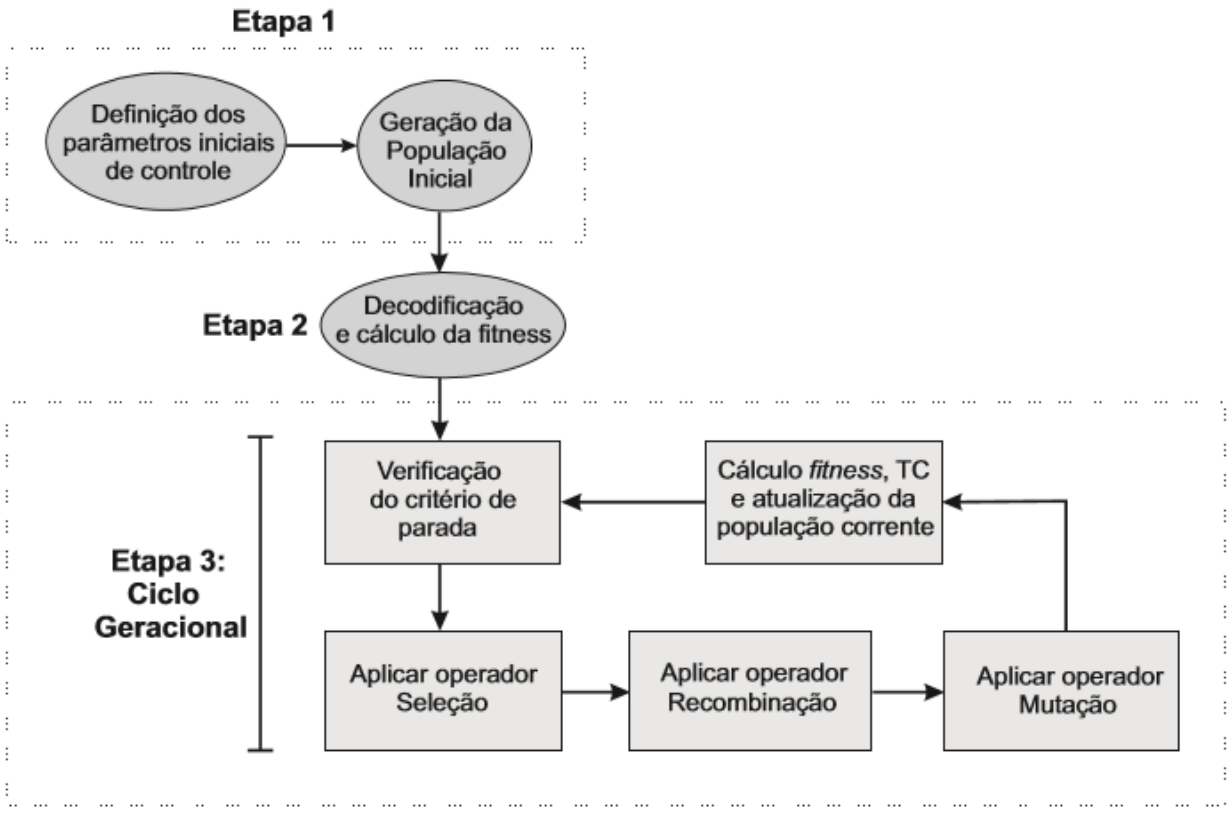
5.2.4 Visão geral da meta-heurística AGE

O algoritmo AGE foi dividido em três etapas, as duas primeiras contendo os dados referentes à inicialização do algoritmo, incluindo a geração da População Inicial, e, a terceira e última, contendo todo o Ciclo Geracional:

- **Etapa 1**: Definição dos parâmetros iniciais de controle e geração da população inicial;
- **Etapa 2**: Decodificação dos cromossomos da população inicial e cálculo das respectivas *fitness*;
- **Etapa 3**: Execução do Ciclo Geracional: (a) Verificação do CP, (b) Operador Seleção, (c) Operador Recombinação, (d) Operador Mutação, e (e) Cálculo da *fitness* do cromossomo descendente, TC (para a possível inserção do cromossomo descendente mutante na população a ser evoluída), e atualização da população corrente.

Na Figura 5.2 ilustra-se esta sequência de etapas que forma o AGE.

Figura 5.2 - Estrutura Geral do AGE.



Fonte: Elaboração do próprio autor.

5.3 DETALHAMENTO DA META-HEURÍSTICA AGE

5.3.1 A heurística HFPI

A Heurística de Formação da População Inicial (HFPI) foi desenvolvida com o objetivo de determinar uma sequência não-aleatória de caixas, tal que itens de maior volume tivessem maior probabilidade de serem carregadas primeiro. A estratégia da HFPI altera a geração aleatória da sequência 1a-S dos cromossomos, para uma ordenação que está atrelada a uma nova medida, denominada r_i, criada para cada caixa $i = 1, ..., n$. Os valores r_i são obtidos a partir de k_i, um valor aleatório no intervalo (0,1], de v_i, o volume da caixa i, e de $v_{min} = min\{v_i\}$, o volume da caixa de menor capacidade, da seguinte forma:

$$r_i = \frac{k_i}{v_i} \times v_{min} \qquad (5.1)$$

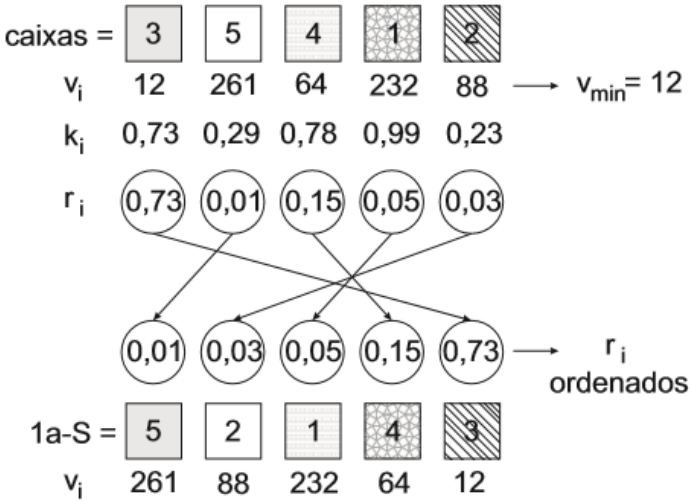
para todo $i = 1,\ldots,n$.

O valor $v_{min} = min\{v_i\}$ foi considerado apenas para garantir que a medida r_i fosse um valor tangível

para o trabalho. A fração é, portanto, o valor que proporciona valores de r_i correspondentes com a proposta de promover um ordenamento fracamente (e não estritamente) decrescente de caixas, uma vez que quanto maior v_i, maior probabilidade de que a fração seja um valor menor. Porém, é importante ressaltar que os fatores aleatórios k_i possibilitam ainda que caixas de menor volume sejam carregadas primeiramente, em detrimento de caixas de maior volume.

Na Figura 5.3 apresenta-se o vetor 1a-S do cromossomo da Figura 5.1, obtido após a aplicação da HFPI, e, na qual, é possível observar a nova sequência de carregamento das caixas *i* que estão associadas ao ordenamento crescente dos valores r_i. Inicialmente, gera-se uma sequência aleatória de caixas de volume v_i, seguida por valores aleatórios k_i em (0, 1]. Em seguida, calcula-se a medida r_i para cada caixa *i*, que são reordenadas de forma crescente, juntamente com as caixas de mesmo índice associadas. Dessa forma, não há um arranjo em que a maior caixa é carregada, seguida da segunda maior caixa, e, assim, sucessivamente. Sempre existe a possibilidade de que caixas menores possam ser carregadas logo no início do processo, com o objetivo de incutir certo grau de arbitrariedade. A caixa 2, por exemplo, possui volume 88 e, mesmo assim, foi carregada primeiro que a caixa 1, de volume 232.

Figura 5.3 - Formação da sequência 1a-S de acordo com a Heurística HFPI.



Fonte: Elaboração do próprio autor.

A heurística HFPI foi aplicada na segunda metade do total *p* de cromossomos da População Inicial (ou à $\frac{p+1}{2}$, um número ímpar). Dois propósitos nortearam a criação dessa heurística: facilitar o encaixe de caixas maiores (consideradas as caixas mais difíceis de se arranjar) para agilizar o AGE, e também simular as características de um carregamento tipicamente tradicional, que depende do fator humano, no qual caixas maiores são primeiramente carregadas.

5.3.2 A recombinação PMX

A Recombinação PMX (do inglês, *Partially - Mapped Crossover*) foi criada em 1985, por Goldberg e Lingle [104], para resolver o Problema do Caixeiro Viajante através de algoritmos genéticos. Desde então, tem sido apresentada por diversos livros e artigos como um dos tipos bem conhecidos de Recombinação que pode ser utilizado em algoritmos genéticos que resolvem problemas de otimização inteira. Considerada como uma recombinação de dois pontos, a Recombinação PMX considera dois cromossomos pais para a geração de um único cromossomo filho, copiando-se parte de uma sequência do primeiropai, e, o restante, completando com a sequência do segundo pai. Quando há a repetição de genes a serem copiados para o cromossomo descendente, um rearranjo é realizado de forma a excluir tais valores concomitantes, a fim de que valores inteiros iguais não sejam copiados para o cromossomo filho.

De acordo com a estrutura do cromossomo na meta-heurística AGE, a Recombinação PMX é aplicada apenas na sequência 1a-S, porém, os genes das sequências 2a-S e 3a-S também seguem de forma análoga com o que foi determinado para a sequência 1a-S. Isto quer dizer que o comportamento do gene relacionado à caixa i irá ocasionar o respectivo comportamento dos genes $n + i$ e $2n + i$, para cada caixa i.

A estratégia PMX, inicialmente seleciona dois pontos aleatórios $k1$ e $k2$ para indicar o início e fim da posição de uma subsequência dentro de 1a-S do primeiro cromossomo pai, p_1, que será copiada para as mesmas posições do cromossomo filho. Os demais genes do descentende são copiados do pai p_2, desde que não existam repetições de valores de gene com a subsequência do primeiro pai. As posições que existem repetições são copiadas a partir da subsequência de genes das posições $k1$ até $k2$ do segundo pai, que não tenham repetições com os genes da subsequência de p_1.

Por exemplo, sejam $p_1 = (1,2,4,6,9,3,7,5,8)$ e $p_2 = (5,4,1,9,7,2,6,8,3)$ as sequências 1a-S dos dois cromossomos pais, ou seja, considerando-se um problema que carrega nove caixas. Sorteando-se aleatoriamente $k1 = 3$ e $k2 = 6$, então a subsequência de p_1 será $s_1 = <4, 6, 9, 3>$, e a de p_2, $s_2 = <1, 9, 7, 2>$. Dado que s_1 é copiado diretamente para os genes de mesma posição para o cromossomo filho, assim como os genes não repetidos de p_2 das posições 1 até $k_1 - 1$ e de $k_2 + 1$ até n, então o descendente atualizado será $f = (5, *, 4, 6, 9, 3, *, 8, *)$. Os genes com $*$ indicam que houve repetição entre genes, e que serão copiados, portanto, dos genes de s_2, que não coincidem com os genes de s_1, na sequência com que aparecem: $1, 7, 2$ (excluindo-se o número 9 que estava repetido). Logo, o cromossomo filho será $f = (5, 1, 4, 6, 9, 3, 7, 8, 2)$.

5.3.3 O operador mutação

A mutação realizada no AGE é considerada como uma pequena perturbação no cromossomo filho gerado pela Recombinação PMX: a permuta entre dois genes do cromossomo descendente dentro de alguma das três sequências de genes.

Para isso, um valor m é sorteado aleatoriamente no intervalo $[0,1]$, a cada iteração do ciclo geracional, dentro do procedimento Mutação. Se o valor $m \in [0, 0.33]$, então a mutação é realizada na sequência 1a-S do cromossomo filho. Se $m \in [0.34, 0.66]$, a mutação acontece em 2a-S, e se $m \in [0.67, 1]$, em 3a-S.

Para haver uma mudança significativa, foi estipulado que os genes trocados devem ter uma distância mínima e, para isso, duas posições t_1 e t_2 foram geradas aleatoriamente para a permuta, tal que $|t_1 - t_2| \geq \beta n$, onde β é um percentual mínimo do número de caixas, que seja relevante para a troca de genes dentro de alguma das sequências.

5.3.4 O decodificador carregamento e cálculo da *fitness*

O decodificador utilizado no AGE foi chamado *Decodificador Carregamento*, criado considerando-se aspectos do Decodificador *Procedimento de Carregamento* [17] (descrito na Seção 3.2). Por ser a parte dependente da meta-heurística AGE, o Decodificador Carregamento pode também ser utilizado para outros tipos de problema.

Nas Subseções 3.2.3 e 3.2.4 descrevem-se as 5 etapas de decodificação do *Procedimento de Carregamento*, cuja estrutura foi mantida para o *Decodificador Carregamento*. Porém, desde que o cromossomo do AGE possui uma estrutura diferente que o cromossomo do BRKGA, o processo de decodificação também foi realizado de maneira distinta. As subsequências de genes 1a-S, 2a-S e 3a-S substituem as três primeiras etapas do processo de decodificação, no lugar dos vetores BPS, VPH e VBO, respectivamente, e, na Etapa 4, dentro da *Estratégia de Carregamento* de Lai e Chan [20]. O cálculo da *fitness* da Etapa 5 é idêntico ao descrito na Subseção 3.2.4. Logo, as etapas do *Decodificador Carregamento* seguem o formato:

1. *Etapa 1: Decodificação da sequência das caixas* - decodificação da primeira parte do cromossomo no que concerne à sequência com que as caixas são carregadas dentro do contêiner (**1a-S**);

2. *Etapa 2: Decodificação da Heurística de Carregamento* - decodificação da segunda parte

do cromossomo, no que concerne à escolha da heurística de carregamento BBL ou BLB (**2a-S**) para ser usado no *Decodificador Carregamento*;

3. *Etapa 3: Decodificação da orientação das caixas* - decodificação do vetor de orientação das caixas (**3a-S**), para ser usado no *Decodificador Carregamento*;

4. *Etapa 4: Estratégia de Carregamento Modificada* - utilização das subsequências 1aS, 2a-S e 3a-S, definidos nas etapas 1-3, para construir um carregamento de caixas dentro dos contêiners;

5. *Etapa 5: Cálculo da Fitness* - cálculo do valor αNB.

Etapas 1-3 do *Decodificador Carregamento*

A decodificação das subsequências 1a-S, 2a-S e 3a-S nas etapas 1-3 é realizada de forma direta, considerando valores inteiros gerados nos intervalos [1,*n*], [0,1] e [1,6], respectivamente, assim como apresentado na Figura 5.1. Na Etapa 1, entretanto, foi aplicada a Heurística HFPI (Subseção 5.3.1), na tentativa de iniciar o AGE já com cromossomos que podem formar boas soluções para o 3D-BPP. Na Etapa 2, realiza-se a decodificação do vetor 2a-S da seguinte forma:

$$(2a-S)_i = \begin{cases} BBL, & \text{se } \text{gene}_{n+i} = 0 \\ BLB, & \text{se } \text{gene}_{n+i} = 1 \end{cases} \tag{5.2}$$

com $i = 1, ..., n$. Na Etapa 3, se $(3a-S)_i = k$, $k \in [1, 6]$, inteiro, então carrega-se a caixa i com a orienação k.

Etapa 4 do *Decodificador Carregamento*

A Etapa 4 do *Decodificador Carregamento* também considera a *Estratégia de Carregamento* de Lai e Chan [20], assim como descrito na Subseção 3.2.4, porém com a realização de duas melhorias.

Primeiramente, apresenta-se um quinto teste de eliminação de EMS dentro do *Processo DP*, que visa à diminuição de EMS dentro das listas S_b de cada contêiner aberto b: O AGE só realiza tentativas de carregamento de uma caixa, dentro de um EMS, se o volume dessa caixa for menor do que o volume do EMS. Apesar de acrescentar um novo teste efetuado no decorrer de cada tentativa de carregamento de uma caixa, dentro de cada EMS, trata-se de um teste simples que, em

exemplos de grande porte, faz diminuir as múltiplas possibilidades de carregamento já que não há a necessidade de se tentar carregar uma caixa dentro de um EMS que é menor do que a caixa. Isso faz muita diferença, quando se consideram as seis rotações de caixas. As dimensões poderiam também ter sido testadas, comparando-se com as dimensões do EMS, mas, neste caso, acarretariam em mais prejuízos do que ganhos, no que concerne ao tempo computacional.

A segunda melhoria está relacionada à criação de uma única condição matemática que verifica as restrições de sobreposição dentro do *Processo DP*, a qual foi chamada *Relação Matemática Universal de Sobreposição* (RMUS). Grande parte da dificuldade da implementação inicial do BRKGA, na primeira parte deste trabalho, foi definir as relações matemáticas envolvidas nas restrições de projeção da geração de novos EMS, que são criados após a colocação de uma caixa que foi carregada. Essas relações são importantes, pois geram os valores de entrada do *Processo DP* que é aplicado diversas vezes tanto para EMS em que houve o carregamento da caixa, quanto em todos aqueles que são parcialmente sobrepostos pela caixa carregada (assim como descrito na Subseção 3.2.4). Apesar de os trabalhos que utilizam o *DP* dentro de seus decodificadores afirmarem que consideram as sobreposições entre caixas e EMS, nenhuma relação matemática foi encontrada, em nenhum trabalho da bibliografia. Sendo assim, foi desenvolvida neste trabalho uma única e revolucionária relação lógica-matemática para o AGE, a RMUS, que prevê as sobreposições da caixa carregada a partir de EMS utilizados ou parcialmente sobrepostos. Porém, não é possível assumir exclusividade, uma vez que não é revelado o detalhamento do processo de sobreposição utilizado em outros estudos.

Para entender como essa relação matemática de sobreposição foi criada, considera-se que $[(x_1,y_1,z_1),(x_2,y_2,z_2)]$ são: ou as coordenadas do EMS onde houve o carregamento, ou as coordenadas de um EMS parcialmente sobreposto pela caixa carregada. Considera-se ainda que $[(x_3,y_3,z_3),(x_4, y_4,z_4)]$ são as coordenadas da caixa, e os seguintes pressupostos iniciais:

$$x_1 \leq x_3 \leq x_4 \leq x_2, \qquad y_1 \leq y_3 \leq y_4 \leq y_2 \qquad \text{e} \qquad z_1 \leq z_3 \leq z_4 \leq z_2 \tag{5.3}$$

Então, a RMUS criada segue a relação lógico-matemática apresentada no pseudocódigo da Figura 5.4. A condicional principal que inicia RMUS garante que há sobreposição da caixa dentro do EMS ou em alguma parte do EMS, se, no mínimo, três condições matemáticas são verificadas. O primeiro par de condições, $[(x_3 \leq x_1 \text{ e } x_4 > x_1) \text{ ou } (x_3 \geq x_1 \text{ e } x_3 < x_2)]$, considera o caso em que a coordenada inicial em x da caixa está antes da coordenada inicial em x do EMS, porém, com coordenada final da caixa após o início do EMS, como é o caso das Figuras 5.5 e 5.8, ou com início antes do final do EMS (em x), respectivamente, como é o caso das Figuras 5.6 e 5.7. Condições

semelhantes também devem ser consideradas em termos dos eixos y e z, formando os três pares de condições. Se houver sobreposição, e, principalmente, se ela for parcial, então as coordenadas da caixa devem ser projetadas para dentro do EMS, formando-se uma *pseudo-caixa*. Esta ideia é retratada pelas seis condicionais dentro da condicional principal, que realiza a troca das coordenadas da caixa. Isto porque os valores de entrada do *Processo DP* não são as coordenadas da caixa, mas sim as coordenadas da pseudo-caixa, ou seja, da parte da caixa que está de fato sobrepondo o EMS. Logo, deve haver a transferência das coordenadas mínimas e máximas da caixas (Pontos A e C das figuras 5.5 - 5.8), para os pontos B e D (pseudo-caixa), que são as projeções dos pontos A e B sobre o EMS considerado (utilizado ou parcialmente sobreposto).

Figura 5.4 - Pseudocódigo das Relações Matemáticas de Sobreposição.

```
Se {[(x3 <= x1 e x4 > x1) ou (x3 >= x1e x3 < x2)] e
    [(y3 <= y1 e y4 > y1) ou (y3 >= y1 e y3 < y2)] e
    [(z3 <= z1 e z4 > z1) ou (z3 >= z1 e z3 < z2)] então:
        Se x3 < x1 então x3 = x1
        Se y3 < y1 então y3 = y1
        Se z3 < z1 então z3 = z1
        Se x4 > x2 então x4 = x2
        Se y4 > y2 então y4 = y2
        Se z4 > z2 então z4 = z2
```

Fonte: Elaboração do próprio autor.

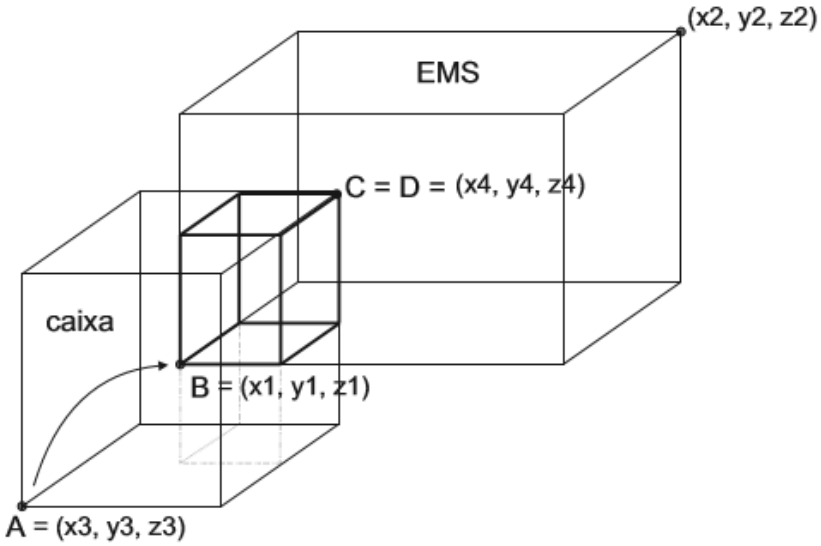
A Figura 5.5, por exemplo, possui uma caixa que sobrepõe parcialmente o EMS, verificando a condicional principal: ($x_3 \leq x_1$ e $x_4 > x_1$), e ($y_3 \leq y_1$ e $y_4 > y_1$), e ($z_3 \leq z_1$ e $z_4 > z_1$), ou seja, sobrepondo o EMS pela esquerda, pela frente, e por baixo, respectivamente. Dessa forma, houve atualização do ponto inicial A da caixa, projetando-as para o ponto B, que serão as coordenadas a serem inseridas como valores de entrada no *Processo DP*. Como $x_3 < x_1$, então $x_3 = x_1$, como $y_3 < y_1$, então $y_3 = y_1$, e como $z_3 < z_1$, então $z_3 = z_1$. Os valores finais da caixa não são atualizados, uma vez que $x_4 \not> x_2$, $y_4 \not> y_2$ e $z_4 \not> z_2$.

Na Figura 5.6 apresenta-se um exemplo em que houve sobreposição pela direita, pela frente e por baixo. Diferentemente do exemplo anterior, a caixa não apenas sobrepõe o EMS, como passa por além dele, isto é, a caixa não termina com ponto final B no meio do contêiner. São satisfeitos os pares de desigualdades na condicional principal: ($x_3 \geq x_1$ e $x_3 < x_2$), e ($y_3 \leq y_1$ e $y_4 > y_1$), e ($z_3 \leq z_1$ e $z_4 > z_1$). Atualizando-se as coordenadas para a pseudo-caixa, tem-se que uma atualização de

$y_3 = y_1$, pois $y_3 < y_1$, de $z_3 = z_1$, pois $z_3 <_1$, de $x_4 = x_2$, pois $x_4 > x_2$ e de $y_4 = y_2$, pois $y_4 > y_2$.

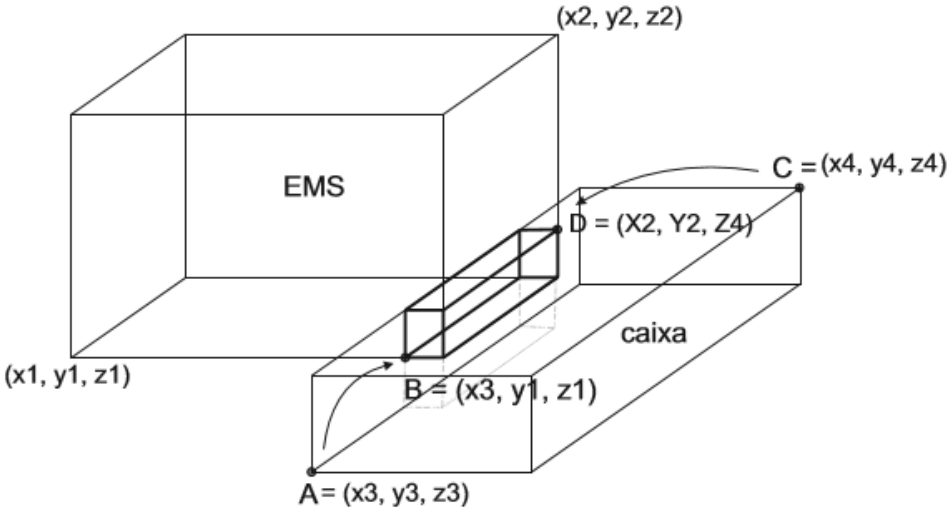
Na Figura 5.7 apresenta-se um caso em que a caixa possui coordenada $y_3 = y_1$, o que também é considerado nas restrições de igualdade da condicional principal. Logo, são satisfeitas, em x, ($x_3 \geq x_1$ e $x_3 < x_2$), e em z, ($z_3 \leq z_1$ e $z_4 > z_1$). Em y, ambos os pares de condições são satisfeitos, devido à igualdade. Há atualização de $z_3 = z_1$ e de $x_4 = x_2$ apenas.

Figura 5.5 - Exemplo 1 das Relações Matemáticas de Sobreposição.



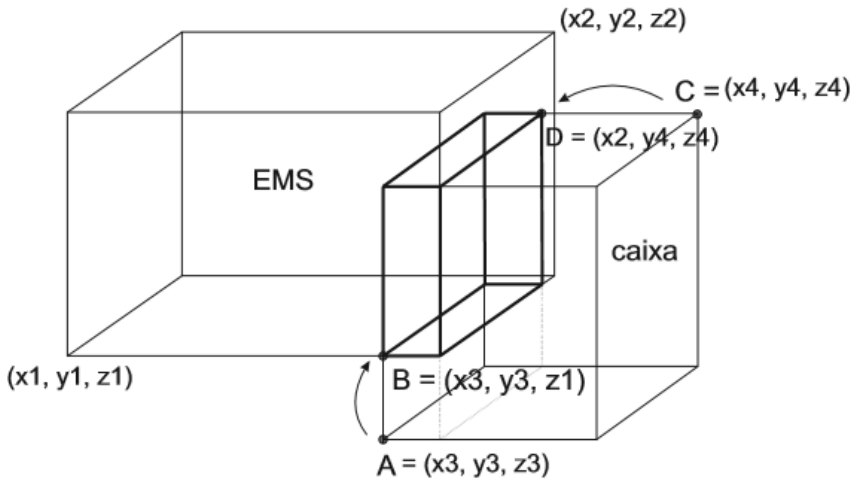
Fonte: Elaboração do próprio autor.

Figura 5.6 - Exemplo 2 das Relações Matemáticas de Sobreposição.



Fonte: Elaboração do próprio autor.

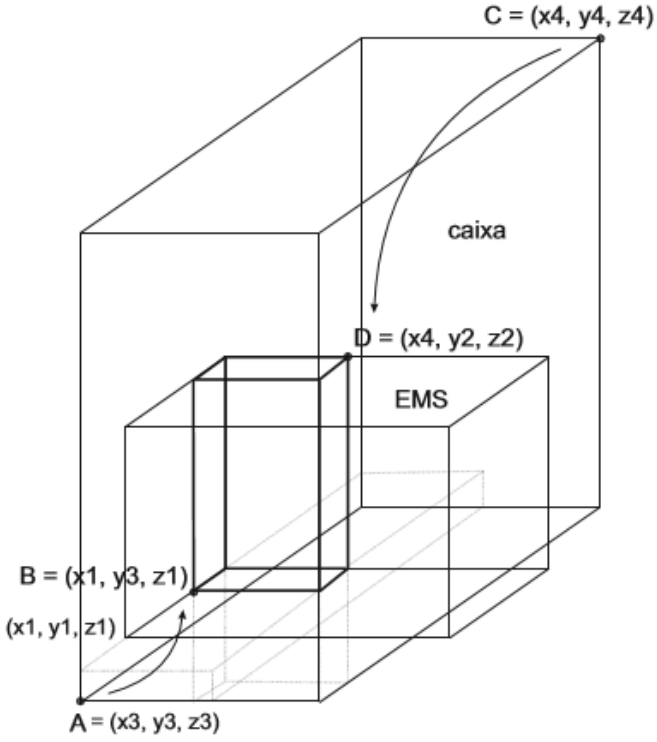
Figura 5.7 - Exemplo 3 das Relações Matemáticas de Sobreposição.



Fonte: Elaboração do próprio autor.

Na última ilustração, apresentada pela Figura 5.8, considera-se um caso em que a caixa sobrepõe o EMS pela esquerda, por trás, e por baixo, ou seja: ($x_3 \leq x_1$ e $x_4 > x_1$), e ($y_3 \geq y_1$ e $y_3 < y_2$), e ($z_3 \leq z_1$ e $z_4 > z_1$). As coordenadas inicial e final da caixa tiveram de ser transferidas dos pontos A e C para os pontos B e D, respectivamente, tomando-se $x_3 = x_1$, $z_3 = z_1$, $y_4 = y_2$ e $z_4 = z_2$.

Figura 5.8 - Exemplo 4 das Relações Matemáticas de Sobreposição.



Fonte: Elaboração do próprio autor.

Etapa 5 do *Decodificador Carregamento*

O *Decodificador Carregamento* considera o cálculo da *fitness* exatamente como é realizado no BRKGA: utilizando-se o valor αNB (Subseção 3.2.4).

5.4 A META-HEURÍSTICA AGE-E

O Algoritmo Genético Evolucionário Especializado (AGE-E) é a meta-heurística que foi de fato definida como o objeto final de estudo para este tabalho de doutorado, e que resolve o 3D-BPP. O AGE-E é uma especialização do Algoritmo Genético Evolucionário (AGE), da mesma forma que o BRKGA é definido a partir do RKGA, por Resende e Gonçalves, em seu trabalho [12]. A diferença entre AGE e AGE-E acontece no final de cada iteração do Ciclo Geracional, no qual é realizada uma tentativa de melhoria do mutante através de um Algoritmo Heurístico de Busca Através de Vizinhança, que é nomeado na literatura como SDH (do inglês, *Steepest Descent Heuristic*).

A escolha pela inclusão da Heurística SDH deve-se ao fato de que, além de ser facilmente implementável, em geral, a SDH já inicia com uma solução factível e, a partir daí, desenvolve-se à medida que vai obtendo outras soluções factíveis. Esta ideia é o que acontece quando o 3D-BPP é resolvido com o AGE-E: novas propostas de solução factíveis (propostas de carregamento viáveis) vão sendo obtidas a partir de propostas factíveis já existentes, e na tentativa de melhorar a configuração de carregamento final. Ou seja, AGE e AGE-E trabalham apenas com carregamentos factíveis e dentro das condições iniciais de existência da solução, durante todas as etapas da execução do algoritmo.

5.4.1 A heurística SDH

Assim como descrito na Subseção 2.3, as heurísticas são algoritmos simples que apresentam soluções de boa qualidade para problemas de otimização complexos, em tempo computacional compatível. Porém, para problemas de grande porte, raramente as heurísticas conseguem determinar a solução ótima devido à explosão combinatória, ou por não conseguirem escapar de ótimos locais. Assim, as heurísticas foram substitutídas pelas meta-heurísticas, que são algoritmos mais eficientes que, além de conseguirem superar os ótimos locais, têm maior probabilidade de atingir a solução ótima do problema de otimização. Entretanto, as heurísticas são métodos importantes e têm grande valor para a Pesquisa Operacional, inclusive para problemas de grande porte, principalmente por promoverem o melhoramento das soluções através de uma implementação rápida

e simples. Não demorou para que as heurísticas passassem a ser combinadas dentro das meta-heurísticas, como umas de suas estratégias, e de forma a agilizar e potencializar o processo de obtenção das soluções dentro dos algoritmos meta-heurísticos.

Segundo Romero [106], existem heurísticas que podem ser classificadas como: algoritmos heurísticos construtivos (AHC), algoritmos de divisão, algoritmos de redução, algoritmos de manipulação do modelo matemático e algoritmos de busca através de vizinhança (SDH, do inglês, *Steepest Descent Heuristic*).

O AHC, por exemplo, é muito utilizado para a resolução de problemas de otimização (principalmente de pequeno porte), ou ainda, aparece integrado à outras heurísticas ou meta-heurísticas. Existem vários tipos de AHC, o mais conhecido é o Algoritmo Guloso (do inglês, *Greedy Algorithm*). Mas o que é comum a qualquer tipo de AHC é o indicador de sensibilidade que, em cada passo do algoritmo, determina uma solução em construção e indica a direção que o AHC deve seguir, até que uma solução factível seja encontrada. De acordo com Romero [106], pode-se resumir um AHC através da seguinte forma:

1. Armazenar os dados do problema e escolher o indicador de sensibilidade a ser usado. Escolher as componentes que podem ser incorporadas na solução em construção (geralmente o processo é iniciado sem componentes).

2. Verificar se a solução em construção já representa uma solução factível. Caso seja factível então pare o processo porque foi encontrada a solução factível procurada. Caso contrário, ir ao passo 3.

3. Usando a solução em construção, resolver o problema que permite identificar o indicador de sensibilidade de todas as componentes do problema que ainda não foram incorporadas na solução em construção.

4. Usando a informação dos indicadores de sensibilidade encontrados no passo anterior identificar a componente que deve ser incorporada na solução em construção. Adicionar a componente identificada na solução em construção e voltar ao passo 2.

O AHC termina com a obtenção de uma solução factível, após uma sequência de soluções infactíveis, e essa será a solução final determinada pelo AHC.

Para SDH, no entanto, há uma relativa diferença, se comparado com as heurísticas do tipo AHC. Em geral, o SDH já inicia com uma solução factível e existe um espaço de busca contendo outras soluções factíveis. Neste caso, o SDH atualiza a solução factível atual para a melhor

solução vizinha à solução corrente. Logo, é necessário definir-se o conceito de *Vizinhança* e de geração do *Espaço de Busca* que sejam compatíveis com o problema tratado.

Romero [106] também resume o SDH, de acordo com os itens a seguir:

1. O processo de otimização é iniciado através de uma solução inicial (factível ou infactível) que passa a ser chamada de solução corrente.
2. Define-se uma estrutura de vizinhança. Assim, deve existir uma forma de identificar as soluções que são consideradas vizinhas da solução corrente. As soluções vizinhas podem ser factíveis ou infactíveis.
3. Na heurística SDH, atualiza-se a solução corrente para a melhor solução vizinha.
4. O processo termina quando todas as soluções vizinhas são de pior qualidade que a solução corrente.

A estrutura de vizinhança definida do Passo 2 está correlacionada em como as soluções estão codificadas pela própria SDH, ou pela meta-heurística na qual a SDH está inserida. Mas, de maneira geral, a estrutura de vizinhança é definida da seguinte forma: uma solução é vizinha à solução corrente, se é obtida através de algum tipo de perturbação na estrutura da solução corrente. O número de soluções vizinhas, através do mecanismo de perturbação que foi definido, forma o Espaço de Busca para a iteração atual, e é definido, em geral, empiricamente.

Neste trabalho, o SDH foi inserido na meta-heurística AGE, formando a metaheurística AGE-E, utilizando-se a estrutura de vizinhança conhecida como *2-opt*, e que é apresentada em [106] através do seguinte exemplo: Considere que existe uma proposta de solução corrente:

$$p_1 = (1, 9, 10, 7, 4, 2, 3, 6, 8, 5)$$

De acordo com *2-opt*, uma solução p_2 será vizinha à p_1 se, selecionando-se aleatoriamente duas componentes c_i e c_j de p_1, $i \geq j$, a subsequência $s_1 = (c_i, c_{i+1}, \ldots, c_{j-1}, c_j)$ é invertida em p_2, ou seja, tal que:

$$c'_{i+k} = c_{j-k}, \qquad \text{com } k = 0, 1, \ldots, j - i \tag{5.4}$$

onde c'_{i+k} são as componentes de p_2.

Por exemplo, se $i = 3$ e $j = 8$, então p_2 terá componentes iguais às de p_1, com exceção daquelas que estão inseridas na subsequência $s_1 = (10,7,4,2,3,6)$, que deverá ser invertida para $(6,3,2,4,7,10)$, ou seja, com k variando de zero até $j - i = 8 - 3 = 5$. Seguindo a estratégia (5.4), as

componentes de p_2 que pertencem à subsequência terá a inversão realizada da seguinte forma:

$$c'_3 = c'_{3+0} = c_{8-0} = c_8 = 6, \text{ com } k = 0$$
$$c'_4 = c'_{3+1} = c_{8-1} = c_7 = 3, \text{ com } k = 1$$
$$c'_5 = c'_{3+2} = c_{8-2} = c_6 = 2, \text{ com } k = 2$$
$$c'_6 = c'_{3+3} = c_{8-3} = c_5 = 4, \text{ com } k = 3$$
$$c'_7 = c'_{3+4} = c_{8-4} = c_4 = 7, \text{ com } k = 4$$
$$c'_8 = c'_{3+5} = c_{8-5} = c_3 = 10, \text{ com } k = 5$$

formando o vizinho:

$$p_2 = (1, 9, 6, 3, 2, 4, 7, 10, 8, 5)$$

5.4.2 AGE-E: especialização do AGE por SDH

A meta-heurística AGE-E segue exatamente o mesmo formato do algoritmo AGE, descrito nas seções 5.2 e 5.3, porém, com uma diferença ao final do Ciclo Geracional, para a evolução da população. A Etapa 3(e) é alterada e uma subetapa 3(f) é acrescentada, na qual o TC, apesar de ser mantido o nome, tem estratégia de comparação modicada. A atualização da população no AGE é realizada pelo TC que compara apenas a *fitness* do descendente mutante obtido ao final do *Ciclo Geracional* com a pior *fitness* de uma solução da população atual (Subseção 5.2.3). No AGE-E, antes dessa comparação, a Heurística SDH *2-opt* é inserida na parte 1a-S do cromossomo mutante, logo após o processo de Mutação, e com a finalidade de gerar vizinhos ao mutante. O comportamento dos genes das sequências 2a-S e 3a-S nesse processo também segue, de maneira análoga, ao que foi obtido para a sequência 1a-S, perturbando-se os genes correspondentes. Essa perturbação é realizada selecionando-se duas componentes de 1a-S e, após, invertendo-se as subsequências em 1a-S, 2a-S e 3a-S, formadas pelas componentes selecionadas: c_i e c_j, c_{n+i} e c_{n+j}, e c_{2n+i} e c_{2n+j}, respectivamente. Esse processo é realizado um número w de vezes, sempre sobre o cromossomo mutante original, formando portanto um Espaço de Busca composto por w soluções vizinhas à solução mutante corrente. As *fitness* são calculadas, ao final, para todos os vizinhos e mutante, finalizando a subetapa 3(e). Na sequência, 3(f) executa o TC, comparando-se as *fitness* dos $w+1$ cromossomos (mutante e seus vizinhos); a melhor proposta de solução dentre esses é aquela cuja *fitness* será comparada com a pior *fitness* da população corrente, candidata, portanto, a ser inserida na população evoluída.

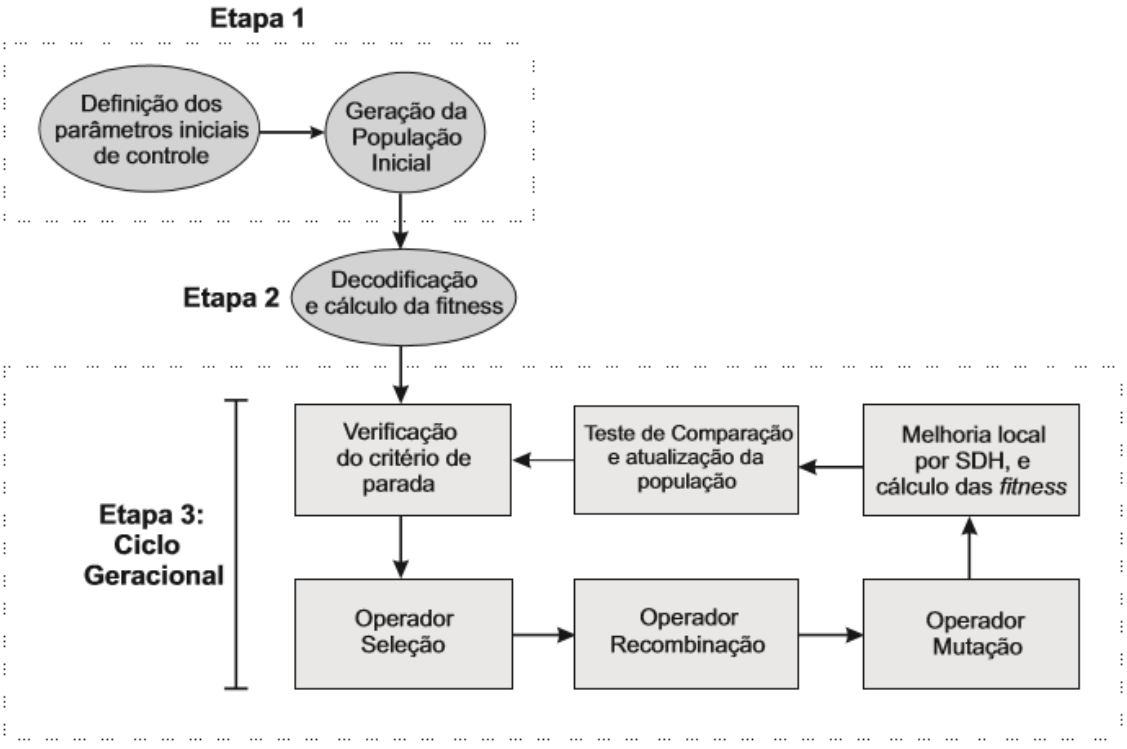
5.4.3 Visão-geral da meta-heurística AGE-E

O algoritmo de AGE-E possui o mesmo formato de etapas da meta-heurística AGE, apresentada na Subseção 5.2.4, porém, com as adaptações supracitadas, ao final da Etapa 3:

- **Etapa 1**: Definição dos parâmetros iniciais de controle e geração da população inicial;
- **Etapa 2**: Decodificação dos cromossomos da população inicial e cálculo das respectivas *fitness*;
- **Etapa 3**: Execução do *Ciclo Geracional* : (a) Verificação do *Critério de Parada*, (b) Operador *Seleção* aleatória, (c) Operador *Recombinação*, (d) Operador *Mutação*, (e) Processo de Melhoria Local por SDH e cálculo das *fitness* (do cromossomo descendente e de seus vizinhos), (f) Teste de Comparação (TC) e Atualização da População atual.

Na Figura 5.9 ilustra-se esta nova sequência de etapas que forma o Algoritmo Genético Evolucioniário Especializado.

Figura 5.9 - Estrutura Geral do AGE-E.



Fonte: Elaboração do próprio autor.

5.5 IMPLEMENTAÇÃO COMPUTACIONAL DO AGE

A implementação computacional do AGE foi realizada de acordo com os métodos e código apresentados nas subseções 5.5.1 e 5.5.2, respectivamente. Também é realizado um detalhamento da implementação do Decodificador Carregamento em 5.5.3, cuja subseção é finalizada com a apresentação do pseudocódigo do AGE. As etapas de implementação também são unificadas na Seção 5.5.4 e os exemplos de grande porte testados dispostos em 5.5.5.

5.5.1 Detalhes da implementação computacional

A implementação computacional do AGE também foi realizada utilizando-se a linguagem C por meio da ferramenta de desenvolimento Open Source DevC++, compilador GCC 4.8.1. Os resultados computacionais foram obtidos utilizando um computador com CPU Intel Core i3-M380 @2.53 GHz, com o Sistema Operacional Microsoft Windows 7 Professional (32 bits). Os métodos utilizados estão apresentados a seguir, alguns, reaproveitados do BRKGA, principalmente no que concerne à *Estratégia de Carregamento* de Lai e Chan [20], dentro do *Decodificador Procedimento de Carregamento* de Resende e Gonçalves [12]:

1. GetTickCount(): Realiza a contagem de tempo que o Algoritmo AGE utiliza para apresentar a proposta de solução final, com respectiva *fitness*.
2. AlimentarValores(): Método aproveitado da implementação do BRKGA (Subseção 4.3.3), no qual definem-se os parâmetros de entrada do algoritmo, e criam-se uma estrutura inicial para cada contêiner, contendo uma lista de EMS, uma variável booleana para indicar se o contêiner já foi aberto e uma variável que atualiza o volume utilizado dentro do contêiner. O método também inicializa as matrizes auxiliares que serão utilizadas no Processo DP: *EmsReaproveitados*, *EmsSobrepostos*, *EmsUtilizados*. Todo contêiner inicializa contendo um EMS útil de tamanho máximo, que é o próprio contêiner. E cada cromossomo possui n contêiners disponíveis para efetuar o carregamento.
3. GerarPopulacaoInicial(): Gera a população inicial utilizando duas estratégias (valores aleatórios e Heurística HFPI, respectivamente), armazenando os valores em uma matriz nomeada A_{pop} de $p = \alpha n + 1$ linhas e $3n$ colunas. As linhas da matriz são os cromossomos, e os valores de cada linha são os genes de cada cromossomo das subsequências 1a-S, 2a-S e 3a-S. A primeira estratégia gera a primeira metade de cromossomos da população

inicial, cromossomo 1 a $\frac{p}{2}$, de maneira puramente aleatória e tal como definido na Subseção 5.2.2. A segunda estratégia privilegia o carregamento de caixas maiores, utilizando a Heurística HFPI (Subseção 5.3.1). Uma linha a mais é adicionada e preenchida inicialmente apenas com valores iguais a zero, sendo posteriormente utilizada para armazenar os dados do cromossomo descendente mutante.

4. CarregarCaixa(): Método aproveitado da implementação do BRKGA, verifica qual o primeiro contêiner aberto, para cada cromossomo. Em seguida, verifica qual o primeiro EMS útil de cada contêiner aberto em que seja possível carregar a caixa atual *i*, descrita na matriz A_{pop}, que é então atualizada a cada iteração. Esse método será repetido para todas as caixas e para todos os cromossomos.

5. VerificarSobreposicao(): Verifica se a caixa carregada sobrepõe EMS úteis dentro do contêiner utilizado e de acordo com o pseudocódigo 5.4. Se verdadeiro, o EMS sobreposto será subtraído da parcela da caixa sobreposta pelo *Processo DP* dentro do método *CalcularEMSsobrepostos* (abaixo). Caso contrário, o EMS não sobreposto, será armazenado na matriz auxiliar *EMSReaproveitados*.

6. CalcularEMSsobrepostos(): Calcula o processo DP para os EMS que foram sobrepostos pela caixa atual carregada de acordo com o pseudocódigo 5.4, não incluindo o próprio EMS selecionado para realizar o carregamento. Os EMS gerados são armazenados na matriz *EMSsobrepostos*.

7. TestarEMSsobrepostos(): Verifica cinco testes de eliminação de EMS (quatro destes aproveitados da implementação do BRKGA), em todas as listas de EMS de cada contêiner que foram sobrepostos pela caixa atual. É um teste prévio, visando à diminuição da lista final de EMS do método *calcularEMS*.

8. CalcularEMS(): Método aproveitado da implementação do BRKGA, calcula o *Processo DP* apenas no EMS em que a caixa *i* foi carregada. São gerados seis EMS que são armazenados na lista de EMS do contêiner utilizado. Nesta mesma lista, também são incluídos os EMS contidos nas matrizes *EMSsobrepostos* e *EMSreaproveitados*.

9. TestarEMS(): Verifica os cinco testes de eliminação de EMS, na lista de EMS do contêiner utilizado, quatro destes, aproveitados da implementação do BRKGA.

10. OrdenarEMS(): Ordena a lista de EMS do contêiner utilizado, segundo as heurísticas BBL e BLB dentro do *Processo DP*, e a partir da decodificação da subsequência 2a-S.

11. CalcularFitness(): Método aproveitado da implementação do BRKGA, calcula o volume não utilizado de todos os contêiners abertos, finalizando com o cálculo do valor da *fitness* final.

12. OrdenarPopulacaoInicial(): Promove a ordenação das p primeiras linhas da matriz A_{pop} (desconsiderando a última linha que é destinada aos valores do cromossomo mutante) de acordo com a ordem crescente das *fitness* associadas a cada linha (cromossomo).

13. GerarPais(): Seleciona aleatoriamente quatro linhas distintas da matriz A_{pop}. Após, compara a *fitness* associada às duas primeiras linhas da matriz, e a que possuir menor valor será o primeiro cromossomo pai selecionado. Compara, na sequência, as *fitness* associadas às duas últimas linhas selecionadas aleatoriamente, computando a linha com menor valor de *fitness* como sendo o segundo cromossomo pai escolhido a gerar um descendente.

14. RecombinacaoPMX(): Gera um número aleatório k_1 até a metade da sequência 1a-S, e um segundo número aleatório k_2 que pode variar desde k_1 até 50% do tamanho de 1a-S. Em seguida, realiza a Recombinação PMX entre as linhas selecionadas pelo método GerarPais(), segundo a lógica descrita em 5.3.2. Analogamente ao que acontece com a recombinação das subsequências 1a-S dos dois cromossomos pais, executa-se também para os genes associados das subsequências 2a-S e 3a-S.

15. MutacaoFilho(): Gera aleatoriamente dois valores que possuam uma distância de t% entre si, porcentagem esta que é definida como um parâmetro de entrada para o algoritmo AGE. Depois, o algoritmo permuta as caixas da subsequência 1a-S, e os respectivos valores associados nas subsequências 2a-S e 3a-S, armazenando os dados na última linha da matriz A_{pop}, definindo esta como o cromossomo filho mutante da iteração i.

16. CriarCarregamentoMutante(): Realiza a execução do Algoritmo AGE apenas para a última linha da matriz A_{pop}, utilizando desde o método 4, CarregarCaixa, até o método 10, ordenarEMS.

17. VerificarFitnessMutante(): Calcula-se a *fitness* apenas para a última linha da matriz A_{pop} (cromossomo mutante) através do método 11, calcularFitness(). Realiza-se também a execução do método 12, ordenarPopulacaoInicial(), mas considerando todas as linhas da matriz A_{pop}, incluindo a linha que inclui o mutante descendente (que pode ou não estar incluído na atualização da população, dentro das p primeiras melhores linhas).

18. ListarMelhorCarregamento(): Apresenta a melhor solução encontrada em k execuções

do algoritmo.

19. Main(): É o método principal do programa que descreve toda a lógica do AGE-E. Unindo todas as informações, apresenta a melhor solução encontrada a partir da sequência de métodos: GetTickCount(), AlimentarValores(), GerarPopulacaoInicial(), CarregarCaixa(), CalcularFitness(), OrdenarPopulacaoInicial(), GerarPais(), RecombinacaoPMX(), MutacaoFilho(), CriarCarregamentoMutante(), CarregarCaixa() (mesmo método aplicado, dessa vez, ao cromossomo filho mutante), CalcularFitness() (novamente, aplicado ao descendente mutante), VerificarFitnessMutante(), OrdenarPopulacaoInicial(), ListarFitness() e ListarMelhorCarregamento(), apresenta a melhor solução encontrada.

5.5.2 Etapas da implementação do AGE

A implementação computacional do AGE é dividida em sete passos, Passo 0 a Passo 6, nos quais compreendem as três etapas do algoritmo (apresentadas na Subseção 5.4.3), como a seguir:

- *Passo 0 (Etapa 1 - Definição dos Parâmetros Iniciais e População Inicial)*: O método GetTickCount inicia a contagem de tempo de execução do algoritmo para cada exemplo testado. Definem-se o parâmetro inicial de controle, $p = \alpha n$, bem como os demais valores de entrada secundários e matrizes auxiliares, através do método AlimentarValores. Uma população inicial de vetores é gerada pelo método gerarPopulacaoInicial, que é armazenada na matriz A_{pop} de dimensão $(p+1) \times 3n$, em que a i-ésima linha da matriz corresponde ao i-ésimo indivíduo da população, para todo $i = 1,...,p$, e tal que a linha $p+1$ é inicializa com valores nulos. As linhas 1 a $\frac{p}{2}$ são preenchidas com números aleatórios da seguinte forma: colunas 1 a n com valores inteiros aleatórios de 1 a n (sequência 1a-S); colunas $n+1$ a $2n$ com valores booleanos 0 ou 1 (sequência 2a-S); e colunas $2n+1$ a $3n$, com valores inteiros de 1 a 6 (sequência 3a-S). As linhas $\frac{P}{2}+1$ a p são preenchidas de acordo com a Heurística HFPI: primeiramente, geram-se os valores aleatórios de maneira idêntica ao estabelecido para as $\frac{P}{2}$ primeiras linhas; depois, geram-se valores aleatórios $k_i \in [0,1)$, $i = 1,...,n$, para o cálculo dos parâmetros r_i; e, por último, reordena-se 1a-S (e os genes correspondentes de 2a-S e de 3a-S) de acordo com a ordenação crescente dos valores r_i associados ao índice i dos respectivos genes.

- *Passo 1 (Etapa 2 - Decodificação e Cálculo da Fitness)*: Utilização do *Decodificador*

Carregamento, compreendido como a utilização dos métodos 4 a 11, CarregarCaixa, VerificarSobreposicao, CalcularEMSsobrepostos, TestarEMSsobrepostos, CalcularEMS, TestarEMS, OrdenarEMS e calcularFitness, para decodificar todos os cromossomos contidos nas primeiras p linhas da matriz A_{pop} (em formato codificado), obtendo-se também os valores de *fitness* associados que são armazenadas em um vetor $f_{fitness}$, vetor de dimensão $(p + 1) \times 1$. Esse vetor é reorganizado, contendo suas componentes em ordem crescente de valores, bem como as linhas da matriz A_{pop}, por meio do método ordenarPopulacaoInicial.

- *Passo 2 (Etapa 3(a) - Início do Ciclo Geracional)*: Verificação do Critério de Parada (CP). Caso CP = falso, dar continuidade ao Ciclo Geracional contido nos Passos 3 a 6.

- *Passo 3 (Etapa 3(b) - Seleção)*: No método GerarPais, executa-se o operador Seleção aleatória em quatro linhas da matriz A_{pop} e compara-se, duas a duas, entre as componentes associadas às linhas selecionadas do vetor $f_{fitness}$ (dois torneios de dois possíveis propostas de solução pais). Os ganhadores dos dois torneios são aqueles que possuem menores *fitness* dos dois conjuntos de três valores *fitness*. As linhas de A_{pop} que forneceram as menores *fitness* são selecionadas e guardadas como índice das linhas p_1 e p_2, constituindo-se, portanto, como os cromossomos pais da iteração atual que irão gerar um único descendente, no Passo 4.

- *Passo 4 (Etapa 3(c) - Recombinação PMX)*: Uma única nova solução é obtida através da Recombinação PMX (método recombinacaoPMX) geram-se dois valores inteiros aleatórios k_1 e k_2 no intervalo 1 a $\frac{n}{2}$, para k_1, e k_1 a n, para k_2; criam-se os vetores s_1 e s_2 (subsequências s_1 e s_2 da Subseção 5.3.2) tal que $s_1(i) = A_{pop}(p_1, k_1 + i - 1)$ e $s_2(i) = A_{pop}(p_2, k_1 + i - 1)$, $i = 1, \ldots, k_2 - k_1 + 1$; em seguida, cria-se o vetor *cromo-filho* de $3n$ componentes e de genes iguais a zero, que é atualizado de acordo com a estratégia PMX. Para essa atualização, inicialmente copiam-se para os genes de *cromo-filho* os valores das componentes de s_1: *cromo-filho*$(k_1 + i - 1) = s_1(i)$. Se $k_1 f = 0$, então existem genes antes da subsequência de genes de k_1 a k_2 para serem atualizados no vetor *cromo-filho*. Para isso, conta-se o número l_1 de genes de p_2, das posições 1 a $k-1$, que possuem valores iguais aos armazenados no vetor s_1, copiando-se nas componentes de posição 1 a k_1-1 do *cromo-filho*, que são diferentes dos genes de s_1, os genes de p_2 de índices associados: *cromo-filho*$(j) = p_2(j)$, $j = 1, \ldots, k_1 - 1$, quando $p_2(j)\ f = s_1(i)$, para todo $i = 1, \ldots, k_2 - k_1 + 1$. Para as l_1 posições não copiadas, copiam-se os l_1 primeiros valores de s_2 que

também são distintos dos valores de s_1. Um contador armazena o número de posições de s_1, l_3, que obtiveram valores de s_2 iguais aos de s_1, até obter as l_1 primeiras posições de genes distintos. De forma análoga, se k_2 f= n, então existem genes depois da subsequência de genes de k_1 a k_2 para serem atualizados no vetor *cromo-filho*. Então, conta-se o número l_2 de genes de p_2, das posições $k_2 + 1$ a n, que possuem valores iguais aos armazenados no vetor s_1, copiando-se nas componentes de posição k_2+1 a n do *cromo-filho*, que são diferentes dos genes de s_1, os genes de p_2 de índices associados: *cromo-filho*(j) = $p_2(j)$, $j = k_2 + 1, \ldots, n$, quando $p_2(j)$ f= $s_1(i)$, para todo $i = 1,\ldots,k_2-k_1+1$. Para as l_2 posições não copiadas, copiam-se os l_2 valores de s_2, investigados a partir das posições l_1+l_3+1 até k_2, que também são distintos dos valores de s_1. Essa estratégia da Recombinação PMX refere-se estritamente à formação dos primeiros n componentes do *cromo-filho* (sequência 1a-S): *cromofilho*(l), $l = 1, \ldots, n$. Porém, as sequências 2a-S e 3a-S também são formadas em *cromo-filho* (posições n+1 a 3n), de acordo com o que foi determinado na formação de 1a-S. Isto quer dizer que se *cromo-filho*(l) é atualizado como um gene m de p_1, $p_1(m)$, para algum $m = 1, \ldots, n$, ou seja, *cromo-filho*(l) = $p_1(m)$, então também atualizam-se *cromo-filho*($l + n$) = $p_1(m + n)$ e *cromo-filho*($l + 2n$) = $p_1(m + 2n)$. Análogo, se a atualização de *cromo-filho* for a partir de p_2.

- *Passo 5 (Etapa 3(d) - Mutação)*: Utilizando-se o método MutacaoFilho, geram-se dois valores inteiros aleatórios t_1 e t_2, de 1 a n, que tenham uma distância de t% entre si. O cromossomo filho mutante tem os valores correspondentes à posição t_1, $t_1 + n$ e $t_1 + 2n$ permutados, respectivamente, com os genes da posição t_2, $t_2 + n$ e $t_2 + 2n$. Ao final, o método criarCarregamentoMutante é executado para construir um carregamento para o cromossomo mutante.

- *Passo 6 (Etapa 3(e) - Fitness e Atualização da População)*: Utilizando o *Decodificador Carregamento*, calcula-se a *fitness* do descendente mutante através do método verificarFitnessMutante, comparando-a com a pior *fitness* da população inicial (que está armazenada na primeira posição do vetor $f_{fitness}$, por ser o maior valor). Se a *fitness* do descendente mutante for menor que a pior *fitness*, então a pior fitness é descartada, juntamente com a respectiva linha da matriz A_{pop}. Neste caso, há uma atualização da população, tal que os valores da fitness e os genes do cromosso mutante são inseridos no vetor $f_{fitness}$ (preservando-se a ordenação crescente de valores) e na matriz A_{pop}, respectivamente, voltando o algoritmo ao Passo 2. O cromossomo descartado é atualizado para a linha p+ 1 de A_{pop}. Caso a *fitness* do descendente mutante for igual ou pior à fitness da primeira componente do vetor $f_{fitness}$, então, retorna-se diretamente ao Passo 2, para uma nova tentativa de produzir um mutante que forneça o melhoramento da população atual.

5.5.3 Implementação do decodificador e pseudocódigo AGE

O *Decodificador Carregamento* é compreendido pelos métodos 4 a 12 da Subseção 5.5.1, e, apesar de alguns nomes de métodos terem sido mantidos a partir daqueles que fizeram parte da implementação do BRKGA, muitas modificações substanciais foram realizadas, a fim de estabelecer-se uma adaptação com a nova estrutura do AGE. Primeiramente, não há a criação de vetores BPS, VPH e VBO, a decodificação é realizada diretamente das sequências 1a-S, 2a-S e 3a-S, e de tal forma que a geração dos cromossomos já é realizada com a finalidade de privilegiar essa facilitação (Etapas 1 a 3 do Decodificador). Na Etapa 4, utiliza-se da *Estratégia de Carregamento* que também foi melhorada no que se refere aos métodos 5, 6 e 7, VerificaSopreposicao, CalcularEMSsobrepostos e TestarEMSsobrepostos, respectivamente, e assim como descrito na Subseção 5.5.1. Os métodos 9, TestarEMS, e 10, OrdenarEMS, também foram modificados, porém, parcialmente. Outros métodos periféricos também foram utilizados, mas como auxiliares dos métodos principais de 5.5.1. O método 12, ordenarPopulacaoInicial, foi inserido ao final do *Decodificador Carregamento*.

O pseudocódigo da Figura 5.10 apresenta todo o algoritmo AGE, incluindo o detalhamento do *Decodificador Carregamento*, a partir do comentário *Processo de Carregamento de Caixas - Método Decodificador* (aplicado para todos os cromossomos da População Inicial, linhas 1 a p de A_{pop}, na Etapa 2). A partir do comentário *Início do Ciclo Geracional*, tem início o começo da Etapa 3, a partir de *Verificação do Critério de Parada*, até *Atualização da População*. Dentro da implementação da Etapa 3, a mesma estrutura de implementação da Etapa 2 é realizada em *Método Decodificador*, porém, aplicado somente ao cromossomo mutante, armazenado na linha $p + 1$ de A_{pop}. É reaproveitado o decodificador completo, não sendo necessário implementar-se uma segunda vez. Uma variável booleana foi definida para este procedimento: se igual a zero, o *Decodificador Carregamento* é aplicado para a População Inicial; se igual a um, é aplicado para a última linha de A_{pop}, linha $p + 1$, onde está armazenado o cromossomo mutante.

No detalhamento do *Decodificador Carregamento* ainda é possível observar-se que há a inicialização da transformação de cada cromossomo da matriz A_{pop} em uma *fitness* para o problema original. Em seguida, para cada cromossomo, em cada contêiner aberto, e para o primeiro EMS útil possível, é realizado o carregamento da caixa atual, unindo-se os conceitos teóricos previamentes descritos: RMUS, Processo DP e Testes de Eliminação de EMS que serão utilizados para a próxima iteração.

Figura 5.10 - Pseudocódigo do AGE com detalhamento do Decodificador Carregamento.

```
// Inicialização
TotalCromossomos = an;     // n = número de caixas
Definição dos parâmetros iniciais de controle

1a-S()  // Recebimento da ordem de caixas a serem carregadas
2a-S()  // Geração das heurísticas de carregamento, por caixa
3a-S()  // Determinação da orientação inicial das caixas

// Processo de Carregamento de Caixas - Método Decodificador
Para t = 1 até TotalCromossomos
Início
  Para k = 1 até n
  Início
    Para i = 1 até últimoCONTEINERutilizado
    Início
      Verificar se o contêiner i já está aberto
      Para b = 1 até últimoEMSutilizado
      Início
        Para c = 1 até 6        // Número total de rotações
        Início
          Se a caixa atual pôde ser carregada, então
          Início
            Atualizar informações da caixa carregada
            Vericar RMUS
            Realização do Processo DP
            Para d = 1 até totalEMS
              Início
                Teste de Volumes
                Teste 1 de [10]
                Teste 2 de [10]
                Eliminação de EMS inscritos [13]
                Eliminação de EMS de larguras ínfimas [13]
              Fim
            Atualização da lista de EMS
          Fim
        Fim
      Fim
    Fim
  Fim
  CalcularFitness
Fim

Verificação do Critério de Parada     //Início do Ciclo Geracional
Operador Seleção
Operador Recombinação
Operador Mutação
Método Decodificador      //Aplicado ao cromossomo  mutante
Teste de Comparação
Atualização da População

// Finalização
```

Fonte: Elaboração do próprio autor.

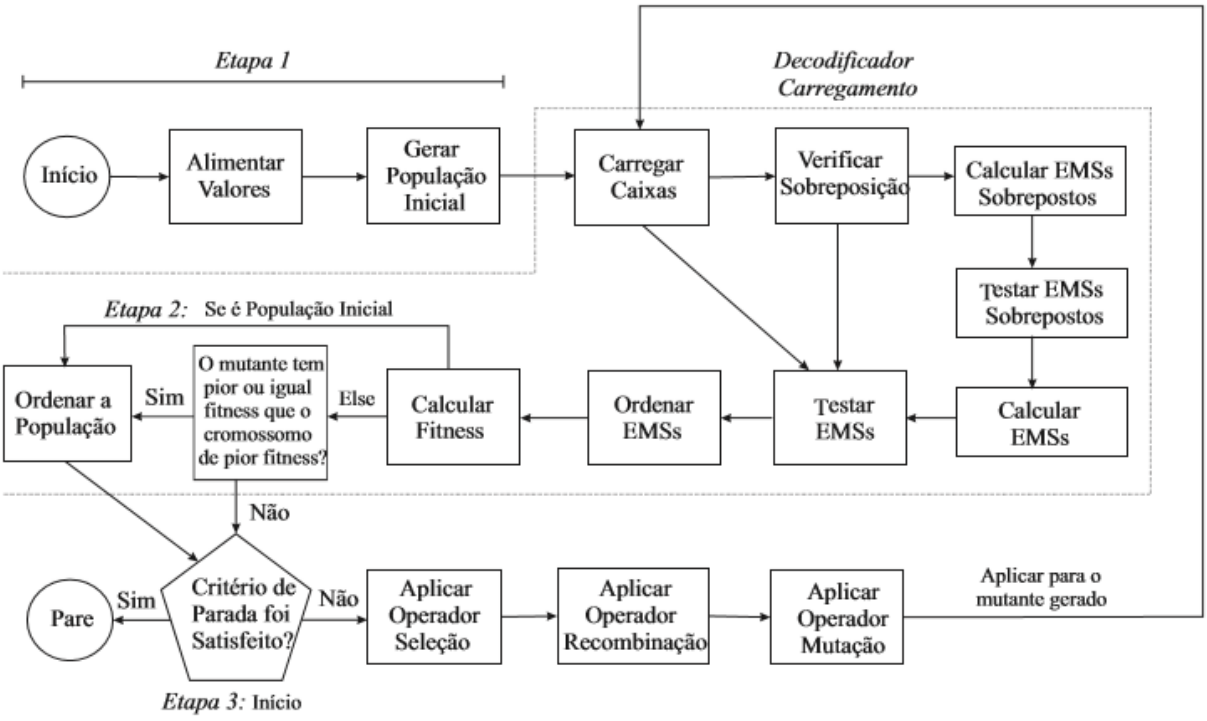
5.5.4 Visão global da implementação do AGE

A estrutura da implementação completa da meta-heurística AGE é apresentada em formato de esquema, na Figura 5.11, no qual considera a cadeia lógica do algoritmo. Como anteriormente apresentado, o *Decodificador Carregamento* pode ser reaproveitado para outros tipos de problema, uma vez que configura-se como a parte dependente da implementação.

Mais uma vez, apresenta-se o processo de decodificação tanto para a Etapa 2, para calcular a *fitness* de todos os cromossomos da População Inicial, quanto para a Etapa 3, no cálculo das *fitness* do mutante descendente de cada iteração.

A evolução da População só é realizada (atualização de A_{pop}) da iteração atual para a próxima, caso a *fitness* do mutante (linha p + 1) seja melhor do que a pior *fitness* do cromossomo que já se encontra na linha p da matriz A_{pop} (após aplicação do Método 12). Essa atualização é automática a partir da reordenação da População corrente, que exclui o cromossomo que possui pior fitness, para a posição p+1, que será posteriormente sobreescrito pelas componentes do próximo vetor mutante. Nessa reordenação, o cromossomo mutante que foi inserido na população atual é atualizado para alguma linha da matriz A_{pop}, de acordo com a ordenação das *fitness* entre a população atual (sem considerar o cromossomo já excluído) e o cromossomo mutante (Método 12).

Figura 5.11 - Estrutura Completa da Meta-Heurística AGE.



Fonte: Elaboração do próprio autor.

5.5.5 Os exemplos de grande porte

Os exemplos de grande porte usados nos testes computacionais foram extraídos do trabalho de Martello *et. al* [9], assim como realizado para os testes com o BRKGA, descritos na Subseção 4.4.2. Porém, para o BRKGA, foi considerado apenas uma das oito categorias de dez problemas, a *Classe 1*, considerando-se o carregamento de 50 caixas.

Para o AGE, foram testadas todas as oito categorias de problemas 3D-BPP envolvendo 50 caixas, nomeadas, assim como em Resende e Gonçalves [12], por *Classe 1* a *Classe 8*. As classes 1 a 5 têm contêiners que possuem todas as dimensões sempre iguais a 100, e que são geradas de acordo com os tipos 1 a 5, descritos na Subseção 4.4.2. Para uma *Classe k*, $k = 1, \ldots, 5$, uma caixa é gerada com 60% de probabilidade de ter suas dimensões geradas a partir do *Tipo k*. A Categoria 6 a 8 foram geradas de forma diferente, com contêiners de dimensões idênticas e iguais a 10, 40, e 100, respectivamente. Estas últimas três categorias possuem caixas cujas componentes w_j, h_j e d_j são números inteiros aleatórios pertencentes aos intervalos [1,10], [1,35], e [1,100], nesta ordem. No Apêndice A apresentam-se todas as dimensões de caixa, de todos os exemplos, dentro de cada categoria, a partir da Tabela A.1 até a Tabela A.40.

Os mesmos exemplos de grande porte também foram utilizados nos trabalhos que estão apresentados na Tabela 5.1, e, por isso, serviram de referência para a comparação dos resultados obtidos e para a análise de desempenho do AGE. Todos os algoritmos comparados consideram apenas o carregamento das caixas em uma única orientação, a orientação inicial. Resende e Gonçalves [12], implementaram, além disso, o BRKGA, com caixas rotacionadas em qualquer orientação possível, mas não é apresentado o tempo total de implementação. Dessa forma, o AGE foi implementado para o carregamento de caixas em orientação única e em múltiplas orientações, e, para todos os casos, apresenta o tempo computacional global para obtenção das soluções finais.

Tabela 5.1 - Algoritmos Eficientes para o Teste de Execução.

Algoritmo	Trabalho de origem	Tipo de método
BRKGA	Resende e Gonçalves [12]	Algoritmo Genético
TS3	Lodi et al. [73] e Lodi et al. [74]	Busca Tabu
GLS	Faroe et al. [72]	Busca Local Guiada
TS^2PACK	Crainic et al. [76]	Busca Tabu Paralela
GVND	Parreño et al. [81]	GRASP/VND

Fonte: Adaptado de Resende e Gonçalves (2013) [12].

Por apresentarem bons resultados, os parâmetros dos dados iniciais de controle foram definidos como:

- $p = \alpha \times n$, com $\alpha = 10$;
- $|t_1 - t_2| = t = \beta * n$, com $\beta = 0{,}05$.

Estes valores indicam que a população considera um número de cromossomos dez vezes maior do que o número total de caixas, e que a distância entre os genes selecionados para mutação deve ser de 5% entre si. Uma investigação empírica dos parâmetros iniciais de controle foi realizada para determinar o impacto com que a distância entre os genes mutacionados influenciam na alteração dos resultados finais. O valor p = 10 foi pré-definido, de acordo com os resultados da implementação do BRKGA.

O CP foi estabelecido como sendo o de 300 gerações do Ciclo Geracional, idêntico ao considerado pelo BRKGA, para maior correlação na análise de desempenho.

5.6 IMPLEMENTAÇÃO COMPUTACIONAL DO AGE-E

Os métodos utilizados na implementação computacional da meta-heurística AGE-E são os mesmos que foram apresentados na Seção 5.5.1. Alterações foram realizadas ao final das etapas de implementação, apresentadas inicialmente na Seção 5.5.2, e, para demais etapas, no que concerne à dimensão de A_{pop}. As Etapas da implementação do AGE-E são apresentadas na Subseção 5.6.1, o pseudo-código na Subseção 5.6.2, e visão global, na Subseção 5.6.3. Os dados computacionais e parâmetros iniciais de controle são os mesmos que utilizados para o AGE, e apresentados na Subseção 5.5.5.

5.6.1 Etapas da implementação do AGE-E

A implementação do algoritmo AGE-E foi dividida em um conjunto de oito passos, um a mais do que o estipulado para o algoritmo AGE, devido à inserção da Etapa 3(f), ao final da implementação. Além disso, os passos anteriores são semelhantes em ambos os algoritmos, com exceção da Etapa 3(e), que é alterada de AGE para AGE-E, e das dimensões da matriz A_{pop}, que inclui o espaço para inserção de w vizinhos. Todos os passos do algoritmo AGE-E são apresentados a seguir.

- *Passo 0 (Etapa 1 - Definição dos Parâmetros Iniciais e População Inicial)*: O método GetTickCount inicia a contagem de tempo de execução do algoritmo para cada exemplo testado. Definem-se o parâmetro inicial de controle, $p = \alpha n$, bem como os demais valores de entrada secundários e matrizes auxiliares, através do método AlimentarValores. Uma população inicial de vetores é gerada pelo método gerarPopulacaoInicial, que é armazenada na matriz A_{pop} de dimensão $(p+w+1)\times 3n$, em que a i-ésima linha da matriz corresponde ao i-ésimo indivíduo da população, para todo $i = 1,...,p$, e tal que a linha $p + w + 1$ inicializa com valores nulos. As linhas 1 a $\frac{P}{2}$ são preenchidas com números aleatórios da seguinte forma: colunas 1 a n com valores inteiros aleatórios de 1 a n (sequência 1a-S); colunas $n + 1$ a $2n$ com valores booleanos 0 ou 1 (sequência 2a-S); e colunas $2n+1$ a $3n$, com valores inteiros de 1 a 6 (sequência 3a-S). As linhas $\frac{P}{2} + 1$ a p são preenchidas de acordo com a Heurística HFPI: primeiramente, geram-se os valores aleatórios de maneira idêntica ao estabelecido para as $\frac{P}{2}$ primeiras linhas; depois, geram-se valores aleatórios $k_i \in [0,1)$, $i = 1,...,n$, para o cálculo dos parâmetros r_i; e, por último, reordena-se 1a-S (e os genes correspondentes de 2a-S e de 3a-S) de acordo com a ordenação crescente dos valores r_i associados ao índice i dos respectivos genes.

- *Passo 1 (Etapa 2 - Decodificação e Cálculo da Fitness)*: Utilização do *Decodificador Carregamento*, composto pelos métodos 4 a 11, CarregarCaixa, VerificarSobreposicao, CalcularEMSsobrepostos, TestarEMSsobrepostos, CalcularEMS, TestarEMS, OrdenarEMS e calcularFitness, para decodificar todos os cromossomos contidos nas primeiras p linhas da matriz A_{pop} (em formato codi-ficado), obtendo-se também os valores de *fitness* associados que são armazenadas em um vetor $f_{fitness}$, de dimensão $(p+1)\times 1$. Esse vetor é reorganizado, contendo suas componentes em ordem crescente de valores, bem como as linhas da matriz A_{pop}, por meio do método ordenarPopulacaoInicial.

- *Passo 2 (Etapa 3(a) - Início do Ciclo Geracional)*: Verificação do Critério de Parada (CP). Caso CP = falso, dar continuidade ao Ciclo Geracional contido nos Passos 3 a 6.

- *Passo 3 (Etapa 3(b) - Seleção)*: No método GerarPais, executa-se o operador Seleção aleatória em quatro linhas da matriz A_{pop} e compara-se, duas a duas, entre as componentes associadas às linhas selecionadas do vetor $f_{fitness}$ (dois torneios de dois possíveis propostas de solução pais). Os ganhadores dos dois torneios são aqueles que possuem menores *fitness* dos dois conjuntos de três valores *fitness*. As linhas de A_{pop} que forneceram as menores *fitness* são selecionadas e guardadas como índice das linhas p_1 e p_2, constituindo-

se, portanto, como os cromossomos pais da iteração atual que irão gerar um único descendente, no Passo 4.

- *Passo 4 (Etapa 3(c) - Recombinação PMX)*: Uma única nova solução é obtida através da Recombinação PMX (método recombinacaoPMX) geram-se dois valores inteiros aleatórios k_1 e k_2 no intervalo 1 a $\frac{n}{2}$, para k_1, e k_1 a n, para k_2; criam-se os vetores s_1 e s_2 (subsequências s_1 e s_2 da Subseção 5.3.2) tal que $s_1(i) = A_{pop}(p_1, k_1 + i - 1)$ e $s_2(i) = A_{pop}(p_2, k_1 + i - 1)$, $i = 1, \ldots, k_2 - k_1 + 1$; em seguida, cria-se o vetor *cromo-filho* de $3n$ componentes e de genes iguais a zero, que é atualizado de acordo com a estratégia PMX. Para essa atualização, inicialmente copiam-se para os genes de *cromo-filho* os valores das componentes de s_1: *cromo-filho*$(k_1 + i - 1) = s_1(i)$. Se $k_1 \neq 0$, então existem genes antes da subsequência de genes de k_1 a k_2 para serem atualizados no vetor *cromo-filho*. Para isso, conta-se o número l_1 de genes de p_2, das posições 1 a $k-1$, que possuem valores iguais aos armazenados no vetor s_1, copiando-se nas componentes de posição 1 a k_1-1 do *cromo-filho*, que são diferentes dos genes de s_1, os genes de p_2 de índices associados: *cromo-filho*$(j) = p_2(j)$, $j = 1,\ldots,k_1 - 1$, quando $p_2(j) \neq s_1(i)$, para todo $i = 1,\ldots,k_2 - k_1 + 1$. Para as l_1 posições não copiadas, copiam-se os l_1 primeiros valores de s_2 que também são distintos dos valores de s_1. Um contador armazena o número de posições de s_1, l_3, que obtiveram valores de s_2 iguais aos de s_1, até obter as l_1 primeiras posições de genes distintos. De forma análoga, se $k_2 \neq n$, então existem genes depois da subsequência de genes de k_1 a k_2 para serem atualizados no vetor *cromo-filho*. Então, conta-se o número l_2 de genes de p_2, das posições $k_2 + 1$ a n, que possuem valores iguais aos armazenados no vetor s_1, copiando-se nas componentes de posição k_2+1 a n do *cromo-filho*, que são diferentes dos genes de s_1, os genes de p_2 de índices associados: *cromo-filho*$(j) = p_2(j)$, $j = k_2 + 1, \ldots, n$, quando $p_2(j) \neq s_1(i)$, para todo $i = 1,\ldots,k_2-k_1+1$. Para as l_2 posições não copiadas, copiam-se os l_2 valores de s_2, investigados a partir das posições l_1+l_3+1 até k_2, que também são distintos dos valores de s_1. Essa estratégia da Recombinação PMX refere-se estritamente à formação dos primeiros n componentes do *cromo-filho* (sequência 1a-S): *cromofilho*(l), $l = 1, \ldots, n$. Porém, as sequências 2a-S e 3a-S também são formadas em *cromo-filho* (posições $n+1$ a $3n$), de acordo com o que foi determinado na formação de 1a-S. Isto quer dizer que se *cromo-filho*(l) é atualizado como um gene m de p_1, $p_1(m)$, para algum $m = 1, \ldots, n$, ou seja, *cromo-filho*$(l) = p_1(m)$, então também atualizam-se *cromo-filho*$(l + n) = p_1(m + n)$ e *cromo-filho*$(l + 2n) = p_1(m + 2n)$. Análogo, se a atualização de *cromo-filho* for a partir de p_2.

- *Passo 5 (Etapa 3(d) - Mutação)*: Utilizando-se o método MutacaoFilho, geram-se dois valores inteiros aleatórios t_1 e t_2, de 1 a n, que tenham uma distância de t% entre si. O cromossomo filho mutante tem os valores correspondentes à posição t_1, $t_1 + n$ e $t_1 + 2n$ permutados, respectivamente, com os genes da posição t_2, $t_2 + n$ e $t_2 + 2n$. Ao final, o método criarCarregamentoMutante é executado para construir um carregamento para o cromossomo mutante.

- *Passo 6 (Etapa 3(e) - Processo de Melhoria Local por SDH e cálculo da fitness)*: Geram-se w cromossomos vizinhos ao cromossomo mutante, por meio da estrutura de vizinhança *2-opt*. Para isso, realizam-se w vezes o processo de seleção aleatória de duas componentes da linha $p+1$ de A_{pop}, invertendo-se a subsequência $A_{pop}(p+1,i)$ a $A_{pop}(p+1, j)$, $i,j = 1,\ldots,n$, e copiando-se os demais genes que não pertencem à subsequência, (armazenando-os na linha l, $l = p+2,\ldots,p+w+1$, respectivamente). Utilizando-se o *Decodificador Carregamento*, é efetuado o cálculo das *fitness* associadas a cada linha correspondente ao cromossomo mutante e seus vizinhos (linhas $p + 1$ a $p + w + 1$). Utilizando o *Decodificador Carregamento*, calcula-se a *fitness* do descendente mutante e dos seus vizinhos.

- *Passo 7 (Etapa 3(f) - Teste de Comparação e Atualização da população)*: Comparam-se as *fitness* do cromossomo mutante e de seus vizinhos; aquele que possuir menor valor é rearranjado para a linha $p+1$ de A_{pop}, com fitness associada para a posição $p+1$. Uma segunda comparação entre *fitness* é realizada, desta vez, entre os valores apresentados pelos cromossomos das linhas p e $p+1$. Caso $f_{fitness}(p+1) < f_{fitness}(p)$, há uma atualização de A_{pop} (reordenando-a da linha 1 até $p + 1$) e de $f_{fitness}$, incluindo, portanto o cromossomo da posição $p + 1$ nas linhas 1 a p, e excluindo o cromossomo da posição p para a posição $p + 1$ (posteriormente sobre-escrito pelo mutante da próxima iteração), e os valores *fitness*, analogamente. Ao final desse processo, retorna-se ao Passo 2. Caso $f_{fitness}(p + 1) \geq f_{fitness}(p)$, a população não é evoluída, e retorna-se diretamente ao Passo 2.

5.6.2 Pseudocódigo do AGE-E

O pseudocódigo do AGE-E é apresentado na Figura 5.12, que mostra, de maneira unificada, todas as oito etapas da implementação, descritas anteriormente. A estrutura de implementação do AGE-E é idêntica à estrutura do AGE, apresentada na Figura 5.10, com exceção da parte final

do Ciclo Geracional que começa em *Melhoria SDH* e termina em *Atualização da População*, na Figura 5.12. O Teste de Comparação, apesar de permanecer o nome, também é um método que é modificado do AGE para o AGE-E, devido ao aumento de cromossomos candidatos a evoluir a população atual (considerando o cromossomo mutante descendente e os vizinhos gerados por SDH).

Figura 5.12 - Pseudo-código do AGE-E com detalhamento do Decodificador Carregamento.

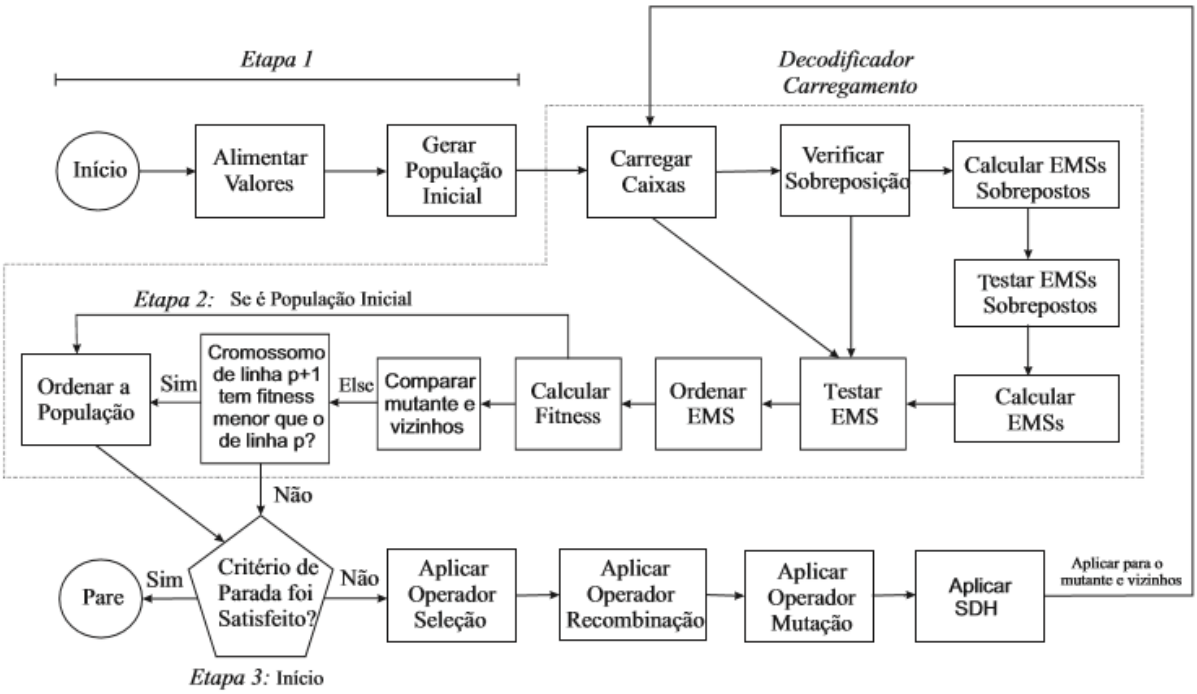
```
// Inicialização
TotalCromossomos = an;     // n = número de caixas
Definição dos parâmetros iniciais de controle

1a-S()  // Recebimento da ordem de caixas a serem carregadas
2a-S()  // Geração das heurísticas de carregamento, por caixa
3a-S()  // Determinação da orientação inicial das caixas

// Processo de Carregamento de Caixas - Método Decodificador
Para t = 1 até TotalCromossomos
Início
  Para k = 1 até n
  Início
    Para i = 1 até últimoCONTEINERutilizado
    Início
      Verificar se o contêiner i já está aberto
      Para b = 1 até últimoEMSutilizado
      Início
        Para c = 1 até 6        // Número total de rotações
        Início
          Se a caixa atual pôde ser carregada, então
          Início
            Atualizar informações da caixa carregada
            Vericar RMUS
            Realização do Processo DP
            Para d = 1 até totalEMS
              Início
                Teste de Volumes
                Teste 1 de [10]
                Teste 2 de [10]
                Eliminação de EMS inscritos [13]
                Eliminação de EMS de larguras ínfimas [13]
              Fim
            Atualização da lista de EMS
          Fim
        Fim
      Fim
    Fim
  Fim
  CalcularFitness
Fim

Verificação do Critério de Parada     //Início do Ciclo Geracional
Operador Seleção
Operador Recombinação
Operador Mutação
Melhoria SDH
Método Decodificador      //Aplicado ao mutante e seus vizinhos
Teste de Comparação
Atualização da População

// Finalização
```

Fonte: Adaptado de Resende e Gonçalves (2013) [12].

5.6.3 Visão global da implementação do AGE-E

O esquema que sintetiza a meta-heurística AGE-E é apresentado na Figura 5.13. A principal diferença, se comparado com o esquema do AGE da Figura 5.11, é a inserção da Heurística SDH para a geração dos cromossomos mutantes. Na sequência, o Decodificador Carregamento é aplicado tanto no cromossomo descendente mutante, quanto nos *w* vizinhos gerados. Com isso, também é introduzido o teste de comparação de *fitness* que verifica qual dentre estes, será direcionado para a posição $p + 1$ de A_{pop}.

Figura 5.13 - Estrutura da Implementação do AGE-E.



Fonte: Elaborado pelo próprio autor.

5.7 RESULTADOS COMPUTACIONAIS FINAIS

A implementação computacional do BRKGA foi realizada utilizando-se a linguagem C por meio da ferramenta de desenvolimento Open Source DevC++, compilador GCC 4.8.1. Os resultados computacionais foram obtidos utilizando um computador com CPU Intel Core i3-M380 @2.53 GHz, com o Sistema Operacional Microsoft Windows 7 Professional (32 bits). Estas são as mesmas especificações sobre o computador e softaware utilizados para a implementação do BRKGA, na primeira parte deste trabalho.

Testes computacionais prévios foram realizados com a primeira categoria dos problemas de grande porte, para a definição empírica do número w de vizinhos a serem gerados por SDH, em AGE-E. Desde o começo dos testes, foi possível verificar que o aumento do número de vizinhos ocasionava o crescimento exponencial do tempo computacional. O maior número que se pôde trabalhar de maneira funcional, foi para $w = 6$, o que provocou um aumento de tempo de 4 a 5 vezes para a obtenção das soluções, de AGE para AGE-E. Além disso, outros testes computacionais prévios foram realizados também para a Categoria 1, para a definição de β. O valor $\beta = 0,05$ foi fixado, após tentativas com os valores: 0,05, 0,1, 0,2, 0,3 e 0,4. Inicialmente, os resultados foram muito semelhantes, não sendo possível extrair muitas informações ao variar β. Porém, ao se comparar os dados com $\beta = 0,05$ e $\beta = 0,40$, os resultados empíricos foram, em média, de 5% a 10% piores para quando $\beta = 0,40$.

Nas tabelas 5.2 e 5.3 apresentam-se os resultados computacionais das meta-heurísticas AGE e AGE-E, com implementação para caixas com orientação única e orientação múltipla (seis orientações), respectivamente. As linhas das tabelas representam os exemplos testados, e as colunas indicam as categorias de problemas. A última linha apresenta as médias de cada categoria. Os valores para t_1 e t_2 mostram os tempos globais, em minutos, para a obtenção das melhores soluções, após 300 iterações do Ciclo Geracional (Critério de Parada). A disposição dos dados foi apresentada de forma que fosse possível comparar os resultados e tempos computacionais obtidos para AGE-E e AGE, nessa ordem.

Foi encontrado um padrão nos resultados das soluções das Categorias 1 a 5, devido à semelhança entre as dimensões das caixas, entre categorias diferentes, provenientes do gerador de exemplos de Pisinger [105] (disponibilizadas no Apêndice A). Em torno de 30 a 50% das caixas eram iguais entre as categorias, e as que eram distintas, pelo menos uma das dimensões eram iguais. Os valores decimais de muitas soluções, mesmo que em categorias diferentes, tiveram valores decimais iguais, apesar de nem sempre terem mesmo número inteiro de contêiners.

Os resultados demonstram que em todas as categorias, a *fitness* média final de contêiners diminui da meta-heurística AGE para AGE-E, com a inserção da Melhoria Local SDH. Os valores destacados demonstram inclusive a diminuição de um contêiner completo, o que, para casos reais, fariam economizar substancialmente os custos totais. Para a categoria 2, por exemplo, houve a diminuição total de dois contêiners, tanto no caso de orientações múltiplas, quanto para orientação única. Porém, os tempos computacionais para AGE-E aumentaram exponencialmente, tornando sua utilização uma escolha relevante de acordo com o que é mais importante: tempo computacional ou obtenção de melhores *fitness*.

Tabela 5.2 - Resultados Computacionais - orientação única

		Categorias							
Prs		1	2	3	4	5	6	7	8
1	AGE-E	15,2523	15,1692	15,1237	25,1643	11,0041	12,1200	10,1500	10,2178
	t_1	258	262	264	371	180	189	196	195
	AGE	15,3350	15,2038	15,3015	25,2690	11,2523	12,1200	10,1500	10,4772
	t_2	63	61	63	91	53	53	43	58
2	AGE-E	15,0840	**15,5403**	14,1237	28,1643	9,2523	10,1080	6,0840	7,0806
	t_1	282	264	255	394	178	173	185	187
	AGE	15,1012	**16,1329**	14,0882	28,1986	9,2523	10,1120	6,1443	7,0800
	t_2	66	67	61	95	51	48	43	53
3	AGE-E	15,0364	15,2255	13,0325	30,2070	7,1892	12,0010	7,0840	11,0111
	t_1	255	268	246	444	171	199	185	195
	AGE	15,0945	15,3368	13,0325	30,2070	7,2143	12,1440	7,0840	11,1651
	t_2	65	66	60	112	48	56	44	55
4	AGE-E	**11,2523**	12,1209	13,0882	30,2210	7,2304	10,0010	6,0169	**8,0111**
	t_1	219	212	238	402	182	171	184	198
	AGE	**12,0945**	12,0953	13,0563	30,0961	7,2304	10,0840	6,1380	**9,0905**
	t_2	54	53	56	102	49	48	43	53
5	AGE-E	13,0731	14,1440	14,1237	30,1233	7,2523	9,4810	8,0840	8,0111
	t_1	234	241	245	465	164	170	170	187
	AGE	13,1218	14,0953	14,0882	30,2286	7,2523	9,2000	8,0840	8,1250
	t_2	57	59	62	114	47	49	43	53
6	AGE-E	13,0840	14,1329	12,1237	28,1966	8,2523	12,0010	7,1443	7,0111
	t_1	233	237	195	389	168	202	190	155
	AGE	13,0945	14,1454	12,3003	28,1966	8,2616	12,0720	7,2271	7,0905
	t_2	57	59	54	94	47	57	44	52
7	AGE-E	16,0197	15,1440	16,0819	28,1487	11,2018	9,0010	9,0840	13,0845
	t_1	297	279	320	385	210	184	184	198
	AGE	16,0945	15,1209	16,6128	28,1487	11,2523	9,7140	9,0960	13,3306
	t_2	70	67	73	95	57	51	44	62
8	AGE-E	18,1331	**16,4561**	16,0828	30,1432	11,0570	10,0010	6,0169	11,1075
	t_1	308	280	281	425	190	198	179	228
	AGE	18,1331	**17,0717**	16,0828	30,1432	11,1400	10,2240	6,1172	11,1075
	t_2	68	69	66	106	52	56	44	58
9	AGE-E	12,1331	13,0636	**11,2649**	27,1716	7,1216	8,0010	6,0169	10,2535
	t_1	245	258	234	395	171	167	184	191
	AGE	12,1331	13,1964	**12,0828**	27,1716	7,1400	8,1800	6,2034	10,0602
	t_2	57	60	58	94	49	46	43	56
10	AGE-E	12,1378	11,1247	13,1420	27,2412	5,3238	9,0010	10,0840	9,1845
	t_1	215	193	236	350	162	179	195	192
	AGE	12,1179	11,1029	13,0850	27,2412	5,1542	9,0842	10,0840	9,0602
	t_2	53	53	55	90	46	47	47	55
Final	AGE-E	14,1206	14,2121	13,8187	28,4781	8,4885	10,1716	7,5765	9,4973
	t_1	254,6	249,4	251,4	402,0	177,6	183,2	185,2	192,6
	AGE	14,2320	14,3501	13,9730	28,4901	8,5150	10,2934	7,6328	9,6587
	t_2	61,0	61,4	60,8	99,3	49,9	51,1	43,8	55,5

Fonte: Elaborado pelo próprio autor.

Tabela 5.3 - Resultados Computacionais - orientações múltiplas

		Categorias							
Prs		1	2	3	4	5	6	7	8
1	AGE-E	13,0251	13,0059	13,3608	24,0059	10,0775	10,1200	9,0008	9,0079
	t_1	169	236	237	375	129	177	170	200
	AGE	13,4649	13,3763	13,4981	24,2406	10,3886	10,5390	9,0840	9,3650
	t_2	62	60	58	85	49	52	48	54
2	AGE-E	13,1785	**14,0059**	13,0184	27,0122	8,0752	9,0960	**5,1443**	6,0079
	t_1	182	264	236	369	131	164	172	190
	AGE	13,2387	**15,0273**	13,1423	27,1986	8,0752	9,1200	**6,0840**	6,0800
	t_2	64	62	59	89	49	44	46	52
3	AGE-E	12,2616	14,0299	11,0184	30,1069	6,0752	12,0010	6,0840	10,0543
	t_1	167	263	230	464	114	181	176	190
	AGE	12,1942	14,3178	11,1237	30,2070	6,0752	12,0800	6,0840	10,1089
	t_2	61	64	55	108	44	51	49	54
4	AGE-E	10,1012	10,0305	11,0184	30,2195	6,2616	8,2000	5,0840	8,0543
	t_1	148	215	208	426	131	175	178	192
	AGE	10,1256	10,1822	11,1423	30,2210	6,0752	8,2000	5,0840	8,0684
	t_2	52	51	54	98	45	44	47	50
5	AGE-E	11,0790	**11,0305**	12,0184	30,2286	7,0752	8,0480	7,0840	7,0006
	t_1	149	228	219	480	119	153	173	189
	AGE	11,1256	**12,1440**	12,1405	30,2286	7,0752	8,0800	7,0840	7,1250
	t_2	54	56	56	112	45	44	47	52
6	AGE-E	12,1012	13,0305	11,0184	28,1966	**7,5891**	10,0480	6,0443	6,0170
	t_1	152	240	224	429	119	191	176	178
	AGE	12,1256	13,1405	11,2078	28,1966	**8,0752**	10,2000	6,5817	6,0684
	t_2	57	57	52	89	46	53	47	48
7	AGE-E	14,0197	14,0305	15,0184	27,3008	10,0040	8,0480	8,0840	12,0543
	t_1	189	268	283	444	135	169	179	206
	AGE	14,2523	14,1405	15,2897	27,3008	10,0752	8,6090	8,0840	12,2704
	t_2	68	65	68	89	55	47	48	58
8	AGE-E	16,0172	15,0202	15,1216	30,1432	10,0040	9,0480	5,0840	10,0219
	t_1	193	270	271	426	130	194	186	199
	AGE	16,3920	15,1406	15,1400	30,1432	10,1397	9,2450	5,0840	10,2320
	t_2	68	64	62	99	51	53	49	54
9	AGE-E	**10,1731**	12,0583	10,0039	26,1716	6,0351	7,0800	5,2302	9,0219
	t_1	157	250	226	382	122	152	173	192
	AGE	**11,1731**	12,2189	10,1791	26,1716	6,1791	7,3250	5,2302	9,0919
	t_2	57	59	54	88	47	42	50	53
10	AGE-E	11,1705	10,0833	11,1791	26,2412	5,0351	8,0480	9,0840	8,0219
	t_1	149	208	213	371	122	168	179	191
	AGE	11,1731	10,1454	11,1791	26,2412	5,0779	8,0800	9,0840	8,2704
	t_2	53	49	54	84	46	44	51	54
Final	AGE-E	12,3127	12,6326	12,2775	27,9627	7,6232	8,9737	6,5923	8,5262
	t_1	165,5	244,2	234,7	416,6	125,2	172,4	176,2	192,7
	AGE	12,5270	12,9834	12,4043	28,0149	7,7237	9,1478	6,7484	8,6680
	t_2	59,6	58,7	57,2	94,1	47,7	47,4	48,2	52,9

Fonte: Elaborado pelo próprio autor.

Na Tabela 5.4 apresenta-se o Teste de Desempenho de AGE e AGE-E, comparando-os com os melhores algoritmos da literatura, listados na Tabela 5.1. A primeira coluna indica o número de orientações, 1-r para orientação única, e 6-r para múltiplas orientações de caixa, e a segunda coluna, os algoritmos testados. A última coluna indica as médias finais de resultados, por algoritmo, considerando-se todas as 8 categorias testadas. As demais colunas indicam as médias, por categoria, para cada algoritmo; logo, a análise comparativa é realizada por coluna. Em todos os casos, os resultados obtidos com o AGE e, principalmente, com AGE-E são muito próximos aos demais valores obtidos com os outros algoritmos. Em particular, para o caso 6-r, foram valores exatamente idênticos para a Categoria 6 (8,9), e resultados melhores para a Categoria 4 (28,9 para 27,9), ambos em destaque. A média final de contêiners, 13,0, é o menor valor obtido pela literatura, para o caso 1-r (BRKGA e GVND), sendo que para o AGE-E, essa média foi igual a 13,2. Para o caso 6-r, esses valores foram de 11,8 do BRKGA (único parâmetro de comparação) para 12,1 com AGE-E. Não foi possível estipular critérios comparativos para os tempos globais, uma vez que nenhum dos algoritmos comparados apresentam estes valores.

Tabela 5.4 - Teste de Desempenho para o AGE-E.

		Categorias								
	Algoritmos	1	2	3	4	5	6	7	8	Final
1-r	AGE-E	14,1	14,2	13,8	28,4	8,4	10,1	7,5	9,4	13,2
	AGE	14,2	14,3	13,9	28,4	8,5	10,2	7,6	9,6	13,3
	BRKGA	13,4	13,8	13,3	29,4	8,3	9,8	7,4	9,2	13,0
	TS3	13,4	13,8	13,3	29,4	8,4	9,9	7,5	9,3	13,1
	GVND	13,4	13,8	13,3	29,4	8,3	9,8	7,4	9,2	13,0
	TS[2]	13,4	-	-	29,4	8,3	9,8	7,4	9,2	-
	GLS	13,4	-	-	29,4	8,3	9,8	7,4	9,2	-
6-r	AGE-E	12,3	12,6	12,2	**27,9**	7,6	**8,9**	6,5	8,5	12,1
	AGE	12,5	12,9	12,4	28,0	7,7	9,1	6,7	8,6	12,2
	BRKGA	11,5	11,7	11,6	**28,9**	7,5	**8,9**	6,4	8,3	11,8

Fonte: Elaborado pelo próprio autor.

6 CONCLUSÕES

Este trabalho apresenta duas novas meta-heurísticas, o Algoritmo Genético Evolucionário (AGE), e sua especialização, o Algoritmo Genético Evolucionário Especializado (AGE-E), que resolvem um dos problemas menos estudados na literatura para o carregamento de contêiners, o Problema da Embalagem tridimensional (3D-SBSBPP, do inglês, *Three-dimensional Single Bin Size Bin Packing Problem*). Também conhecido na literatura apenas por 3D-BPP, estes problemas consideram caixas (itens) de tamanhos variados, com sortimento fortemente heterogêneo, que devem ser carregadas dentro de contêiners (objetos maiores) de iguais dimensões. Quanto maior o número de caixas ou quanto mais elevado o número de caixas por contêiner, maior é a dificuldade computacional em resolver esses problemas, pois o número de possibilidades de arranjos entre caixas, dentro dos contêiners, cresce de forma exponencial.

Os poucos algoritmos exatos conhecidos na literatura que resolvem o 3D-BPP quase sempre são aplicados a categorias de exemplos mais simples, com características triviais, como, por exemplo, considerando-se problemas de pequeno porte, com poucas caixas, ou com a utilização de poucos contêiners. Isto porque, além do crescimento exponencial proveniente da diversidade de combinações, a complexidade computacional destes métodos aplicados a problemas maiores também cresce de forma exponencial, por utilizarem algoritmos de busca em árvore que podem chegar a 2^n resoluções de subproblemas (n = número de caixas a serem carregadas). O trabalho com heurísticas e meta-heurísticas vieram por transformar esse contexto e permitir a obtenção de boas soluções para problemas de carregamento mais reais e, consequentemente, mais complexos, que possuem um contexto de caixas e contêiners de diveros tamanhos e dimensões, com especificidades particulares que cada situação real exige.

Nesse contexto, as meta-heurísticas AGE e AGE-E foram desenvolvidas para contribuir com o arcabouço de conhecimentos voltados ao estudo de problemas de carregamento de contêiners mais complexos, sob uma nova ótica de algoritmo genético, na qual a evolução da população só é realizada, um indivíduo por vez, caso seja verificado o melhoramento do conjunto de soluções atuais já determinadas para a população atual. Essa ideia foi inspirada pelo trabalho de Chu e Beasley [21], no qual os autores apresentaram um algoritmo genético que resolve Problemas de Atribuição Generalizada.

A estratégia de carregamento do AGE e AGE-E baseia-se no processo de geração de espaços maximais vazios, propostos por Lai e Chan [20], que estão mescladas com estratégias do BRKGA (do inglês, *Biased Radom-Key Genetic Algorithm*) de Resende e Gonçalves [12], o que inclui o cálculo da *fitness* associada a cada carregamento. O operador Recombinação foi

utilizado considerando-se o PMX (do inglês, *Partially-Matched Crossover*), criado em 1985, por Goldberg e Lingle [104]. Além disso, AGE e AGE-E apresentam uma nova sequência de etapas de Algoritmo Genético e incluem novos processos que foram criados por este estudo: a forma de decodificação dos cromossomos, a Restrição Matemática Universal de Sobreposição (RMUS), a Heurística de Formação da População Inicial (HFPI) e um novo teste de eliminação de espaços maximais vazios. Um procedimento de melhoria local SDH (do inglês, *Steepest Descent Heuristic*), baseado na estrutura de vizinhança *2-opt* também foi incluída ao final do AGE-E, com a finalidade de potencializar os resultados finais. Condições de estabilidade da carga também foram adicionalmente consideradas, dado que, o canto inferior esquerdo das caixas carregadas devem estar diretamente alocados sobre o chão do contêiner ou sobre outras caixas já carregadas.

Os experimentos computacionais tiveram como parâmetro de comparação os resultados obtidos pela escassa bibliografia que apresenta algoritmos que resolvem 3D-BPP complexos: TS3, GVND, TS^{2}PACK, GLS e BRKGA-1r, todos estes implementados apenas para única orientação de caixas. Também foi considerado como critério de desempenho para AGE e AGE-E o algoritmos BRKGA-6r, que, além de apresentarem os melhores resultados da bibliografia para o 3D-BPP, também realizaram a implementação para o caso de múltiplas orientações para as caixas.

Os resultados obtidos demonstraram a eficiência do AGE-E, que é comparável aos melhores algoritmos da literatura, e que apresenta as soluções em tempo computacional compatível. Foram testados 320 problemas, divididos em 4 variações do AGE-E contendo 80 problemas (oito categorias de 10 problemas): com orientações múltiplas ou única orientação de caixas, para AGE e AGE-E. Uma das categorias, a Categoria 4, que utiliza um maior número de contêiners, foi obtida uma solução ainda melhor, se comparado ao BRKGA (o único algoritmo da atualidade encontrado que resolve o 3D-BPP com as mesmas especificações que este trabalho): de 28,9 para 27,9, para o caso 6-r, e de 29,4 para 28,4, para o caso 1-r, ambos considerando a estrutura SDH. As demais categorias forneceram resultados muito próximos daqueles já encontrados pela literatura. A implementação do AGE também foi importante, pois, com uma pequena perda da qualidade dos resultados do AGE-E, o tempo computacional foi extremamente eficiente, havendo uma considerável diminuição. É importante ressaltar também que, pela primeira vez na literatura, os tempos globais foram apresentados para a resolução de problemas do tipo 3D-BPP. Além disso, não foram usadas técnicas avançadas de programação, como a inserção de multipopulações, que poderiam ainda ter contribuído para um aumento considerável do número inicial de cromossomos da população inicial, e do número de vizinhos.

O roteiro de cinco objetivos específicos estabelecido no planejamento inicial deste trabalho

foi cumprido, no qual considera: (1) a revisão bibliográfica sobre os PCC, e como o problema 3D-SBSBPP é descrito pela literatura correlata; (2) a revisão bibliográfica de meta-heurísticas que são eficientes e que resolvem o 3D-BPP; (3) a implementação da meta-heurística mais potente encontrada na etapa (2), visando à posterior comparação entre resultados, bem como obtenção dos primeiros resultados computacionais; (4) o desenvolvimento e implementação de uma nova meta-heurística fundamentada nos princípios teóricos de algoritmos genéticos; e (5) a realização dos testes computacionais finais para verificação da eficiência da nova meta-heurística proposta. Adicionalmente, uma especialização da meta-heurística inicial, AGE, foi também desenvolvida, culminando na criação do objeto-fim de estudo, a meta-heurística AGE-E.

As meta-heurísticas AGE e AGE-E possuem uma proposta diferente, mais rápida, e que vislumbra uma possibilidade de maior conexão com o contexto de problemas de grande porte de carregamentos do tipo 3D-BPP da vida real.

REFERÊNCIAS

[1] BRASIL. Ministério da Ciência, Tecnologia e Inovação. **Estratégia nacional de ciência, tecnologia e inovação 2012-2015**: balanço das atividades estruturantes 2011. Brasília: Ministério da Ciência: Tecnologia e Inovação, 2012.

[2] KITAMURA, B. L. A. **Heurística surrogate para o problema de carregamento de paletes do produtor**. 2009. 109 f. Dissertação (Mestrado em Matemática)– Departamento de Matemática Aplicada, Universidade Estadual Paulista – UNESP, São José do Rio Preto, 2009.

[3] UELZE, R. **Logística empresarial**: uma introdução à administração de transportes. São Paulo: Pioneira, 1974. 292 p.

[4] CECILIO, F. O. **Heurísticas para o problema de carregamento de carga dentro de contêiners**. 2003. 112 f. Dissertação (Mestrado em Engenharia de Produção)– Departamento de Engenharia de Produção, Universidade Federal de São Carlos – UFSCAR, São Carlos, 2003.

[5] UTIDA, M. A. **Heurísticas especializadas aplicadas ao problema de carregamento de contêiner**. 2012. 179 f. Tese (Doutorado em Engenharia Elétrica)– Faculdade de Engenharia, Universidade Estadual Paulista– UNESP, Ilha Solteira, 2012.

[6] WANG, Z.; LI, K. W.; LEVY, J. K. A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach. **European Journal of Operational Research**, Amsterdam, v. 191, n. 1, p. 86-99, 2008.

[7] GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting-stock problem. **Operations Research**, Baltimore, v. 9, n. 6, p. 849-859, 1961.

[8] GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problem - Part II. **Operations Research**, Baltimore, v. 11, n. 6, p. 863- 888, 1963.

[9] MARTELLO, S.; PISINGER, D. A.; VIGO, D. The three-dimensional finite bin packing problem. **Operations Research**, Baltimore, v. 48, n. 2, p. 256-267, 2000.

[10] GAREY, M. R.; JOHNSON. D. S. **Computers and intractabibility:** a guide to the theory of NP-completeness. São Francisco: W. H. Freeman, 1979.

[11] BEASLEY, J. E. A population heuristic for constrained two-dimensional non-guillotine cutting. **European Journal of Operational Research**, Amsterdam, v. 156, n. 3, p. 601- 627, 2004.

[12] RESENDE, M. G. C.; GONÇALVES, J. F. A Biased random-key genetic algorithm for a 2D and 3D bin packing problem. **International Journal of Production Economics**, Amsterdam, v. 145, p. 500-510, 2013.

[13] DYCKHOFF, H. A typology of cutting and packing problems. **European Journal of Operational Research**, Amsterdam, v. 44, n. 2, p. 145-159, 1990.

[14] WÄSCHER, G.; HAUBNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. **European Journal of Operational Research**, Amsterdam, v. 183, n. 3, p. 1109-1130, 2007.

[15] BORTFELDT, A.; WÄSCHER, G. Constraints in container loading problems – a state-of-the-art review. **European Journal of Operational Research**, Amsterdam, v. 229, n. 1, p. 1-20, 2013.

[16] PISINGER, D. Heuristics for the container loading problem. **European Journal of Operational Research**, Amsterdam, v. 141, n. 2, p. 382-392, 2002.

[17] GONÇALVES, J. F.; RESENDE, M. G. C. A parallel multi-population biased random-key genetic algorithm for a container loading problem. **Computers & Operations Research**, New York, v. 39, n. 2, p. 179-190, 2012.

[18] RESENDE, M. G. C.; GONÇALVES, J. F. Biased random-key genetic algorithms for combinatorial optimization. **Journal of Heuristics**, New York, v. 17, n. 5, p. 487-

525, 2011.

[19] RESENDE, M. G. C.; SILVA, R. M. A. Finding multiple roots of a box-constrained system of nonlinear equations with a biased random-key genetic algorithm. **Journal of Global Optimization**, New York, v. 60, n. 2, p. 289-306, 2013.

[20] LAI, K. K.; CHAN, J. W. M. Developing a simulated annealing algorithm for the cutting stock problem. **Computers and Industrial Engineering**, Amsterdam, v. 32, n. 1, p. 115-127, 1997.

[21] CHU, P. C.; BEASLEY, J. E. A genetic algorithm for the generalised assignment problem. **Computer & Operations Research**, New York, v. 24, n. 1, p.17-23, 1997.

[22] MARTÍNEZ, D. A. **Estudo dos problemas de corte e empacotamento**. 2014. 161 f. Tese (Doutorado em Engenharia Elétrica) – Faculdade de Engenharia, Universidade Estadual Paulista – UNESP, Ilha Solteira, 2014.

[23] LIU, N. C.; CHEN, L. C. A new algorithm for container loading. In: INTERNATIONAL COMPUTER SOFTWARE AND APPLICATION CONFERENCE ON THE IEEE, 5., 1981, New York. 3URFHHGLQJV New York: Torino, 1981. p. 292-299.

[24] GEHRING, H.; BORTFELDT, A. A genetic algorithm for solving the container loading problem. **International Transactions in Operational Research**, Oxford, v. 4, n. 5, p. 401-418, 1997.

[25] TERNO, J.; SCHEITHAUER, G.; SOMMERWEIB, U; RIEHME. J. An efficient approach for the multi-pallet loading problem. **European Journal of Operational Research**, Amsterdam, v. 123, n. 2, p. 372-381, 2000.

[26] CHAN, F. T. S.; BHAGWAT, R.; KUMAR, N.; TIWARI, M. K.; LAM, P. Development of a decision support system for air-cargo pallets loading problem: a case study. **Expert Systems with Applications**, Amsterdam, v. 31, n. 3, p. 472-485,

2006.

[27] EGEBLAD, J.; GARAVELLI, C.; LISI, L.; PISINGER, D. Heuristics for container loading of furniture. **European Journal of Operational Research**, Amsterdam, v. 200, n. 3, p. 881-892, 2010.

[28] LIU, W. Y.; LIN, C. C.; YU, C. S. On the three-dimensional container packing problem under home delivery service. **Asia-Pacific Journal of Operational Research**, Singapore, v. 28, n. 5, p. 1-20, 2011.

[29] SOMMERWEIB, U**.** Modeling of practical requirements of the distributers packing problem. In: INTERNATIONAL SCIENTIFIC SYMPOSIUM ON OPERATIONS RESEARCH, 3., 1995, Berlin. 3URFHHGLQJV Berlin: Springer, 1996. p. 427-432.

[30] BALAKIRSKY, S.; PROCTOR, F.; KRAMER, T.; KOLHE, P.; CHRISTENSEN, H. I. Using simulation to assess the effectiveness of pallet stacking methods. In: INTERNATIONAL CONFERENCE ON SIMULATION, MODELING AND PROGRAMMING FOR AUTONOMOUS ROBOTS, 2., 2010, Darmstadt. 3URFHHGLQJV« Darmstadt: Passau, 2010. p. 336-349.

[31] BORTFELDT, A.; GEHRING, H. Zur behandlung von restriktionen bei der stauraumoptimierung am beispiel eines genetischen algorithmus für das containerbeladeproblem. In: KOPFER, H.; BIERWIRTH, C. (Ed.). **Logistik management: intelligente I+K technologien.** Berlin: Springer, 1999b. p. 83-100.

[32] REN, J.; TIAN, Y.; SAWARAGI, T. A tree search method for the container loading problem with shipment priority. **European Journal of Operational Research**, Amsterdam, v. 214, n. 3, p. 526-535, 2011.

[33] MORABITO, R.; ARENALES, M. An and/or-graph approach to the container loading problem. **International Transactions in Operational Research**, Oxford, v. 1, n. 1, p. 59-73, 1994.

[34] JUNQUEIRA, L.; MORABITO, R.; YAMASHITA, D. S. Three-dimensional

container loading models with cargo stability and load bearing constraints. **Conputers & Operations Research**, New York, v. 39, n. 1, p. 74-85, 2012.

[35] CHIEN, C. F.; DENG, J. F. A container packing support system for determining and visualizing container packing patterns. **Decision Support Systems**, Amsterdam, v. 37, n. 1, p. 23-34, 2004.

[36] FUELLERER, G.; DOERNER, K.; HARTL, R. F.; IORI, M. Metaheuristics for vehicle routing problems with three-dimensional loading constraints. **European Journal of Operational Research**, Amsterdam, v. 201, n. 3, p. 751-759, 2010.

[37] IORI, M.; MARTELLO, S. Routing problems with loading constraints. **Top**, Heidelberg, v. 18, n. 1, p. 4-27, 2010.

[38] BISCHOFF, E. E.; JANETZ, F.; RATCLIFF, M. S. W. Loading pallets with non-identical items. **European Journal Operational Research**, Amsterdam, v. 84, n. 3, p. 681-692, 1995.

[39] PARREÑO, F.; ALVAREZ-VALDEZ, R.; TAMARIT, J. M.; OLIVEIRA, J. F. A maximal-space algorithm for the container loading problem. **Informs Journal on Computing**, Linthicum, v. 20, n. 3, p. 412-422, 2008.

[40] HE, K.; HUANG, W. An efficient placement heuristic for three-dimensional rectangular packing. **Computers & Operations Research**, New York, v. 38, n. 1, p. 227-233, 2011.

[41] FANSLAU, T.; BORTFELDT, A. A tree search algorithm for solving the container loading problem. **Informs Journal on Computing**, Linthicum, v. 22, n. 2, p. 222-235, 2010.

[42] FAINA, L. A global optimization algorithm for the three-dimensional packing problem. **European Journal of Operational Research**, Amsterdam, v. 126, n. 2, p. 340-354, 2000.

[43] PADBERG, M. Packing small boxes into a big box. **Mathematical Methods of Operational Research**, Heidelberg, v. 52, n. 1, p. 1-21, 2000.

[44] SCIOMACHEN, A.; TANFANI, E. A 3D-BPP approach for optimising stowage plans and terminal productivity. **European Journal of Operational Research**, Amsterdam, v. 183, n. 3, p. 1433-1446, 2007.

[45] BISCHOFF, E. E. Three-dimensional packing of items with limited load bearing strength. **European Journal of Operational Research**, Amsterdam, v. 168, n. 3, p. 952-966, 2006.

[46] CESCHIA, S.; SCHAERF, A. Local search for a multi-drop multi-container loading problem. **Journal of Heuristics**, New York, v. 19, n. 2, p. 275-294, 2013.

[47] ELEY, M. A bottleneck assignment approach to the multiple container loading problem. **OR Spectrum**, Heidelberg, v. 25, n. 1, p. 45-60, 2003.

[48] TSAI, R. D.; MALSTROM, E. M.; KUO, W. Three dimensional palletization of mixed box sizes. **IEE Transactions**, New York, v. 25, n. 4, p. 64-75, 1993.

[49] HODGSON, T. J. A combined approach to the pallet loading problem. **IIE Transactions**, New York, v. 14, n. 3, p. 175-182, 1982.

[50] MAKAREM, O. C.; HARATY, R. A. Smart container loading. **Jounal of Computational Methods in Sciences and Engineering**, Lansdale, v. 10, n. 2, p. 231-245, 2010.

[51] HIFI, M. Approximate algorithms for the container loading problem. **International Transactions in Operacional Research**, Oxford, v. 9, n. 6, p. 747-774, 2002.

[52] EGEBLAD, J.; PISINGER, D. Heuristic approaches for the two- and three-dimensional knapsack packing problem. **Computers & Operations Research**, New York, v. 36, n. 4, p. 1026-1049, 2009.

[53] HIFI, M. Exact algorithms for unconstrained three-dimensional cutting problems: a comparative study. **Computers & Operations Research**, New York, v. 31, n. 5, p. 657- 674, 2004.

[54] FEKETE, S. P.; SCHEPERS, J. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In: BURKARD, R.; WOEGINGER, G. (Ed.). **European symposium on algorithms.** Áustria: Springer, 1997. p. 144-156.

[55] GEORGE, J. A; ROBINSON, D. F. A heuristic for packing boxes into a container. **Computers and Operations Research**, New York, v. 7, n. 3, p. 147-156, 1980.

[56] BISCHOFF, E. E; MARRIOT, M. D. A comparative evaluation of heuristics for container loading. **European Journal of Operational Research**, Amsterdam, v. 44, n. 2, p. 267-276, 1990.

[57] HEMMINK, J. **Container loading with variable strategies in each layer**. Turku: University of Turku, 1994. 19 p.

[58] GEHRING, H.; MENSCHER, K.; MEYER, M. A computer-based heuristic for packing pooled shipment containers. **European Journal of Operational Reserach**, Amsterdam, v. 44, n. 2, p. 277-288, 1990.

[59] LOH, T. H.; NEE, A. Y. C. A packing algorithm for hexaedral boxes. In: CONFERENCE OF INDUSTRIAL AUTOMATION, 1., 1992, Singapore. 3URFHHGLQJV Singapore: [s. n.], 1992. p. 115-126.

[60] GILMORE, P. C.; GOMORY, R. E. A multistage cutting stock problems of two and more dimensions. **Operations Research**, Baltimore, v. 13, n. 1, p. 94-120, 1965.

[61] BISCHOFF, E. E.; RATCLIFF, M. S. W. Issues in the development of approaches to container loading. **Omega**, Elmsford, v. 23, n. 4, p. 377-390, 1995.

[62] BORTFELDT, A.; GEHRING, H. Applying tabu search to container loading problems. In: INTERNATIONAL SCIENTIFIC SYMPOSIUM ON OPERATIONS

RESEARCH, 4., 1997, Berlin. 3URFHHGLQJV Berlin: Springer, 1998. p. 533-538.

[63] ELEY, M. Solving container loading problems by block arrangement. **European Journal of Operational Research**, Amsterdam, v. 141, n. 2, p. 393-409, 2002.

[64] MACK, D.; BORTFELDT, A.; GEHRING, H. A parallel hybrid local search algorithm for the container loading problem. **International Transactions in Operational Research**, Oxford, v. 11, n. 5, p. 511-533, 2004.

[65] BORTFELDT, A.; GEHRING, H.; MACK, D. A parallel tabu search algorithm for solving the container loading problem. **Parallel Computing**, Amsterdam, v. 29, n. 5, p. 641- 662, 2003.

[66] GEHRING, H.; BORTFELDT, A. A parallel genetic algorithm for solving the container loading problem. **International Transactions in Operational Research**, Oxford, v. 9, n. 4, p. 497-511, 2002.

[67] BORTFELDT, A.; GEHRING, H. A hybrid genetic algorithm for the container loading problem. **European Journal of Operational Research**, Amsterdam, v. 131, n. 1, p. 143-161, 2001.

[68] MOURA, A.; OLIVEIRA, J. F. A grasp approach to the container-loading problem. **IEEE Intelligent Systems**, New York, v. 20, n. 4, p. 50-57, 2005.

[69] LIM, A.; RODRIGUES, B.; WANG, Y. A multi-faced buildup algorithm for three-dimensional packing problems. **Omega**, Elmsford, v. 31, n. 6, p. 471-481, 2003.

[70] FEKETE, S.; SCHEPERS, J. A combinatorial characterization of higher-dimensional orthogonal packing. **Mathematics of Operations Research**, New York, v. 29, n. 2, p. 353-368, 2004.

[71] JOHNSON, D. S. **Near-optimal bin packing algorithms**. 1993. 401 f. Tese (Doutorado em Filosofia) – Department of Mathematics, Massachusetts Institute of

Technology, Cambridge, 1993.

[72] FAROE, O.; PISINGER, D.; ZACHARIASEN, M. Guided local search for the three-dimensional bin-packing problem. **Informs Journal on Computing**, Linthicum, v. 15, n. 3, p. 267-283, 2003.

[73] LODI, A.; MARTELLO, S.; VIGO, D. Approximation algorithms for the oriented two-dimensional bin packing problem. **European Journal of Operational Research**, Amsterdam, v. 112, n. 1, p. 158-166, 1999.

[74] LODI, A.; MARTELLO, S.; VIGO, D. Heuristic algorithms for the three-dimensional bin packing problem. **European Journal of Operational Research**, Amsterdam, v. 141, n. 2, p. 410-420, 2002.

[75] LODI, A.; MARTELLO, S.; VIGO, D. TSpack: A unifted tabu search code for multi-dimensional bin packing problems. In: INTERNATIONAL SCIENTIFIC SYMPOSIUM ON OPERATIONS RESEARCH, 6., 2004, Berlin. 3URFHHGLQJV« Berlin: Springer, 2005. p. 203-213.

[76] CRAINIC, T. G.; PERBOLI, G.; TADEI, R. TS^{2}PACK: A two-level tabu search for the three-dimensional bin packing problem. **European Journal of Operational Research**, Amsterdam, v. 195, n. 3, p. 744-760, 2009.

[77] FEKETE, S.; SCHEPERS, J.; VANDERVEEN, J. C. An exact algorithm for higher-dimensional orthogonal packing. **Operations Research**, Baltimore, v. 55, n. 3, p.569-587, 2007.

[78] BOSCHETTI, M. A.; MINGOZZI, A. The two-dimensional finite bin packing problem. Part II: New lower and upper bounds. **4OR: A Quaterly Journal of Operations Research**, Baltimore, v. 1, n. 2, p. 135-147, 2003b.

[79] MONACI, M.; TOTH, P. A set-covering-based heuristic approach for bin-packing problems. **Informs Journal on Computing**, Linthicum, v. 18, n. 1, p. 71-85, 2006.

[80] MARTELLO, S.; VIGO, D. Exact solution of the two-dimensional finite bin packing problem. **Management Science**, Providence, v. 44, n. 3, p. 388-399, 1998.

[81] PARREÑO, F.; ALVAREZ-VALDES, R.; OLIVEIRA, J. F.; TAMARIT, J. M. A hybrid GRASP/VND algorithm for two and three-dimensional bin packing. **Annals of Operations Research**, Baltimore, v. 179, n. 1, p. 203-220, 2010.

[82] HOLLAND, J. H. **Adaptation in natural and artiftcial systems**. Ann Arbor: University of Michigan Press, Ann Arbor, MI, 1975.

[83] VENKATRAMAN, S.; YEN, G. A generic framework for constrained optimization using genetic algorithms. **IEEE Transactions on Evolutionary Computation**, New York, v. 9, n. 4, p. 224-235, 2005.

[84] MITCHELL, M. **An introduction to genetic algorithms**. Cambridge: MIT Press, 1996. 158 p.

[85] ROMERO, R.; RIDER, M. S. **Uma heurística eftciente para a otimização de problemas complexos em sistemas de energia elétrica**. Ilha Solteira: Universidade Estadual Paulista – UNESP, 2016. Notas de Aula.

[86] BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. **ORSA Journal on Computing**, Linthicum, v. 6, n. 2, p. 154-160, 1994.

[87] SPEARS W.; DEJONG, K. On the virtues of parameterized uniform crossover. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 4., 1991, San Diego. 3URFHHGLQJV« San Diego: University of California, 1991. p. 230-236.

[88] LIU, D.; TENG, H. An Improved BL-Algorithm for genetic algorithm of the orthogonal packing rectangles. **European Journal of Operational Research**, Amsterdam, v. 112, n. 2, p. 158-166, 1999.

[89] CHU, P. C.; BEASLEY, J. E. **A genetic algorithm for the set partitionig problem**. London: The Management School Imperial College, 1995. Notas de Aula.

[90] BURIOL, L. S.; RESENDE, M. G. C; RIBEIRO, C. C.; THORUP, M. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. **Networks**, New York, v. 46, n. 1, p. 36-56, 2005.

[91] GONÇALVES, J. F.; ALMEIDA, J. R. A hybrid genetic algorithm for assembly line balancing. **Journal of Heuristics**, New York, v. 8, n. 6, p. 629-642, 2002.

[92] ERICSSON, M.; RESENDE, M. G. C; PARDALOS, P. M. A genetic algorithm for the weight setting problem in OSPF routing. **Journal of Combinatorial Optimization**, Boston, v. 6, n. 3, p. 299-333, 2002.

[93] GONÇALVES, J. F.; RESENDE, M. G. C. An evolutionary algorithm for manufacturing cell formation. **Computers and Industrial Engineering**, Amsterdam, v. 47, n. 2, p.247-273, 2004.

[94] GONÇALVES, J. F.; MENDES, J. J. M.; RESENDE, M. G. C. A hybrid genetic algorithm for the job shop scheduling problem. **European Journal of Operational Research**, Amsterdam, v. 167, n. 1, p. 77-95, 2005.

[95] BURIOL, L. S.; RESENDE, C. C.; THORUP, M. Survivable IP Network design with OSPF routing. **Networks**, New York, v. 49, n. 1, p. 51-64, 2007.

[96] GONÇALVES, J. F. A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. **European Journal of Operational Research**, Amsterdam, v. 183, n. 3, p. 1212-1229, 2007.

[97] GONÇALVES, J. F.; MENDES, J. J. M.; RESENDE, M. G. C. A genetic algorithm for the resource constrained multi-project scheduling problem. **European Journal of Operational Research**, Amsterdam, v.189, n. 3, p. 1171-1190, 2009.

[98] FONTES, D. B. M. M.; GONÇALVES, J. F. A multi-population genetic algorithm for tree-shaped network design problems. In: INTERNATIONAL JOINT CONFERENCE ON COMPUTATIONAL INTELLIGENCE, 1., 2009, Madeira. 3URFHHGLQJV Madeira: Springer, 2009. p. 177-182.

[99] FESTA, P.; GONÇALVES, J. F.; RESENDE, M. G. C.; SILVA, R. M. A. Automatic tuning of GRASP with path-relinking heuristics with a biased random-key genetic algorithm. In: INTERNATIONAL CONFERENCE ON EXPERIMENTAL ALGORITHMS, 9., 2010, Berlin. 3URFHHGLQJV Berlin: Springer, 2010. p. 338-349.

[100] GONÇALVES, J. F.; RESENDE, M. G. C. A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. **Journal of Combinatorial Optimization**, Boston, v. 22, n. 2, p. 180-201, 2011a.

[101] GONÇALVES, J. F.; RESENDE, M. G. C. Biased random-key genetic algorithms for combinatorial optimization. **Journal of Heuristics**, New York, v. 17, n. 5, p. 487-525, 2011b.

[102] GONÇALVES, J. F.; SOUZA, P. S. A. A genetic algorithm for lot sizing and scheduling under capacity constraints and allowing backorders. **International Journal of Production Research**, London, v. 49, n. 9, p. 2683-2703, 2011.

[103] GONÇALVES, J. F.; RESENDE, M. G. C. **A biased random-key genetic algorithms for job-scheduling**. Florham Park: Technical Report, AT&T Labs, 2012.

[104] GOLDBERG, D. E.; LINGLE, R. Alleles, loci, and the traveling salesman problem. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 1., 1985, San Diego. 3URFHHGLQJV San Diego: University of California, 1985. p. 154-159.

[105] PISINGER, D. 3LVLQJHU¶V RSWLPL]DWLRQ FRGHV. [S.l.]: [s.n., 200-]. Available: <http://www.diku.dk/~pisinger/codes.html>. Access in: 2 Aug. 2017.

[106] ROMERO, R. **Meta-heurísticas em sistemas elétricos de potência:** introdução ao estudo e aplicações. Ilha Solteira: Universidade Estadual Paulista – UNESP, 2016. Minicurso.

APÊNDICE A: DADOS PARA OS TESTES COMPUTACIONAIS

Tabela A.1 - Classe 01 (100x100x100) - Problemas 1 e 2

PROB01-050-01								PROB01-050-02							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	38	85	72	26	66	90	25	1	41	92	82	26	84	43	67
2	11	84	96	27	26	78	88	2	17	83	67	27	83	3	74
3	36	10	34	28	37	10	21	3	45	99	88	28	88	84	77
4	12	76	71	29	84	17	69	4	27	84	88	29	44	69	82
5	64	73	56	30	47	96	99	5	34	85	74	30	49	97	98
6	26	77	97	31	19	72	74	6	30	69	88	31	17	74	79
7	30	23	17	32	4	77	73	7	26	80	74	32	33	89	95
8	46	27	47	33	51	97	51	8	10	89	70	33	38	70	75
9	47	69	67	34	38	74	95	9	68	23	93	34	88	70	73
10	29	71	98	35	23	42	26	10	22	78	84	35	92	98	15
11	88	69	64	36	72	86	64	11	77	7	93	36	88	80	8
12	47	95	99	37	6	86	67	12	21	22	43	37	26	84	81
13	20	95	93	38	86	25	88	13	79	49	77	38	14	90	67
14	23	8	32	39	5	74	88	14	39	87	98	39	64	59	92
15	67	73	55	40	92	90	10	15	25	67	75	40	74	50	55
16	69	46	95	41	32	82	70	16	20	72	100	41	44	96	85
17	47	80	85	42	24	80	93	17	11	93	93	42	79	23	80
18	10	87	91	43	24	95	74	18	21	74	97	43	24	91	97
19	10	100	96	44	66	77	86	19	14	8	42	44	5	89	81
20	45	100	84	45	47	82	77	20	88	89	43	45	27	73	88
21	15	99	82	46	7	76	76	21	11	74	74	46	82	46	84
22	93	93	38	47	38	70	99	22	5	80	99	47	91	92	84
23	82	43	95	48	92	80	39	23	44	74	92	48	29	75	97
24	98	38	97	49	46	93	92	24	10	66	84	49	73	26	74
25	69	90	72	50	13	71	91	25	85	37	66	50	12	9	42

Fonte: Elaborado pelo próprio autor.

Tabela A.2 - Classe 01 (100x100x100) - Problemas 3 e 4

PROB01-050-03								PROB01-050-04							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	44	100	79	26	11	82	81	1	44	95	89	26	17	83	73
2	51	100	65	27	49	22	23	2	26	70	92	27	11	92	71
3	1	82	88	28	84	59	67	3	92	86	75	28	74	74	48
4	39	49	17	29	30	92	84	4	4	12	14	29	17	41	32
5	89	79	24	30	94	59	67	5	35	42	35	30	6	77	74
6	35	97	92	31	16	75	85	6	39	66	83	31	14	89	90
7	25	93	67	32	9	100	81	7	82	71	77	32	87	51	88
8	25	88	84	33	98	73	25	8	87	100	22	33	1	87	71
9	50	98	73	34	43	92	79	9	94	77	34	34	71	84	4
10	12	98	81	35	70	33	93	10	5	82	67	35	49	89	72
11	41	67	78	36	43	84	73	11	28	68	97	36	47	66	67
12	46	90	90	37	45	83	82	12	95	39	79	37	15	81	96
13	80	67	64	38	34	99	82	13	47	38	43	38	5	74	84
14	8	67	68	39	98	73	60	14	81	28	74	39	4	80	66
15	94	88	14	40	12	72	93	15	4	85	76	40	8	81	90
16	17	79	71	41	97	86	44	16	13	75	99	41	17	100	69
17	83	72	5	42	73	15	68	17	44	73	98	42	46	8	18
18	31	85	81	43	23	39	34	18	94	37	87	43	21	70	72
19	19	78	87	44	42	84	87	19	23	91	88	44	96	30	80
20	20	89	98	45	6	88	86	20	33	78	94	45	34	67	97
21	83	95	15	46	74	54	53	21	4	70	69	46	26	86	75
22	39	66	74	47	39	90	71	22	26	100	83	47	75	83	27
23	8	97	78	48	50	94	89	23	83	86	15	48	56	97	53
24	8	91	70	49	23	68	92	24	7	94	80	49	37	21	14
25	36	87	72	50	10	83	77	25	5	76	67	50	87	52	53

Fonte: Elaborado pelo próprio autor.

Tabela A.3 - Classe 01 (100x100x100) - Problemas 5 e 6

PROB01-050-05								PROB01-050-06							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	47	68	86	26	81	97	7	1	49	75	84	26	87	35	79
2	31	93	76	27	24	66	70	2	85	92	19	27	34	86	68
3	16	84	99	28	71	78	23	3	25	3	8	28	90	33	99
4	51	83	54	29	88	86	64	4	79	83	92	29	88	6	89
5	33	66	70	30	11	79	74	5	96	56	80	30	95	92	50
6	44	94	75	31	13	90	96	6	48	38	36	31	11	92	66
7	18	83	99	32	77	77	75	7	16	95	68	32	41	100	99
8	2	31	31	33	98	27	81	8	16	88	77	33	14	69	90
9	1	92	91	34	50	75	98	9	97	31	82	34	28	67	85
10	85	77	53	35	7	72	85	10	50	69	57	35	16	68	99
11	14	68	82	36	4	70	97	11	85	65	50	36	8	87	68
12	45	45	39	37	36	80	74	12	17	89	93	37	4	8	3
13	24	93	98	38	100	73	63	13	48	92	82	38	78	93	43
14	45	98	67	39	21	79	75	14	11	79	73	39	97	20	83
15	90	6	82	40	5	66	86	15	33	91	77	40	1	75	83
16	10	70	70	41	30	78	85	16	6	3	43	41	92	92	35
17	9	87	72	42	20	48	11	17	25	100	80	42	45	71	76
18	74	83	6	43	90	50	97	18	90	70	22	43	80	84	44
19	73	68	10	44	62	70	82	19	32	81	91	44	45	79	79
20	95	66	2	45	13	22	5	20	8	20	23	45	66	73	46
21	3	91	95	46	17	85	82	21	98	60	85	46	67	8	78
22	12	86	93	47	38	75	90	22	47	45	14	47	74	21	71
23	79	39	73	48	45	73	86	23	4	98	70	48	17	68	79
24	5	84	66	49	2	78	80	24	2	75	88	49	16	66	97
25	21	66	73	50	7	95	98	25	40	90	80	50	6	13	4

Fonte: Elaborado pelo próprio autor.

Tabela A.4 - Classe 01 (100x100x100) - Problemas 7 e 8

PROB01-050-07								PROB01-050-08							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	50	83	93	26	41	77	93	1	2	78	91	26	47	90	85
2	42	80	66	27	72	72	27	2	46	79	84	27	9	20	13
3	73	1	98	28	87	15	89	3	40	93	74	28	38	80	90
4	67	97	49	29	24	69	91	4	10	93	69	29	11	81	69
5	34	88	91	30	18	94	85	5	33	81	67	30	22	72	84
6	88	91	23	31	91	93	92	6	86	81	74	31	8	17	22
7	94	100	92	32	19	76	73	7	11	85	100	32	46	88	95
8	29	87	91	33	46	84	77	8	43	9	4	33	29	98	86
9	4	86	74	34	89	65	52	9	30	100	100	34	33	97	69
10	30	46	37	35	73	77	97	10	21	20	48	35	68	32	91
11	80	92	23	36	15	43	33	11	22	93	82	36	88	73	71
12	72	12	83	37	100	99	56	12	53	83	71	37	95	45	68
13	25	92	77	38	18	67	81	13	76	79	3	38	41	77	95
14	70	49	78	39	79	50	56	14	58	90	84	39	20	85	88
15	24	89	83	40	50	83	79	15	14	74	78	40	91	92	46
16	88	73	24	41	3	82	69	16	79	46	97	41	80	83	91
17	92	79	37	42	17	78	99	17	6	92	85	42	42	86	74
18	25	81	84	43	19	79	81	18	38	68	67	43	89	38	81
19	89	60	69	44	32	74	73	19	41	94	82	44	16	80	78
20	20	78	91	45	95	3	69	20	68	37	87	45	50	67	68
21	81	75	23	46	97	82	61	21	42	97	70	46	39	95	81
22	76	54	90	47	39	83	74	22	80	18	98	47	40	76	78
23	20	98	91	48	40	87	71	23	80	87	3	48	100	84	87
24	50	47	22	49	91	88	10	24	64	52	94	49	43	76	85
25	6	80	74	50	89	2	84	25	25	4	46	50	3	95	95

Fonte: Elaborado pelo próprio autor.

Tabela A.5 - Classe 01 (100x100x100) - Problemas 9 e 10

PROB01-050-09								PROB01-050-10							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	5	86	100	26	96	69	41	1	7	93	98	26	67	25	91
2	91	67	43	27	20	66	87	2	55	66	74	27	32	86	85
3	48	46	24	28	23	84	80	3	7	95	73	28	9	89	81
4	69	20	85	29	45	69	83	4	39	96	90	29	32	93	96
5	33	87	77	30	27	74	96	5	34	80	88	30	89	87	50
6	79	38	88	31	92	73	13	6	18	92	79	31	85	10	88
7	8	98	93	32	97	31	69	7	89	75	7	32	1	38	1
8	6	75	83	33	9	78	96	8	20	87	84	33	42	45	32
9	7	80	92	34	87	88	28	9	68	39	83	34	40	80	88
10	13	75	100	35	41	77	69	10	4	94	86	35	50	73	83
11	6	93	67	36	76	74	54	11	43	94	86	36	100	71	8
12	41	87	97	37	16	74	81	12	15	91	86	37	34	72	83
13	78	79	43	38	11	86	98	13	4	78	87	38	32	8	27
14	17	90	89	39	37	72	97	14	33	94	95	39	3	93	70
15	3	95	84	40	68	100	20	15	43	12	26	40	39	74	92
16	46	76	68	41	28	74	100	16	42	84	96	41	38	88	68
17	23	71	94	42	16	81	97	17	37	84	67	42	69	9	72
18	48	79	74	43	18	93	68	18	9	66	92	43	16	77	91
19	46	31	34	44	76	5	72	19	50	84	85	44	37	49	42
20	45	45	32	45	29	94	78	20	8	79	66	45	7	48	2
21	39	71	73	46	27	82	76	21	99	69	85	46	87	20	72
22	4	91	73	47	40	68	93	22	41	78	82	47	84	11	97
23	76	35	86	48	33	66	91	23	16	99	83	48	5	84	96
24	77	50	70	49	8	98	68	24	79	96	20	49	22	86	85
25	70	93	56	50	1	84	70	25	90	43	82	50	50	72	81

Fonte: Elaborado pelo próprio autor.

Tabela A.6 - Classe 02 (100x100x100) - Problemas 1 e 2

PROB01-050-01								PROB01-050-02							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	94	25	72	26	66	90	25	1	91	47	82	26	84	43	67
2	11	84	96	27	81	33	88	2	82	38	67	27	83	3	74
3	36	10	34	28	37	10	21	3	45	99	88	28	88	84	77
4	72	26	71	29	84	17	69	4	97	39	88	29	99	24	82
5	64	73	56	30	97	11	99	5	84	5	74	30	49	97	98
6	81	22	97	31	74	37	74	6	30	69	88	31	67	34	79
7	30	23	17	32	84	27	73	7	66	45	74	32	98	34	95
8	46	27	47	33	51	97	51	8	100	4	70	33	93	35	75
9	67	9	67	34	38	74	95	9	68	23	93	34	88	70	73
10	69	1	98	35	23	42	26	10	82	23	84	35	92	98	15
11	88	69	64	36	72	86	64	11	77	7	23	36	88	80	8
12	47	95	99	37	6	86	67	12	21	22	43	37	81	34	81
13	20	95	93	38	86	25	88	13	79	49	77	38	14	90	67
14	23	8	32	39	5	74	88	14	69	47	98	39	64	59	92
15	67	73	55	40	92	90	10	15	90	32	75	40	74	50	55
16	69	46	95	41	92	12	70	16	20	72	100	41	94	21	85
17	67	40	85	42	24	80	93	17	11	93	93	42	79	23	80
18	70	32	91	43	99	50	74	18	86	34	97	43	74	21	97
19	90	25	96	44	66	77	86	19	14	8	42	44	90	39	81
20	45	100	84	45	47	82	77	20	88	89	43	45	92	28	88
21	90	34	82	46	92	41	76	21	86	19	74	46	82	46	84
22	93	93	38	47	93	10	99	22	5	80	99	47	91	92	84
23	82	43	95	48	92	80	39	23	79	44	92	48	99	30	97
24	98	38	97	49	96	3	92	24	10	66	84	49	73	26	74
25	69	90	72	50	88	21	91	25	85	37	66	50	12	9	42

Fonte: Elaborado pelo próprio autor.

Tabela A.7 - Classe 02 (100x100x100) - Problemas 3 e 4

PROB01-050-03								PROB01-050-04							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	89	20	79	26	91	42	81	1	74	40	89	26	97	38	73
2	51	100	65	27	49	22	23	2	66	25	92	27	76	42	71
3	81	7	88	28	84	59	67	3	92	86	75	28	74	74	48
4	39	49	17	29	90	32	84	4	4	12	14	29	17	41	32
5	89	79	24	30	94	59	67	5	35	42	35	30	86	12	74
6	95	37	92	31	96	30	85	6	84	21	83	31	89	29	90
7	90	18	67	32	9	100	81	7	82	71	77	32	87	51	88
8	100	28	84	33	98	73	25	8	87	100	22	33	81	12	71
9	70	38	73	34	73	22	79	9	94	77	34	34	71	84	4
10	12	98	81	35	70	33	93	10	5	82	67	35	84	4	72
11	41	67	78	36	43	84	73	11	83	28	97	36	82	6	67
12	76	30	90	37	70	3	82	12	95	39	79	37	70	21	96
13	80	67	64	38	99	4	82	13	47	38	43	38	100	19	84
14	93	37	68	39	98	73	60	14	81	28	74	39	94	15	66
15	94	88	14	40	92	32	93	15	74	15	76	40	68	1	90
16	87	39	71	41	97	86	44	16	78	10	99	41	87	40	69
17	83	72	5	42	73	15	68	17	79	33	98	42	46	8	18
18	66	35	81	43	23	39	34	18	94	37	87	43	71	10	72
19	99	38	87	44	82	34	87	19	68	21	88	44	96	30	80
20	90	44	98	45	86	43	86	20	93	3	94	45	79	7	97
21	83	95	15	46	74	54	53	21	4	70	69	46	86	1	75
22	99	31	74	47	69	5	71	22	91	20	83	47	75	83	27
23	98	42	78	48	95	39	89	23	83	86	15	48	56	97	53
24	78	41	70	49	98	48	92	24	92	44	80	49	37	21	14
25	36	87	72	50	100	48	77	25	95	26	67	50	87	52	53

Fonte: Elaborado pelo próprio autor.

Tabela A.8 - Classe 02 (100x100x100) - Problemas 5 e 6

PROB01-050-05								PROB01-050-06							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	72	13	86	26	81	97	7	1	69	35	84	26	87	35	79
2	76	18	76	27	79	11	70	2	85	92	19	27	69	31	68
3	16	84	99	28	71	78	23	3	25	3	8	28	90	33	99
4	51	83	54	29	88	86	64	4	79	83	92	29	88	3	89
5	88	11	70	30	91	29	74	5	96	56	80	30	95	92	50
6	74	4	75	31	83	25	96	6	48	38	36	31	76	22	66
7	78	13	99	32	77	77	75	7	66	35	68	32	81	10	99
8	2	31	31	33	98	27	81	8	86	8	77	33	69	39	90
9	96	17	91	34	70	25	98	9	97	31	82	34	68	2	85
10	85	77	53	35	97	22	85	10	50	69	57	35	76	43	99
11	74	38	82	36	74	50	97	11	85	65	50	36	88	47	68
12	45	45	39	37	71	40	74	12	87	4	93	37	4	8	3
13	84	33	98	38	70	73	63	13	73	27	82	38	78	93	43
14	70	18	67	39	96	44	75	14	81	9	73	39	97	20	83
15	90	6	82	40	80	21	86	15	93	46	77	40	91	40	83
16	70	30	70	41	90	48	85	16	3	6	43	41	92	92	35
17	99	32	72	42	20	48	11	17	25	100	80	42	80	41	76
18	74	83	6	43	90	50	97	18	90	70	22	43	80	84	44
19	73	68	10	44	62	70	82	19	77	36	91	44	90	19	79
20	95	66	2	45	13	22	5	20	8	20	23	45	66	73	46
21	88	26	95	46	77	5	82	21	98	60	85	46	67	8	78
22	82	6	93	47	68	50	90	22	47	45	14	47	74	21	71
23	79	39	73	48	45	73	86	23	99	38	70	48	82	18	79
24	70	44	66	49	77	43	80	24	72	45	88	49	66	16	97
25	76	21	73	50	7	95	98	25	70	15	80	50	6	13	4

Fonte: Elaborado pelo próprio autor.

Tabela A.9 - Classe 02 (100x100x100) - Problemas 7 e 8

PROB01-050-07								PROB01-050-08							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	90	8	93	26	71	32	93	1	87	28	91	26	77	30	85
2	72	5	66	27	72	72	27	2	81	49	84	27	9	20	13
3	73	1	98	28	87	15	89	3	80	48	74	28	38	80	90
4	67	97	49	29	24	69	91	4	10	93	69	29	71	21	69
5	99	48	91	30	88	14	85	5	33	81	67	30	92	32	84
6	88	91	23	31	91	93	92	6	86	81	74	31	8	17	22
7	94	100	92	32	94	16	73	7	66	30	100	32	96	23	95
8	74	32	91	33	46	84	77	8	43	9	4	33	69	18	86
9	99	46	74	34	89	65	52	9	30	100	100	34	88	7	69
10	30	46	37	35	43	77	97	10	21	20	48	35	68	32	91
11	80	92	23	36	15	43	33	11	22	93	82	36	88	73	71
12	72	12	83	37	100	99	56	12	53	83	71	37	95	45	68
13	75	22	77	38	18	67	81	13	76	79	3	38	91	27	95
14	70	49	78	39	79	50	56	14	58	90	84	39	100	25	88
15	24	89	83	40	80	8	79	15	89	29	78	40	91	92	46
16	88	73	24	41	83	17	69	16	79	46	97	41	80	83	91
17	22	79	37	42	97	33	99	17	76	27	85	42	92	26	74
18	70	41	84	43	19	79	81	18	98	43	67	43	89	38	81
19	89	60	69	44	82	14	73	19	86	49	82	44	96	10	78
20	100	28	91	45	95	3	69	20	68	37	87	45	100	17	68
21	81	75	23	46	97	82	61	21	77	32	70	46	71	15	81
22	76	54	90	47	79	43	74	22	80	18	98	47	40	76	78
23	95	38	91	48	90	27	71	23	80	87	3	48	100	84	87
24	50	47	22	49	91	88	10	24	64	52	94	49	68	11	85
25	86	10	74	50	89	2	84	25	25	4	46	50	78	40	95

Fonte: Elaborado pelo próprio autor.

Tabela A.10 - Classe 02 (100x100x100) - Problemas 9 e 10

PROB01-050-09								PROB01-050-10							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	85	1	100	26	96	69	41	1	82	23	98	26	67	25	91
2	91	67	43	27	100	41	87	2	100	36	74	27	67	11	85
3	48	46	24	28	23	84	80	3	7	95	73	28	99	19	81
4	69	20	85	29	85	29	83	4	94	31	90	29	77	38	96
5	98	37	77	30	27	74	96	5	69	5	88	30	89	87	50
6	79	38	88	31	92	73	13	6	18	92	79	31	85	10	88
7	8	98	93	32	97	31	69	7	89	75	7	32	1	38	14
8	96	35	83	33	74	33	96	8	95	12	84	33	2	45	32
9	67	25	92	34	87	88	28	9	68	39	83	34	85	10	88
10	98	45	100	35	41	77	69	10	99	19	86	35	95	23	83
11	86	28	67	36	76	74	54	11	78	39	86	36	100	71	8
12	76	27	97	37	16	74	81	12	15	91	86	37	34	72	83
13	78	79	43	38	91	41	98	13	79	3	87	38	32	8	27
14	82	30	89	39	67	2	97	14	33	94	95	39	68	28	70
15	93	20	84	40	68	100	20	15	43	12	26	40	79	14	92
16	71	16	68	41	88	34	100	16	42	84	96	41	78	43	68
17	73	26	94	42	16	81	97	17	37	84	67	42	69	9	72
18	78	44	84	43	88	8	68	18	94	46	92	43	86	27	91
19	46	31	34	44	76	5	72	19	95	14	85	44	37	49	42
20	45	45	32	45	94	34	78	20	73	4	66	45	7	48	2
21	74	16	73	46	97	17	76	21	99	69	85	46	87	20	72
22	4	91	73	47	90	38	93	22	86	43	82	47	84	11	97
23	76	35	86	48	33	66	91	23	96	34	83	48	100	4	96
24	77	50	70	49	93	33	68	24	79	96	20	49	82	6	85
25	70	93	56	50	1	84	70	25	90	43	82	50	90	17	81

Fonte: Elaborado pelo próprio autor.

Tabela A.11 - Classe 03 (100x100x100) - Problemas 1 e 2

PROB01-050-01								PROB01-050-02							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	94	85	27	26	66	90	25	1	91	92	42	26	84	43	67
2	11	84	96	27	81	78	48	2	82	83	37	27	83	3	74
3	36	10	34	28	37	10	21	3	45	99	88	28	88	84	77
4	72	76	26	29	84	17	69	4	97	84	23	29	99	69	47
5	64	73	56	30	97	96	24	5	84	85	14	30	49	97	98
6	81	77	2	31	74	72	39	6	30	69	88	31	67	74	29
7	30	23	17	32	84	77	38	7	66	80	9	32	98	89	35
8	46	27	47	33	51	97	51	8	100	89	5	33	93	70	10
9	67	69	37	34	38	74	95	9	68	23	93	34	88	70	73
10	69	71	18	35	23	42	26	10	82	78	29	35	92	98	15
11	88	69	64	36	72	86	64	11	77	7	93	36	88	80	8
12	47	95	99	37	6	86	67	12	21	22	43	37	81	84	1
13	20	95	93	38	86	25	88	13	79	49	77	38	14	90	67
14	23	8	32	39	5	74	88	14	69	87	33	39	64	59	92
15	67	73	55	40	92	90	10	15	90	67	50	40	74	50	55
16	69	46	95	41	92	82	35	16	20	72	100	41	94	96	15
17	67	80	40	42	24	80	93	17	11	93	93	42	79	23	80
18	70	87	36	43	99	95	29	18	86	74	2	43	74	91	32
19	90	100	36	44	66	77	86	19	14	8	42	44	90	89	26
20	45	100	84	45	47	82	77	20	88	89	43	45	92	73	38
21	90	99	12	46	92	76	6	21	86	74	39	46	82	46	84
22	93	93	38	47	93	70	19	22	5	80	99	47	91	92	84
23	82	43	95	48	92	80	39	23	79	74	22	48	99	75	32
24	98	38	97	49	96	93	17	24	10	66	84	49	73	26	74
25	69	90	72	50	88	71	1	25	85	37	66	50	12	9	42

Fonte: Elaborado pelo próprio autor.

Tabela A.12 - Classe 03 (100x100x100) - Problemas 3 e 4

PROB01-050-03								PROB01-050-04							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	89	100	4	26	91	82	16	1	74	95	19	26	97	83	13
2	51	100	65	27	49	22	23	2	66	70	2	27	76	92	11
3	81	82	43	28	84	59	67	3	92	86	75	28	74	74	48
4	39	49	17	29	90	92	14	4	4	12	14	29	17	41	32
5	89	79	24	30	94	59	67	5	35	42	35	30	86	77	49
6	95	97	27	31	96	75	20	6	84	66	13	31	89	89	10
7	90	93	2	32	9	100	81	7	82	71	77	32	87	51	88
8	100	88	14	33	98	73	25	8	87	100	22	33	81	87	41
9	70	98	18	34	73	92	9	9	94	77	34	34	71	84	4
10	12	98	81	35	70	33	93	10	5	82	67	35	87	89	42
11	41	67	78	36	43	84	73	11	83	68	47	36	82	66	7
12	76	90	10	37	70	83	37	12	95	39	79	37	70	81	26
13	80	67	64	38	99	99	42	13	47	38	43	38	100	74	24
14	93	67	33	39	98	73	60	14	81	28	74	39	94	80	21
15	94	88	14	40	92	72	13	15	74	85	31	40	68	81	40
16	87	79	6	41	97	86	44	16	88	75	34	41	87	100	24
17	83	72	5	42	73	15	68	17	79	73	13	42	46	8	18
18	66	85	21	43	23	39	34	18	94	37	87	43	71	70	37
19	99	78	47	44	82	84	17	19	68	91	3	44	96	30	80
20	90	89	13	45	86	88	26	20	93	78	34	45	79	67	17
21	83	95	15	46	74	54	53	21	4	70	69	46	86	86	10
22	99	66	9	47	69	90	6	22	91	100	43	47	75	83	27
23	98	97	43	48	95	94	24	23	83	86	15	48	56	97	53
24	78	91	10	49	98	68	32	24	92	94	25	49	37	21	14
25	36	87	72	50	100	83	32	25	95	76	12	50	87	52	53

Fonte: Elaborado pelo próprio autor.

Tabela A.13 - Classe 03 (100x100x100) - Problemas 5 e 6

PROB01-050-05								PROB01-050-06							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	72	68	31	26	81	97	7	1	69	75	44	26	87	35	79
2	76	93	11	27	79	66	50	2	85	92	19	27	69	86	38
3	16	84	99	28	71	78	23	3	25	3	8	28	90	33	99
4	51	83	54	29	88	86	64	4	79	83	92	29	88	6	89
5	88	66	45	30	91	79	24	5	96	56	80	30	95	92	50
6	74	94	50	31	83	90	1	6	48	38	36	31	76	92	41
7	78	83	39	32	77	77	75	7	66	95	33	32	81	100	19
8	2	31	31	33	98	27	81	8	86	88	37	33	69	69	20
9	96	92	1	34	70	75	48	9	97	31	82	34	68	67	45
10	85	77	53	35	97	72	30	10	50	69	57	35	76	68	19
11	74	68	22	36	84	70	32	11	85	65	50	36	88	87	8
12	45	45	39	37	71	80	14	12	87	89	3	37	4	8	3
13	84	93	33	38	100	73	63	13	73	92	22	38	78	93	43
14	70	98	32	39	96	79	50	14	81	79	33	39	97	20	83
15	90	6	82	40	5	80	86	16	93	91	12	40	91	75	43
16	70	70	15	41	90	78	5	16	6	3	43	41	92	92	35
17	99	87	22	42	20	48	11	17	25	100	80	42	80	71	1
18	74	83	6	43	90	50	97	18	90	70	22	43	80	84	44
19	73	68	10	44	62	70	82	19	77	81	16	44	90	79	34
20	95	66	2	45	13	22	5	20	8	20	23	45	66	73	46
21	88	91	20	46	77	85	12	21	98	60	85	46	67	8	78
22	82	86	28	47	68	75	45	22	47	45	14	47	74	21	71
23	79	39	73	48	45	73	86	23	99	98	10	48	82	68	4
24	70	84	41	49	77	78	45	24	72	75	8	49	66	66	27
25	76	66	8	50	7	95	98	25	70	90	5	50	6	13	4

Fonte: Elaborado pelo próprio autor.

Tabela A.14 - Classe 03 (100x100x100) - Problemas 7 e 8

PROB01-050-07								PROB01-050-08							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	90	83	8	26	71	77	48	1	87	78	21	26	77	90	45
2	72	80	26	27	72	72	27	2	81	79	34	27	9	20	13
3	73	1	98	28	87	15	89	3	80	93	19	28	38	80	90
4	67	97	49	29	24	69	91	4	10	93	69	29	71	81	4
5	99	88	16	30	88	94	25	5	33	81	67	30	92	72	49
6	88	91	23	31	91	93	92	6	86	81	74	31	8	17	22
7	94	100	92	32	94	76	13	7	66	85	20	32	96	88	10
8	74	87	46	33	46	84	77	8	43	9	4	33	69	98	1
9	99	86	34	34	89	65	52	9	30	100	100	34	88	97	34
10	30	46	37	35	73	77	97	10	21	20	48	35	68	32	91
11	80	92	23	36	15	43	33	11	22	93	82	36	88	73	71
12	72	12	83	37	100	99	56	12	53	83	71	37	95	45	68
13	75	92	12	38	18	67	81	13	76	79	3	38	91	77	10
14	70	49	78	39	79	50	56	14	58	90	84	39	100	85	33
15	24	89	83	40	80	83	19	15	89	74	43	40	91	92	46
16	88	73	24	41	83	82	14	16	79	46	97	41	80	83	91
17	92	79	37	42	97	78	44	17	76	92	45	42	92	86	34
18	70	81	39	43	19	79	81	18	98	68	7	43	89	38	81
19	89	60	69	44	82	74	73	19	86	94	27	44	96	80	13
20	100	78	41	45	95	3	69	20	68	37	87	45	100	67	23
21	81	75	23	46	97	82	61	21	77	97	50	46	71	95	16
22	76	54	90	47	79	83	34	22	80	18	98	47	40	76	78
23	95	98	31	48	90	87	46	23	80	87	3	48	100	84	87
24	50	47	22	49	91	88	10	24	64	52	94	49	68	76	40
25	86	80	49	50	89	2	84	25	25	4	46	50	78	95	35

Fonte: Elaborado pelo próprio autor.

Tabela A.15 - Classe 03 (100x100x100) - Problemas 9 e 10

PROB01-050-09								PROB01-050-10							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	85	86	35	26	96	69	41	1	82	93	48	26	67	25	91
2	91	67	43	27	100	66	2	2	100	66	49	27	67	86	40
3	48	46	24	28	23	84	80	3	7	95	73	28	99	89	1
4	69	20	85	29	85	69	23	4	94	96	35	29	77	93	41
5	98	87	37	30	27	74	96	5	69	80	48	30	89	87	50
6	79	38	88	31	92	73	13	6	18	92	79	31	85	10	88
7	8	98	93	32	97	31	69	7	89	75	7	32	1	38	1
8	96	75	13	33	74	78	16	8	95	87	19	33	42	45	32
9	67	80	17	34	87	88	28	9	68	39	83	34	85	80	23
10	98	75	10	35	41	77	69	10	99	94	21	35	95	73	23
11	86	93	22	36	76	74	54	11	78	94	46	36	100	71	8
12	76	87	47	37	16	74	81	12	15	91	86	37	34	72	83
13	78	79	43	38	91	86	43	13	79	78	32	38	32	8	27
14	82	90	34	39	67	72	12	14	33	94	95	39	68	93	40
15	93	95	9	40	68	100	20	15	43	12	26	40	79	74	47
16	71	76	33	41	88	74	25	16	42	84	96	41	78	88	3
17	73	71	4	42	16	81	97	17	37	84	67	42	69	9	72
18	78	79	24	43	88	93	3	18	94	66	42	43	86	77	6
19	46	31	34	44	76	5	72	19	95	84	40	44	37	49	42
20	45	45	32	45	94	94	13	20	73	79	1	45	7	48	2
21	74	71	28	46	97	82	16	21	99	69	85	46	87	20	72
22	4	91	73	47	90	68	23	22	86	78	2	47	84	11	97
23	76	35	86	48	33	66	91	23	96	99	48	48	100	84	26
24	77	50	70	49	93	98	23	24	79	96	20	49	82	86	5
25	70	93	56	50	1	84	70	25	90	43	82	50	90	72	16

Fonte: Elaborado pelo próprio autor.

Tabela A.16 - Classe 04 (100x100x100) - Problemas 1 e 2

PROB01-050-01								PROB01-050-02							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	84	65	70	26	66	90	25	1	100	97	80	26	84	43	67
2	11	84	96	27	65	50	95	2	59	78	75	27	83	3	74
3	36	10	34	28	37	10	21	3	45	99	88	28	88	84	77
4	68	61	58	29	84	17	69	4	96	86	70	29	91	68	78
5	64	73	56	30	56	88	96	5	66	59	51	30	49	97	98
6	96	78	74	31	94	77	50	6	30	69	88	31	59	76	99
7	30	23	17	32	63	73	99	7	100	69	59	32	54	100	87
8	46	27	47	33	51	97	51	8	65	98	55	33	86	75	66
9	99	89	70	34	38	74	95	9	68	23	93	34	88	70	73
10	73	81	91	35	23	42	26	10	89	99	95	35	92	98	15
11	88	69	64	36	72	86	64	11	77	7	93	36	88	80	8
12	47	95	99	37	6	86	67	12	21	22	43	37	63	100	56
13	20	95	93	38	86	25	88	13	79	49	77	38	14	90	67
14	23	8	32	39	5	74	88	14	71	54	85	39	64	59	92
15	67	73	55	40	92	90	10	15	84	94	59	40	74	50	55
16	69	46	95	41	71	50	65	16	20	72	100	41	79	99	90
17	57	99	55	42	24	80	93	17	11	93	93	42	79	23	80
18	52	79	70	43	59	97	92	18	74	65	65	43	66	60	62
19	59	85	79	44	66	77	86	19	14	8	42	44	71	94	91
20	45	100	84	45	47	82	77	20	88	89	43	45	92	69	58
21	85	99	78	46	50	53	62	21	72	51	85	46	82	46	84
22	93	93	38	47	100	68	81	22	5	80	99	47	91	92	84
23	82	43	95	48	92	80	39	23	55	51	77	48	59	91	74
24	98	38	97	49	83	55	99	24	10	66	84	49	73	26	74
25	69	90	72	50	52	72	66	25	85	37	66	50	12	9	42

Fonte: Elaborado pelo próprio autor.

Tabela A.17 - Classe 04 (100x100x100) - Problemas 3 e 4

PROB01-050-03								PROB01-050-04							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	66	79	64	26	100	75	73	1	57	86	74	26	96	55	51
2	51	100	65	27	49	22	23	2	93	95	80	27	59	65	68
3	100	94	70	28	84	59	67	3	92	86	75	28	74	74	48
4	39	49	17	29	64	65	91	4	4	12	14	29	17	41	32
5	89	79	24	30	94	59	67	5	35	42	35	30	100	69	65
6	86	94	92	31	76	74	98	6	55	50	88	31	92	98	96
7	53	96	56	32	9	100	81	7	82	71	77	32	87	51	88
8	89	57	94	33	98	73	25	8	87	100	22	33	79	58	71
9	63	53	62	34	63	84	94	9	94	77	34	34	71	84	4
10	12	98	81	35	70	33	93	10	5	82	67	35	66	62	54
11	41	67	78	36	43	84	73	11	71	92	86	36	86	66	63
12	88	84	59	37	66	90	61	12	95	39	79	37	93	79	92
13	80	67	64	38	51	93	73	13	47	38	43	38	89	58	55
14	65	51	97	39	98	73	60	14	81	28	74	39	80	86	52
15	94	88	14	40	84	79	85	15	94	60	92	40	93	58	65
16	78	85	77	41	97	86	44	16	51	84	84	41	71	68	65
17	83	72	5	42	73	15	68	17	63	78	91	42	46	8	18
18	95	78	86	43	23	39	34	18	94	37	87	43	82	62	52
19	52	77	59	44	51	61	71	19	99	86	61	44	96	30	80
20	71	82	69	45	99	88	82	20	90	64	83	45	54	55	81
21	83	95	15	46	74	54	53	21	4	70	69	46	86	68	62
22	76	55	67	47	81	64	86	22	70	55	66	47	75	83	27
23	62	68	57	48	71	82	63	23	83	86	15	48	56	97	53
24	91	58	72	49	64	98	70	24	91	62	72	49	37	21	14
25	36	87	72	50	58	93	74	25	73	97	51	50	87	52	53

Fonte: Elaborado pelo próprio autor.

Tabela A.18 - Classe 04 (100x100x100) - Problemas 5 e 6

PROB01-050-05								PROB01-050-06							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	74	68	58	26	81	97	7	1	90	100	94	26	87	35	79
2	85	66	70	27	66	88	68	2	85	92	19	27	98	84	67
3	16	84	99	28	71	78	23	3	25	3	8	28	90	33	99
4	51	83	54	29	88	86	64	4	79	83	92	29	88	6	89
5	69	70	85	30	81	55	64	5	96	56	80	30	95	92	50
6	76	84	85	31	58	96	95	6	48	38	36	31	74	95	93
7	61	98	74	32	77	77	75	7	64	73	95	32	94	53	63
8	2	31	31	33	98	27	81	8	83	62	82	33	72	92	51
9	53	68	79	34	63	62	86	9	97	31	82	34	88	77	82
10	85	77	53	35	68	67	68	10	50	69	57	35	71	98	83
11	90	65	94	36	74	93	72	11	85	65	50	36	87	95	54
12	45	45	39	37	70	69	71	12	67	58	66	37	4	8	3
13	86	72	80	38	100	73	63	13	89	74	89	38	78	93	43
14	52	97	97	39	63	74	96	14	71	95	59	39	97	20	83
15	90	6	82	40	52	61	95	15	53	52	74	40	61	91	75
16	76	82	66	41	80	65	91	16	6	3	43	41	92	92	35
17	100	63	78	42	20	48	11	17	25	100	80	42	57	95	96
18	74	83	6	43	90	50	97	18	90	70	22	43	80	84	44
19	73	68	10	44	62	70	82	19	92	52	66	44	67	87	61
20	95	66	2	45	13	22	5	20	8	20	23	45	66	73	46
21	60	57	78	46	73	55	95	21	98	60	85	46	67	8	78
22	63	80	66	47	88	60	65	22	47	45	14	47	74	21	71
23	79	39	73	48	45	73	86	23	59	70	70	48	78	52	57
24	90	91	96	49	70	90	67	24	65	70	70	49	68	61	52
25	57	66	77	50	7	95	98	25	67	85	53	50	6	13	4

Fonte: Elaborado pelo próprio autor.

Tabela A.19 - Classe 04 (100x100x100) - Problemas 7 e 8

PROB01-050-07								PROB01-050-08							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	82	82	52	26	85	98	86	1	98	89	88	26	81	52	64
2	93	83	75	27	72	72	27	2	84	78	64	27	9	20	13
3	73	1	98	28	87	15	89	3	59	78	73	28	38	80	90
4	67	97	49	29	24	69	91	4	10	93	69	29	60	54	100
5	71	94	74	30	68	51	86	5	33	81	67	30	99	61	84
6	88	91	23	31	91	93	92	6	86	81	74	31	8	17	22
7	94	100	92	32	84	79	76	7	97	75	62	32	50	55	64
8	82	72	70	33	46	84	77	8	43	9	4	33	90	100	56
9	68	83	96	34	89	65	52	9	30	100	100	34	63	80	74
10	30	46	37	35	73	77	97	10	21	20	48	35	68	32	91
11	80	92	23	36	15	43	33	11	22	93	82	36	88	73	71
12	72	12	83	37	100	99	56	12	53	83	71	37	95	45	68
13	67	77	71	38	18	67	81	13	76	79	3	38	85	69	60
14	70	49	78	39	79	50	56	14	58	90	84	39	61	88	99
15	24	89	83	40	96	69	54	15	88	69	56	40	91	92	46
16	88	73	24	41	71	85	66	16	79	46	97	41	80	83	91
17	92	79	37	42	68	87	72	17	55	67	63	42	54	80	73
18	53	76	93	43	19	79	81	18	100	62	62	43	89	38	81
19	89	60	69	44	98	54	67	19	85	95	97	44	52	69	97
20	94	59	73	45	95	3	69	20	68	37	87	45	54	52	51
21	81	75	23	46	97	82	61	21	73	90	98	46	58	70	95
22	76	54	90	47	69	82	95	22	80	18	98	47	40	76	78
23	91	61	50	48	64	94	97	23	80	87	3	48	100	84	87
24	50	47	22	49	91	88	10	24	64	52	94	49	79	53	100
25	51	54	54	50	89	2	84	25	25	4	46	50	99	94	69

Fonte: Elaborado pelo próprio autor.

Tabela A.20 - Classe 04 (100x100x100) - Problemas 9 e 10

PROB01-050-09								PROB01-050-10							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	64	71	97	26	96	69	41	1	80	52	82	26	67	25	91
2	91	67	43	27	93	100	92	2	67	95	69	27	99	96	91
3	48	46	24	28	23	84	80	3	7	95	73	28	75	67	93
4	69	20	85	29	59	51	87	4	62	80	88	29	84	100	73
5	99	92	63	30	27	74	96	5	75	78	58	30	89	87	50
6	79	38	88	31	92	73	13	6	18	92	79	31	85	10	88
7	8	98	93	32	97	31	69	7	89	75	7	32	1	38	1
8	53	92	96	33	99	79	72	8	76	77	58	33	42	45	32
9	83	98	62	34	87	88	28	9	68	39	83	34	63	58	66
10	71	98	66	35	41	77	69	10	62	90	70	35	81	93	90
11	67	87	98	36	76	74	54	11	87	61	54	36	100	71	8
12	72	82	74	37	16	74	81	12	15	91	86	37	34	72	83
13	78	79	43	38	71	84	94	13	50	59	96	38	32	8	27
14	52	87	96	39	95	51	92	14	33	94	95	39	77	64	84
15	81	91	59	40	68	100	20	15	43	12	26	40	73	56	70
16	72	75	95	41	89	80	66	16	42	84	96	41	72	78	66
17	66	52	50	42	16	81	97	17	37	84	67	42	69	9	72
18	70	75	58	43	96	79	51	18	92	61	78	43	78	68	72
19	46	31	34	44	76	5	72	19	78	61	51	44	37	49	42
20	45	45	32	45	61	96	100	20	99	55	89	45	7	48	2
21	61	92	78	46	96	83	52	21	99	69	85	46	87	20	72
22	4	91	73	47	50	78	74	22	83	54	88	47	84	11	97
23	76	35	86	48	33	66	91	23	88	63	63	48	97	90	91
24	77	50	70	49	57	74	86	24	79	96	20	49	85	96	71
25	70	93	56	50	1	84	70	25	90	43	82	50	54	64	77

Fonte: Elaborado pelo próprio autor.

Tabela A.21 - Classe 05 (100x100x100) - Problemas 1 e 2

PROB01-050-01								PROB01-050-02							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	39	25	27	26	66	90	25	1	41	47	42	26	84	43	67
2	11	84	96	27	26	33	48	2	17	38	37	27	83	3	74
3	36	10	34	28	37	10	21	3	45	99	88	28	88	84	77
4	12	26	26	29	84	17	69	4	27	39	23	29	44	24	47
5	64	73	56	30	47	11	24	5	34	5	14	30	49	97	98
6	26	22	2	31	19	37	39	6	30	69	88	31	17	34	29
7	30	23	17	32	4	27	38	7	26	45	9	32	33	34	35
8	46	27	47	33	51	97	51	8	10	4	5	33	38	35	10
9	47	9	37	34	38	74	95	9	68	23	93	34	88	70	73
10	29	1	18	35	23	42	26	10	22	23	29	35	92	98	15
11	88	69	64	36	72	86	64	11	77	7	93	36	88	80	8
12	47	95	99	37	6	86	67	12	21	22	43	37	26	34	1
13	20	95	93	38	86	25	88	13	79	49	77	38	14	90	67
14	23	8	32	39	5	74	88	14	39	47	33	39	64	59	92
15	67	73	55	40	92	90	10	15	25	32	50	40	74	50	55
16	69	46	95	41	32	12	35	16	20	72	100	41	44	21	15
17	47	40	40	42	24	80	93	17	11	93	93	42	79	23	80
18	10	32	36	43	24	50	29	18	21	34	2	43	24	21	32
19	10	25	36	44	66	77	86	19	14	8	42	44	5	39	26
20	45	100	84	45	47	82	77	20	88	89	43	45	27	28	38
21	15	34	12	46	7	41	6	21	11	19	39	46	82	46	84
22	93	93	38	47	38	10	19	22	5	80	99	47	91	92	84
23	82	43	95	48	92	80	39	23	44	44	22	48	29	30	32
24	98	38	97	49	46	3	17	24	10	66	84	49	73	26	74
25	69	90	72	50	13	21	1	25	85	37	66	50	12	9	42

Fonte: Elaborado pelo próprio autor.

Tabela A.22 - Classe 05 (100x100x100) - Problemas 3 e 4

PROB01-050-03								PROB01-050-04							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	44	20	4	26	11	42	16	1	44	40	19	26	17	38	13
2	51	100	6	27	49	22	23	2	26	25	2	27	11	42	11
3	51	7	43	28	84	59	67	3	92	86	75	28	74	74	48
4	39	49	17	29	30	32	14	4	4	12	14	29	17	41	32
5	89	79	24	30	94	59	67	5	35	42	35	30	6	12	49
6	35	37	27	31	16	30	20	6	39	21	13	31	14	29	10
7	25	18	2	32	9	100	81	7	82	71	77	32	87	51	88
8	25	28	14	33	98	73	25	8	87	100	22	33	1	12	41
9	50	38	18	34	43	22	9	9	94	77	34	34	71	84	4
10	12	98	81	35	70	33	93	10	5	82	67	35	49	4	42
11	41	67	78	36	43	84	73	11	28	28	47	36	47	6	7
12	46	30	10	37	45	3	37	12	95	39	79	37	15	21	26
13	80	67	64	38	34	4	2	13	47	38	43	38	5	19	24
14	8	37	33	39	98	73	60	14	81	28	74	39	4	15	21
15	94	88	14	40	12	32	13	15	4	15	31	40	8	1	40
16	17	39	6	41	97	86	44	16	13	10	34	41	17	40	24
17	83	72	5	42	73	15	68	17	44	33	13	42	46	8	18
18	31	35	21	43	23	39	34	18	94	37	87	43	21	10	37
19	19	38	47	44	42	34	17	19	23	21	3	44	96	30	80
20	20	44	13	45	6	43	26	20	33	3	34	45	34	7	17
21	83	95	15	46	74	54	53	21	4	70	69	46	26	1	10
22	39	31	9	47	39	5	6	22	26	20	43	47	75	83	27
23	8	42	43	48	50	39	24	23	83	86	15	48	56	97	53
24	8	41	10	49	23	48	32	24	7	44	25	49	37	21	14
25	36	87	72	50	10	48	32	25	5	26	12	50	87	52	53

Fonte: Elaborado pelo próprio autor.

Tabela A.23 - Classe 05 (100x100x100) - Problemas 5 e 6

PROB01-050-05								PROB01-050-06							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	47	3	31	26	81	97	7	1	49	35	44	26	87	35	79
2	31	18	11	27	24	11	50	2	85	92	19	27	34	31	38
3	16	84	99	28	71	78	23	3	25	3	8	28	90	33	99
4	51	83	54	29	88	86	64	4	79	83	92	29	88	6	89
5	33	11	45	30	11	29	24	5	96	56	80	30	95	92	50
6	44	4	50	31	13	25	1	6	48	38	36	31	11	22	41
7	18	13	39	32	77	77	75	7	16	35	33	32	41	10	19
8	2	31	31	33	98	27	81	8	16	8	37	33	14	39	20
9	1	17	1	34	50	25	48	9	97	31	82	34	28	2	45
10	85	77	53	35	7	22	30	10	50	69	57	35	16	43	19
11	14	38	22	36	4	50	32	11	85	65	50	36	8	47	8
12	45	45	39	37	36	40	14	12	17	4	3	37	4	8	3
13	24	33	33	38	100	73	63	13	48	27	22	38	78	93	43
14	45	18	32	39	21	44	50	14	11	9	33	39	97	20	83
15	90	6	82	40	5	21	16	15	33	46	12	40	1	40	43
16	10	30	15	41	30	48	5	16	6	3	43	41	92	92	35
17	9	32	22	42	20	48	11	17	25	100	80	42	45	41	1
18	74	83	6	43	90	50	97	18	90	70	22	43	80	84	44
19	73	68	10	44	62	70	82	19	32	36	16	44	45	19	34
20	95	66	2	45	13	22	5	20	8	20	23	45	66	73	46
21	3	26	20	46	17	5	12	21	98	60	85	46	67	8	78
22	12	6	28	47	38	50	45	22	47	45	14	47	74	21	71
23	79	39	73	48	45	73	86	23	4	38	10	48	17	18	4
24	5	44	41	49	2	43	45	24	2	45	8	49	16	16	27
25	21	21	8	50	7	95	98	25	40	15	5	50	6	13	4

Fonte: Elaborado pelo próprio autor.

Tabela A.24 - Classe 05 (100x100x100) - Problemas 7 e 8

PROB01-050-07								PROB01-050-08							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	50	8	8	26	41	32	48	1	2	28	21	26	47	30	45
2	42	5	26	27	72	72	27	2	46	49	34	27	9	20	13
3	73	1	98	28	87	15	89	3	40	48	19	28	38	80	90
4	67	97	49	29	24	69	91	4	10	93	69	29	11	21	4
5	34	48	16	30	18	14	25	5	33	81	67	30	22	32	49
6	88	91	23	31	91	93	92	6	86	81	74	31	8	17	22
7	94	100	92	32	19	16	13	7	11	30	20	32	46	23	10
8	29	32	46	33	46	84	77	8	43	9	4	33	29	18	1
9	4	46	34	34	89	65	52	9	30	100	100	34	33	7	34
10	30	46	37	35	73	77	97	10	21	20	48	35	68	32	91
11	80	92	23	36	15	43	33	11	22	93	82	36	88	73	71
12	72	12	83	37	100	99	56	12	53	83	71	37	95	45	68
13	25	22	12	38	18	67	81	13	76	79	3	38	41	27	10
14	70	49	78	39	79	50	56	14	58	90	84	39	20	25	33
15	24	89	83	40	50	8	19	15	14	29	43	40	91	92	46
16	88	73	24	41	3	17	14	16	79	46	97	41	80	83	91
17	92	79	37	42	17	33	44	17	6	27	45	42	42	26	34
18	25	41	39	43	19	79	81	18	38	43	7	43	89	38	81
19	89	60	69	44	32	14	23	19	41	49	27	44	16	10	13
20	20	28	41	45	95	3	69	20	68	37	87	45	50	17	23
21	81	75	23	46	97	82	61	21	42	32	50	46	36	15	16
22	76	54	90	47	39	43	34	22	80	18	98	47	40	76	78
23	20	38	31	48	40	27	46	23	80	87	3	48	100	84	87
24	50	47	22	49	91	88	10	24	64	52	94	49	43	11	40
25	6	10	49	50	89	2	84	25	25	4	46	50	3	40	35

Fonte: Elaborado pelo próprio autor.

Tabela A.25 - Classe 05 (100x100x100) - Problemas 9 e 10

PROB01-050-09								PROB01-050-10							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	5	1	35	26	96	69	41	1	7	23	48	26	67	25	91
2	91	67	43	27	20	41	2	2	5	36	49	27	32	11	40
3	48	46	24	28	23	84	80	3	7	95	73	28	9	19	1
4	69	20	85	29	45	29	23	4	39	31	35	29	32	38	41
5	33	37	37	30	27	74	96	5	34	5	48	30	89	87	50
6	79	38	88	31	92	73	13	6	18	92	79	31	85	10	88
7	8	98	93	32	97	31	69	7	89	75	7	32	1	38	1
8	6	35	13	33	9	33	16	8	20	12	19	33	42	45	32
9	7	25	17	34	87	88	28	9	68	39	83	34	40	10	23
10	13	45	10	35	41	77	69	10	4	19	21	35	50	23	23
11	6	28	22	36	76	74	54	11	43	39	46	36	100	71	8
12	41	27	47	37	16	74	81	12	15	91	86	37	34	72	83
13	78	79	43	38	11	41	43	13	4	3	32	38	32	8	27
14	17	30	34	39	37	2	12	14	33	94	95	39	3	28	40
15	3	20	94	40	68	100	20	15	43	12	26	40	39	14	47
16	6	16	33	41	28	34	25	16	42	84	96	41	38	43	3
17	23	26	4	42	16	81	97	17	37	84	67	42	69	9	72
18	48	44	24	43	18	8	3	18	9	46	42	43	16	27	6
19	46	31	34	44	76	5	72	19	50	14	40	44	37	49	42
20	45	45	32	45	29	34	13	20	8	4	19	45	7	48	28
21	39	16	28	46	27	17	16	21	9	69	85	46	7	20	72
22	4	91	73	47	40	38	23	22	41	43	2	47	84	11	97
23	76	35	86	48	33	66	91	23	16	34	48	48	5	4	26
24	77	50	70	49	8	33	23	24	79	96	20	49	22	6	5
25	70	93	56	50	1	84	70	25	90	43	82	50	50	17	16

Fonte: Elaborado pelo próprio autor.

Tabela A.26 - Classe 06 (10x10x10) - Problemas 1 e 2

PROB01-050-01								PROB01-050-02							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	6	9	5	26	6	1	5	1	7	1	7	26	2	3	8
2	7	1	1	27	5	4	10	2	2	7	7	27	4	3	6
3	4	1	5	28	5	4	2	3	8	7	1	28	1	9	9
4	6	10	4	29	3	8	8	4	5	9	8	29	1	5	5
5	8	2	6	30	8	2	7	5	6	7	9	30	4	8	4
6	6	4	5	31	3	10	2	6	3	8	4	31	4	2	1
7	7	3	10	32	3	8	7	7	5	4	1	32	10	1	4
8	6	2	2	33	4	1	3	8	10	4	8	33	2	10	7
9	5	10	3	34	6	3	6	9	7	6	5	34	1	2	4
10	7	5	6	35	5	5	10	9	8	10	84	35	3	2	2
11	7	7	8	36	6	3	8	11	4	5	2	36	8	3	4
12	7	9	7	37	5	7	10	12	3	3	3	37	4	3	5
13	10	9	1	38	1	2	9	13	10	2	3	38	7	10	4
14	8	4	10	39	7	9	9	14	9	2	7	39	4	7	1
15	6	4	1	40	7	1	4	15	7	8	5	40	9	7	8
16	7	5	9	41	9	9	7	16	1	2	3	41	10	7	4
17	1	10	5	42	9	8	4	17	2	4	9	42	9	9	3
18	3	5	3	43	7	8	4	18	2	7	9	43	4	5	9
19	8	2	4	44	5	1	6	19	7	3	9	44	8	5	10
20	5	2	3	45	1	8	9	20	5	2	10	45	4	4	6
21	2	4	6	46	10	5	3	21	1	10	7	46	5	3	2
22	5	8	7	47	2	6	4	22	5	1	1	47	3	5	3
23	10	10	7	48	2	8	2	23	8	8	9	48	8	5	8
24	10	2	6	49	1	6	6	24	1	4	2	49	9	6	4
25	9	10	5	50	2	2	1	25	5	4	8	50	1	1	4

Fonte: Elaborado pelo próprio autor.

Tabela A.27 - Classe 06 (10x10x10) - Problemas 3 e 4

PROB01-050-03								PROB01-050-04							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	7	4	10	26	7	8	10	1	8	4	10	26	3	10	3
2	4	4	2	27	4	3	3	2	9	10	6	27	3	4	1
3	1	6	8	28	8	5	5	3	5	2	4	28	4	10	4
4	1	7	3	29	10	9	1	4	10	6	9	29	8	6	10
5	5	9	9	30	9	7	8	5	5	4	2	30	3	3	3
6	7	3	4	31	2	3	9	6	4	5	5	31	1	5	10
7	4	4	9	32	8	1	10	7	2	5	8	32	7	4	5
8	5	7	7	33	1	6	2	8	9	1	3	33	9	5	6
9	10	5	8	34	7	8	1	9	4	1	10	34	2	7	7
10	2	10	5	35	2	6	5	10	6	3	7	35	8	3	7
11	8	4	7	36	9	2	3	11	5	2	3	36	1	2	1
12	10	8	8	37	4	8	9	12	4	2	4	37	3	4	4
13	1	2	8	38	2	9	10	13	1	5	2	38	8	5	7
14	1	1	1	39	2	4	4	14	2	7	8	39	1	2	6
15	7	3	8	40	4	4	5	15	8	7	2	40	6	2	9
16	6	10	10	41	8	6	10	16	10	9	4	41	7	4	9
17	4	1	4	42	10	1	9	17	5	7	8	42	10	4	6
18	2	10	8	43	10	1	3	18	3	2	6	43	7	6	8
19	7	3	3	44	8	8	5	19	8	4	8	44	1	2	1
20	4	3	4	45	8	3	2	20	4	5	1	45	3	1	9
21	9	7	9	46	9	2	10	21	8	3	10	46	4	10	9
22	6	3	8	47	3	3	1	22	4	6	4	47	4	2	10
23	7	5	10	48	3	9	3	23	3	3	2	48	7	6	7
24	1	5	1	49	8	5	3	24	4	7	7	49	6	5	1
25	10	9	8	50	7	9	4	25	6	3	1	50	6	6	5

Fonte: Elaborado pelo próprio autor.

Tabela A.28 - Classe 06 (10x10x10) - Problemas 5 e 6

PROB01-050-05								PROB01-050-06							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	8	7	3	26	10	3	5	1	7	9	5	26	6	5	8
2	1	7	1	27	1	2	8	2	4	3	5	27	10	3	4
3	8	1	1	28	3	6	10	3	2	9	5	28	9	1	7
4	6	4	4	29	7	2	6	4	5	3	8	29	5	7	5
5	4	8	4	30	8	2	9	5	4	3	5	30	4	8	4
6	8	10	3	31	9	8	8	6	3	4	4	31	8	10	7
7	1	5	6	32	5	4	1	7	9	6	5	32	2	5	8
8	4	4	10	33	10	1	1	8	8	8	6	33	8	10	5
9	7	8	3	34	8	3	6	9	9	6	5	34	5	2	2
10	9	5	2	35	7	7	10	10	3	8	6	35	5	4	10
11	1	1	8	36	4	1	10	11	8	7	2	36	4	1	8
12	1	7	1	37	3	1	8	12	7	1	7	37	2	5	3
13	4	5	7	38	3	4	1	13	8	8	1	38	9	2	8
14	4	6	4	39	7	1	9	14	5	4	9	39	6	9	3
15	8	2	5	40	1	9	4	15	7	6	7	40	5	7	10
16	5	5	9	41	7	3	5	16	7	4	3	41	6	1	2
17	7	4	3	42	1	4	4	17	10	8	7	42	1	7	1
18	3	7	5	43	3	2	2	18	2	9	1	43	10	9	7
19	8	2	2	44	3	7	6	19	9	3	7	44	4	9	10
20	5	6	7	45	7	10	5	20	3	6	2	45	10	8	2
21	6	10	10	46	8	9	7	21	5	6	3	46	5	7	6
22	5	8	9	47	2	10	8	22	3	1	5	47	3	9	7
23	2	2	3	48	4	10	2	23	10	10	3	48	8	7	8
24	4	8	6	49	7	6	10	24	5	10	2	49	5	4	8
25	3	8	3	50	4	4	7	25	7	2	6	50	3	3	8

Fonte: Elaborado pelo próprio autor.

Tabela A.29 - Classe 06 (10x10x10) - Problemas 7 e 8

PROB01-050-07								PROB01-050-08							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	7	10	8	26	3	10	10	1	8	2	8	26	7	2	3
2	8	10	2	27	8	1	3	2	1	6	6	27	7	2	9
3	5	6	2	28	6	5	3	3	9	4	8	28	2	2	10
4	3	1	3	29	4	3	1	4	10	8	9	29	2	10	8
5	3	7	7	30	9	7	10	5	1	10	8	30	3	3	5
6	9	9	4	31	8	1	5	6	4	1	3	31	7	3	4
7	8	6	3	32	10	7	2	7	6	7	4	32	9	8	9
8	3	1	3	33	7	6	10	8	7	5	9	33	5	5	4
9	4	3	8	34	9	8	1	9	6	1	10	34	6	7	7
10	6	10	9	35	2	8	3	10	10	5	3	35	10	5	5
11	2	6	7	36	7	2	7	11	9	4	1	36	9	10	3
12	4	6	4	37	2	2	5	12	10	10	1	37	1	8	10
13	5	10	6	38	4	1	4	13	5	1	10	38	10	9	1
14	7	3	5	39	4	6	6	14	8	1	2	39	1	4	8
15	7	3	2	40	8	4	5	15	8	7	4	40	2	2	9
16	2	2	8	41	4	10	8	16	6	9	2	41	5	8	7
17	10	5	2	42	2	9	9	17	3	1	4	42	2	10	6
18	2	2	10	43	6	3	1	18	3	4	8	43	3	10	6
19	9	3	1	44	6	4	7	19	10	4	6	44	9	8	1
20	4	9	8	45	4	5	10	20	4	9	3	45	9	3	7
21	3	3	3	46	9	4	4	21	2	9	6	46	4	2	3
22	4	3	2	47	3	7	5	22	2	6	6	47	2	6	4
23	9	7	6	48	5	3	3	23	7	5	6	48	9	8	7
24	5	1	9	49	4	5	7	24	8	3	7	49	2	5	5
25	4	7	8	50	9	1	8	25	8	1	9	50	8	8	1

Fonte: Elaborado pelo próprio autor.

Tabela A.30 - Classe 06 (10x10x10) - Problemas 9 e 10

PROB01-050-09								PROB01-050-10							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	8	5	1	26	4	5	5	1	9	7	3	26	10	9	8
2	5	3	1	27	5	2	6	2	8	9	5	27	4	1	4
3	2	3	5	28	9	6	8	3	6	9	1	28	5	3	5
4	8	6	4	29	1	4	6	4	7	5	8	29	9	1	3
5	2	4	10	30	8	2	1	5	10	9	1	30	2	10	6
6	10	6	3	31	5	6	2	6	5	10	4	31	4	8	3
7	7	7	2	32	7	10	5	7	5	8	1	32	4	1	10
8	4	8	8	33	4	3	9	8	8	2	4	33	2	10	3
9	1	8	3	34	2	3	6	9	3	4	5	34	7	2	2
10	3	7	6	35	9	1	8	10	7	10	10	35	5	6	10
11	5	3	6	36	10	1	2	11	2	9	2	36	2	1	10
12	7	5	7	37	1	3	4	12	3	9	3	37	10	9	9
13	6	3	5	38	5	8	5	13	6	4	9	38	1	6	2
14	10	10	6	39	9	3	1	14	1	8	3	39	8	1	3
15	8	2	7	40	7	9	6	15	9	6	1	40	9	7	10
16	1	7	7	41	3	7	3	16	5	6	1	41	2	5	10
17	3	8	9	42	3	2	2	17	6	4	3	42	3	5	1
18	3	9	7	43	1	4	10	18	2	1	3	43	8	1	5
19	10	4	10	44	9	3	6	19	9	5	5	44	2	5	2
20	3	10	9	45	3	2	3	20	3	2	6	45	6	10	10
21	10	6	6	46	8	1	1	21	1	2	9	46	3	9	10
22	3	8	3	47	2	4	4	22	1	1	7	47	3	3	3
23	6	4	9	48	6	4	4	23	4	2	9	48	10	1	8
24	8	4	4	49	1	6	4	24	9	6	2	49	1	4	2
25	5	6	1	50	6	6	1	25	9	10	4	50	3	5	2

Fonte: Elaborado pelo próprio autor.

Tabela A.31 - Classe 07 (40x40x40) - Problemas 1 e 2

PROB01-050-01								PROB01-050-02							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	1	29	202	26	31	21	20	1	7	26	27	26	32	18	23
2	7	21	31	27	35	19	15	2	17	17	17	27	24	3	11
3	19	31	15	28	25	34	17	3	18	2	21	28	21	9	9
4	26	15	24	29	18	28	28	4	10	34	23	29	21	20	15
5	28	7	11	30	13	17	17	5	31	32	19	30	34	8	14
6	6	4	25	31	8	30	22	6	23	28	19	31	9	27	16
7	27	33	30	32	33	33	32	7	20	9	1	32	35	1	19
8	16	12	32	33	24	26	8	8	5	4	23	33	2	20	32
9	15	25	3	34	6	3	1	9	32	1	15	34	1	7	9
10	17	30	1	35	25	10	30	10	9	3	35	35	3	2	22
11	12	27	13	36	16	13	23	11	24	5	7	36	18	33	9
12	2	4	2	37	15	32	30	12	3	18	28	37	4	28	35
13	10	4	6	38	16	2	19	13	5	17	13	38	17	10	34
14	33	34	20	39	27	4	14	14	19	32	12	39	4	17	6
15	1	9	6	40	32	31	34	15	2	28	15	40	24	32	33
16	7	30	34	41	14	9	7	16	26	22	23	41	5	2	9
17	21	25	30	42	9	28	19	17	17	14	29	42	14	34	33
18	28	35	28	43	12	8	19	18	12	7	4	43	24	30	14
19	18	27	19	44	10	26	1	19	22	33	14	44	28	5	10
20	10	17	3	45	16	23	9	20	25	2	10	45	4	9	1
21	32	4	11	46	30	25	13	21	6	30	7	46	5	28	27
22	30	18	2	47	2	1	14	22	35	16	21	47	33	15	28
23	15	20	32	48	32	33	7	23	28	28	14	48	23	15	13
24	5	22	26	49	16	16	21	24	21	9	32	49	4	16	19
25	29	25	35	50	2	32	21	25	15	29	13	50	16	6	34

Fonte: Elaborado pelo próprio autor.

Tabela A.32 - Classe 07 (40x40x40) - Problemas 3 e 4

PROB01-050-03								PROB01-050-04							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	2	24	35	26	22	28	25	1	18	9	30	26	23	25	28
2	14	14	27	27	24	33	8	2	24	10	1	27	13	29	16
3	6	21	28	28	18	30	35	3	5	27	34	28	14	5	4
4	16	17	23	29	25	34	1	4	35	1	34	29	28	26	35
5	35	9	14	30	9	12	33	5	15	34	22	30	18	3	18
6	27	18	24	31	32	13	9	6	9	30	30	31	21	10	15
7	14	19	29	32	13	26	5	7	7	30	23	32	27	29	15
8	30	32	27	33	16	1	22	8	19	1	18	33	29	30	11
9	15	25	28	34	7	33	26	9	9	1	5	34	2	2	32
10	2	10	35	35	17	16	15	10	6	18	22	35	18	8	7
11	23	19	2	36	9	7	8	11	35	32	8	36	11	27	6
12	5	33	8	37	4	13	4	12	29	12	34	37	28	9	9
13	1	17	33	38	7	19	25	13	31	30	17	38	8	15	17
14	16	31	26	39	27	19	34	14	2	17	18	39	16	32	26
15	2	13	13	40	34	10	4	15	3	32	22	40	21	12	9
16	11	25	25	41	18	31	10	16	30	29	14	41	32	24	24
17	24	16	29	42	20	16	34	17	20	17	28	42	25	34	1
18	7	15	28	43	35	16	8	18	3	22	16	43	12	26	3
19	2	3	8	44	33	8	20	19	18	9	3	44	16	22	6
20	29	23	4	45	3	8	27	20	9	20	11	45	3	6	19
21	14	22	14	46	14	32	5	21	23	13	10	46	24	35	19
22	6	13	18	47	28	28	6	22	34	11	14	47	24	7	20
23	7	25	20	48	3	19	8	23	8	33	2	48	17	1	2
24	1	20	16	49	28	5	18	24	29	7	22	49	16	5	16
25	25	34	13	50	17	14	34	25	11	3	26	50	31	11	35

Fonte: Elaborado pelo próprio autor.

Tabela A.33 - Classe 07 (40x40x40) - Problemas 5 e 6

PROB01-050-05								PROB01-050-06							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	23	7	3	26	25	23	30	1	17	4	10	26	26	20	33
2	21	7	11	27	1	12	13	2	19	3	20	27	25	8	9
3	28	11	6	28	23	26	30	3	27	29	35	28	19	1	22
4	6	19	34	29	32	17	21	4	25	3	33	29	35	32	20
5	19	23	29	30	28	7	14	5	34	13	25	30	14	33	34
6	13	20	23	31	9	8	8	6	18	9	29	31	33	5	2
7	1	5	16	32	5	19	1	7	29	16	10	32	7	10	23
8	9	29	10	33	20	11	1	8	33	33	1	33	33	5	25
9	27	13	18	34	8	28	16	9	9	1	30	34	15	32	22
10	34	25	22	35	32	22	35	10	3	33	21	35	10	14	15
11	11	11	3	36	14	1	5	11	23	12	32	36	4	21	3
12	31	27	26	37	28	6	13	12	32	6	17	37	17	25	18
13	4	30	2	38	33	24	31	13	34	8	21	38	34	32	23
14	34	16	9	39	27	11	19	14	20	14	24	39	16	24	23
15	3	17	20	40	26	14	9	15	27	1	17	40	30	27	20
16	15	20	4	41	22	18	25	16	22	24	28	41	1	11	27
17	27	19	28	42	31	4	14	17	35	8	27	42	1	22	16
18	33	7	5	43	23	12	32	18	17	14	16	43	35	34	27
19	33	2	32	44	33	2	16	19	14	8	27	44	4	4	25
20	25	6	17	45	2	5	10	20	28	26	12	45	25	3	2
21	31	5	5	46	33	4	32	21	5	31	13	46	20	7	11
22	5	8	34	47	7	20	33	22	33	6	30	47	3	34	12
23	22	7	8	48	9	5	32	23	35	15	13	48	23	22	3
24	9	18	6	49	17	6	15	24	25	5	12	49	5	29	13
25	33	8	3	50	9	195	12	25	7	12	16	50	23	28	13

Fonte: Elaborado pelo próprio autor.

Tabela A.34 - Classe 07 (40x40x40) - Problemas 7 e 8

PROB01-050-07								PROB01-050-08							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	22	25	18	26	13	26	18	1	28	22	13	26	17	27	3
2	28	35	7	27	16	10	13	2	26	31	16	27	2	22	14
3	15	1	7	28	4	23	6	3	14	19	13	28	12	32	5
4	8	21	33	29	24	2	30	4	15	28	9	29	7	15	28
5	3	2	32	30	33	26	30	5	6	15	28	30	33	28	15
6	34	34	34	31	20	12	32	6	4	11	28	31	22	23	24
7	23	26	3	32	12	21	15	7	16	2	9	32	34	3	19
8	4	13	8	33	9	23	6	8	12	30	19	33	25	15	4
9	31	5	9	34	12	28	8	9	21	1	20	34	16	27	12
10	22	26	27	35	7	7	2	10	35	25	8	35	25	20	35
11	34	21	9	36	17	22	10	11	34	4	21	36	9	15	23
12	30	20	6	37	24	6	14	12	35	35	35	37	6	18	15
13	17	13	15	38	4	26	16	13	25	21	25	38	25	14	6
14	27	33	27	39	23	29	20	14	3	11	7	39	16	4	8
15	7	27	18	40	14	5	28	15	28	17	24	40	27	7	19
16	30	10	27	41	7	4	29	16	26	19	7	41	5	33	7
17	12	22	5	42	11	8	21	17	3	11	14	42	12	10	31
18	29	13	21	43	21	19	12	18	8	29	28	43	23	30	16
19	9	24	18	44	24	25	5	19	10	19	16	44	4	33	21
20	13	23	8	45	29	34	24	20	24	9	13	45	24	23	32
21	4	3	27	46	33	12	25	21	22	14	16	46	4	2	3
22	14	12	31	47	15	3	33	22	32	1	11	47	17	26	4
23	5	16	19	48	29	30	12	23	27	20	1	48	29	8	27
24	29	17	28	49	24	1	13	24	33	3	2	49	17	30	10
25	28	30	35	50	23	26	28	25	3	21	29	50	3	33	26

Fonte: Elaborado pelo próprio autor.

Tabela A.35 - Classe 07 (40x40x40) - Problemas 9 e 10

PROB01-050-09								PROB01-050-10							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	33	20	21	26	19	25	5	1	4	17	28	26	20	34	8
2	35	28	26	27	25	17	11	2	33	24	35	27	14	1	19
3	2	3	20	28	9	6	8	3	1	9	26	28	5	28	35
4	33	11	9	29	11	29	26	4	17	30	8	29	14	21	13
5	22	4	35	30	8	32	11	5	25	29	31	30	17	35	31
6	20	1	33	31	10	21	17	6	25	25	4	31	34	18	23
7	22	12	2	32	12	5	5	7	15	23	31	32	14	31	15
8	14	23	23	33	4	8	29	8	13	27	14	33	17	25	18
9	16	13	33	34	22	18	31	9	33	24	10	34	17	22	2
10	28	32	31	35	4	11	28	10	32	5	30	35	5	26	20
11	10	18	16	36	35	1	22	11	22	19	22	36	2	21	20
12	2	15	27	37	6	3	19	12	13	29	18	37	30	34	24
13	21	33	10	38	15	23	20	13	16	34	29	38	16	31	12
14	35	10	21	39	4	18	1	14	21	8	13	39	28	31	28
15	28	2	22	40	32	9	31	15	29	21	31	40	24	22	30
16	11	22	32	41	18	27	8	16	30	26	21	41	32	20	10
17	33	13	14	42	18	27	32	17	6	14	13	42	23	10	11
18	3	14	17	43	11	4	10	18	22	21	28	43	23	26	5
19	25	24	10	44	9	13	31	19	29	30	5	44	27	15	17
20	28	30	19	45	23	22	23	20	8	27	26	45	11	20	15
21	30	6	11	46	13	6	16	21	16	32	19	46	23	9	30
22	3	33	8	47	12	4	29	22	31	31	27	47	8	18	8
23	6	29	19	48	21	24	34	23	19	2	24	48	35	6	28
24	13	14	9	49	6	31	9	24	29	1	27	49	6	19	7
25	25	26	6	50	16	6	26	25	34	30	19	50	18	15	27

Fonte: Elaborado pelo próprio autor.

Tabela A.36 - Classe 08 (100x100x100) - Problemas 1 e 2

PROB01-050-01								PROB01-050-02							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	46	39	75	26	36	51	95	1	17	91	97	26	92	13	8
2	27	21	61	27	25	24	50	2	92	27	17	27	84	93	56
3	94	81	35	28	65	84	62	3	88	37	41	28	61	19	39
4	86	60	34	29	53	18	8	4	95	59	38	29	11	55	95
5	8	62	26	30	88	42	27	5	66	27	89	30	24	8	44
6	76	4	35	31	93	100	32	6	73	18	84	31	44	72	51
7	37	3	50	32	13	88	77	7	55	64	1	32	10	91	94
8	76	22	52	33	74	1	43	8	80	4	88	33	52	70	37
9	35	80	23	34	26	13	96	9	57	76	45	34	71	32	4
10	67	75	96	35	95	75	80	10	59	98	60	35	93	72	2
11	27	97	8	36	76	83	98	11	54	5	22	36	88	53	34
12	47	59	87	37	35	37	10	12	23	73	3	37	54	73	45
13	100	79	1	38	71	52	59	13	30	72	23	38	47	70	44
14	18	64	70	39	17	29	19	14	79	2	7	39	24	97	61
15	46	24	91	40	97	61	24	15	57	48	65	40	49	27	48
16	97	65	29	41	99	69	37	16	21	72	93	41	70	17	84
17	11	70	55	42	39	38	54	17	2	94	49	42	79	69	83
18	23	5	73	43	77	38	94	18	12	67	89	43	84	35	9
19	58	32	24	44	55	71	46	19	97	33	39	44	88	85	10
20	35	42	33	45	71	38	69	20	75	32	50	45	14	14	96
21	32	74	46	46	70	35	73	21	31	70	17	46	15	33	82
22	45	78	97	47	92	26	4	22	25	21	61	47	63	15	3
23	40	40	7	48	82	68	32	23	38	48	9	48	38	15	8
24	60	32	86	49	1	6	66	24	21	34	2	49	99	26	84
25	99	10	75	50	62	2	91	25	25	14	58	50	51	41	64

Fonte: Elaborado pelo próprio autor.

Tabela A.37 - Classe 08 (100x100x100) - Problemas 3 e 4

PROB01-050-03								PROB01-050-04							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	87	44	20	26	97	28	20	1	58	44	90	26	53	90	33
2	4	34	22	27	94	13	63	2	69	40	26	27	53	34	21
3	31	46	48	28	58	5	15	3	25	2	54	28	54	40	44
4	51	57	43	29	70	39	81	4	60	56	99	29	28	76	20
5	25	39	99	30	9	27	8	5	35	4	62	30	93	93	73
6	17	33	84	31	42	93	69	6	14	95	85	31	41	65	40
7	74	24	99	32	58	41	10	7	92	85	98	32	7	44	75
8	85	87	77	33	31	86	32	8	89	21	13	33	9	55	26
9	80	25	68	34	67	98	61	9	54	21	90	34	12	17	17
10	52	20	25	35	92	16	25	10	96	43	37	35	38	13	47
11	28	14	37	36	49	72	23	11	55	22	3	36	61	42	11
12	100	88	68	37	24	58	79	12	24	2	84	37	43	94	14
13	61	12	98	38	72	89	80	13	91	5	72	38	48	55	17
14	91	41	91	39	82	14	4	14	52	27	28	39	41	82	46
15	67	73	88	40	54	94	25	15	78	97	62	40	6	12	49
16	46	30	10	41	88	66	30	16	70	89	74	41	7	14	29
17	44	71	44	42	20	51	59	17	35	47	38	42	60	34	36
18	52	30	58	43	90	31	23	18	93	92	26	43	97	76	38
19	87	33	53	44	68	48	75	19	78	34	68	44	1	62	91
20	64	23	14	45	8	43	22	20	4	65	31	45	3	71	49
21	29	67	39	46	59	32	90	21	28	63	10	46	4	30	99
22	6	63	78	47	33	3	1	22	34	6	94	47	4	92	100
23	37	5	60	48	43	9	33	23	83	13	62	48	47	56	57
24	81	85	71	49	98	95	3	24	94	87	87	49	96	15	21
25	100	19	88	50	87	79	84	25	26	23	71	50	76	66	5

Fonte: Elaborado pelo próprio autor.

Tabela A.38 - Classe 08 (100x100x100) - Problemas 5 e 6

PROB01-050-05								PROB01-050-06							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	28	97	13	26	10	53	45	1	47	49	35	26	66	15	58
2	81	47	31	27	11	2	28	2	94	53	35	27	70	23	34
3	68	11	61	28	3	26	20	3	62	19	15	28	99	61	97
4	16	54	4	29	87	12	6	4	25	53	8	29	45	97	45
5	94	68	24	30	78	12	89	5	4	33	35	30	14	78	54
6	58	10	33	31	39	38	58	6	3	24	34	31	38	10	77
7	11	45	96	32	55	94	91	7	29	6	95	32	52	45	8
8	94	4	50	33	40	71	21	8	98	38	86	33	18	40	15
9	77	18	13	34	8	83	26	9	99	66	35	34	5	2	82
10	89	65	2	35	37	57	70	10	33	88	66	35	35	54	40
11	81	31	18	36	74	61	100	11	8	87	32	36	34	31	88
12	1	17	1	37	13	31	48	12	77	31	17	37	32	15	83
13	74	45	47	38	73	74	1	13	4	38	21	38	49	92	38
14	64	66	64	39	47	51	89	14	25	4	49	39	6	19	83
15	88	22	85	40	11	79	74	15	47	46	7	40	15	97	50
16	95	95	39	41	77	63	75	16	67	54	3	41	96	11	22
17	77	24	33	42	1	64	64	17	20	48	27	42	41	47	41
18	33	7	95	43	3	72	52	18	22	69	11	43	10	69	67
19	68	82	82	44	33	77	56	19	59	83	97	44	14	39	20
20	45	56	47	45	97	100	75	20	33	46	12	45	40	28	2
21	26	60	80	46	48	29	7	21	25	56	3	46	45	27	16
22	15	48	59	47	22	80	98	22	43	91	75	47	93	69	97
23	82	22	13	48	4	50	82	23	80	30	63	48	8	97	58
24	54	38	56	49	47	36	40	24	15	40	72	49	45	4	58
25	53	28	53	50	64	4	77	25	27	32	33	50	53	43	98

Fonte: Elaborado pelo próprio autor.

Tabela A.39 - Classe 08 (100x100x100) - Problemas 7 e 8

PROB01-050-07								PROB01-050-08							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	17	50	58	26	23	30	70	1	88	2	28	26	27	92	83
2	58	60	92	27	28	91	93	2	71	66	96	27	87	12	99
3	5	76	22	28	96	95	73	3	99	84	28	28	92	82	50
4	33	51	13	29	4	33	31	4	90	98	69	29	62	70	18
5	63	97	97	30	99	97	70	5	21	10	8	30	83	63	35
6	99	39	34	31	88	31	95	6	44	1	83	31	87	3	14
7	48	66	93	32	100	47	72	7	66	27	44	32	49	98	89
8	3	21	23	33	97	56	10	8	7	55	59	33	75	25	4
9	74	63	58	34	49	68	91	9	96	11	80	34	46	87	47
10	26	10	79	35	82	98	63	10	70	85	43	35	80	95	85
11	82	96	47	36	47	2	77	11	9	4	61	36	59	20	13
12	54	46	34	37	2	52	65	12	30	60	50	37	21	88	100
13	35	30	96	38	74	11	74	13	68	71	70	38	50	29	11
14	37	43	85	39	64	36	26	14	98	81	22	39	71	4	68
15	57	23	82	40	68	64	75	15	68	47	4	40	72	82	9
16	92	12	68	41	14	60	68	16	16	19	32	41	85	8	67
17	10	25	22	42	82	29	69	17	53	1	64	42	22	60	46
18	62	32	80	43	16	13	81	18	3	94	48	43	23	10	96
19	49	83	11	44	46	54	37	19	40	84	26	44	79	68	1
20	74	89	28	45	34	5	80	20	14	79	93	45	29	33	7
21	23	53	73	46	89	74	24	21	22	49	96	46	34	72	33
22	24	33	92	47	63	57	95	22	52	76	56	47	82	46	94
23	79	87	66	48	65	43	83	23	77	95	16	48	69	38	7
24	75	91	89	49	44	25	77	24	88	93	57	49	42	45	95
25	54	37	18	50	89	81	18	25	28	41	49	50	78	68	91

Fonte: Elaborado pelo próprio autor.

Tabela A.40 - Classe 08 (100x100x100) - Problemas 9 e 10

PROB01-050-09								PROB01-050-10							
Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j	Caixa	w_j	h_j	d_j
1	58	55	51	26	84	55	95	1	29	7	73	26	40	69	8
2	35	73	1	27	45	32	6	2	48	79	5	27	4	1	64
3	42	93	35	28	89	16	78	3	36	49	41	28	85	3	55
4	98	96	74	29	21	54	56	4	7	95	78	29	79	91	43
5	32	74	70	30	68	82	51	5	90	39	81	30	52	100	16
6	40	16	83	31	85	76	32	6	85	30	84	31	84	48	3
7	37	87	42	32	97	100	5	7	55	48	41	32	94	51	70
8	64	38	48	33	54	93	99	8	68	72	84	33	32	10	93
9	71	8	3	34	42	53	56	9	93	4	25	34	87	72	12
10	63	7	56	35	79	91	8	10	7	30	20	35	25	36	30
11	35	13	76	36	20	91	2	11	62	69	42	36	32	61	90
12	7	75	67	37	91	73	34	12	83	89	83	37	10	9	69
13	96	63	45	38	75	48	95	13	26	4	19	38	51	66	32
14	10	20	6	39	29	73	11	14	71	58	43	39	88	41	53
15	78	72	27	40	77	49	76	15	89	96	1	40	29	67	100
16	41	77	97	41	3	57	13	16	65	36	61	41	22	5	60
17	43	78	59	42	63	42	22	17	86	54	53	42	3	25	51
18	43	9	17	43	81	54	10	18	32	71	33	43	88	51	25
19	30	84	40	44	59	83	66	19	69	85	55	44	92	45	82
20	3	70	9	45	23	62	33	20	43	12	26	45	66	90	60
21	20	46	66	46	78	71	41	21	71	42	89	46	23	69	50
22	33	18	73	47	52	34	44	22	61	61	37	47	23	23	43
23	76	4	19	48	26	84	84	23	74	12	69	48	30	31	8
24	48	44	74	49	41	66	14	24	9	46	42	49	91	34	32
25	55	46	31	50	66	6	11	25	29	50	14	50	3	45	32

Fonte: Elaborado pelo próprio autor.