Otimização do Problema de Carregamento de Container Usando uma Metaheurística Eficiente

Eliane Vendramini

Orientador: *Prof. Dr. Rubén Romero* - DEE/FEIS/UNESP

Dissertação submetida ao Programa de Pós Graduação em Engenharia Elétrica da Faculdade de Engenharia de Ilha Solteira - UNESP, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Ilha Solteira - SP, Fevereiro de 2007.

FICHA CATALOGRÁFICA

Elaborada pela Seção Técnica de Aquisição e Tratamento da Informação/Serviço Técnico
de Biblioteca e Documentação da UNESP-Ilha Solteira

Vendramini, Eliane.
V453o Otimização do problema de carregamento de container usando uma metaheurística eficiente / Eliane Vendramini -- Ilha Solteira : [s.n.], 2007
103 p. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2007

Orientador: Rubén Romero
Bibliografia: p. 95-97

1. Otimização combinatória. 2. Sistemas de unitização de cargas. 3. Cargas – Manuseio. 4. Programação heurística. 5. Algoritmos genéticos.

unesp UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO: Otimização do Problema de Carregamento de Container Usando uma Metaheurística Eficiente

AUTORA: ELIANE VENDRAMINI
ORIENTADOR: Prof. Dr. RUBEN AUGUSTO ROMERO LÁZARO

Aprovada com parte das exigências para obtenção do Título de MESTRE em ENGENHARIA ELÉTRICA pela Comissão Examinadora:

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

Prof. Dr. ANTONIO PADILHA FELTRIN
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. ARIOVALDO VERANDIO GARCIA
Departamento de Sistemas de Energia Elétrica /Universidade Estadual de Campinas

Data da realização: 22 de fevereiro de 2007.

Presidente da Comissão Examinadora
Prof. Dr. Ruben Augusto Romero Lázaro

Ao meu Pai e a minha Mãe...

Rosalino e Lucilei - pela dedicação aos filhos, pelos ensinamentos e conselhos sempre oportunos, pelo apoio e incentivo aos estudos acreditando e torcendo pela minha vitória e principalmente pelo amor transmitido em todos os momentos e circunstâncias de minha vida.

"Tudo posso naquele que me fortalece."

(Fp 4:13)

Agradecimentos

A Deus... que na sua infinita misericórdia deu-me forças para chegar até aqui.

A meus pais... pela educação e apoio, mesmo nos momentos mais difíceis, dando a oportunidade de prosseguir nos estudos.

As minhas irmãs Daniele e Camile, familiares e amigos... pela paciência e compreensão por estar ausente, dedicando-me aos estudos.

Ao meu noivo Márcio... pelo incentivo, auxílio, companheirismo e compreensão de minha ausência e principalmente por estar sempre do meu lado, me apoiando, torcendo pelo meu sucesso. Estará sempre em meu coração.

Aos meus colegas em especial a Silvia... que me proporcionaram bons momentos juntos.

Ao Prof. Dr. Rubén Romero... pela orientação, paciência, incentivo e dedicação durante todo o tempo que trabalhamos juntos.

A todos os professores... pela paciência em ensinar, pela dedicação aos alunos e pelas lições transmitidas.

E a Capes... pelo apoio financeiro, contribuindo para a conclusão deste trabalho.

Resumo

No âmbito de pesquisa operacional o problema de carregamento de *container* é conhecido por determinar uma configuração de carga que procure otimizar o que será carregado em um *container*, levando em consideração o máximo de volume ocupado pela carga.

Este problema tem diversas variantes para casos específicos. Existem casos onde a carga é homogênea ou heterogênea, onde a carga pode ser rotacionada em todas as suas dimensões, onde um lucro é associado a cada caixa carregada, entre outras variantes, onde a questão não é a carga e sim o *container*. A classificação do problema está diretamente ligada a suas restrições.

O estudo de carregamento de *container* aqui no Brasil começou ser realizado com mais ênfase há pouco tempo, por ter despertado interesses financeiros em empresas públicas e privadas, já que o transporte utilizando *containers* é oneroso e cobrado por *container* alugado e não pela quantidade de itens que serão carregados. Por isso a vantagem de aproveitar o volume do *container* ao máximo.

Na literatura podem ser encontradas diversas propostas de solução para cada variante do problema, sendo estas propostas determinísticas ou utilizando heurísticas e metaheurísticas.

O estudo realizado para a apresentação desta dissertação descreve de maneira ampla as heurísticas que estão sendo empregadas na resolução do problema estudado, bem como propõe uma nova heurística especializada.

O trabalho aqui apresentado traz ainda uma metaheurística especializada, o algoritmo genético Chu-Beasley. Portanto, foram desenvolvidos dois algoritmos: um heurístico e um metaheurístico. Estes algoritmos simularam o carregamento de um *container* com caixas retangulares e de diferentes tamanhos, sendo no final comparados os resultados obtidos pelos dois algoritmos.

Os dois algoritmos aqui propostos trabalham de maneira diferente dos encontrados na literatura. Neste estudo um dos métodos de resolução, o heurístico, propõe uma divisão do *container* em quatro partes: Parte Principal ou Corpo Principal do *Container*, Espaço Lateral Residual, Espaço Superior Residual e Espaço Frontal Residual e cada uma dessas partes são encaradas como um novo problema de carregamento de *container*. Já o outro

método de resolução relatado neste estudo, o algoritmo genético Chu-Beasley, propõe o carregamento do *container* através de torres formadas por caixas retangulares. As torres são alocadas no *container* de maneira sistemática seguindo regras de decomposição do espaço do *container*.

Assim sendo, o estudo realizado tem como principal objetivo contribuir com pesquisas afins, definindo conceitos sobre o problema e trazendo novas propostas de solução para o problema de carregamento de *container*.

Palavras chave: Problema de Carregamento de *Container*, Heurísticas, Algoritmo Genético.

Abstract

In the ambit of the operational research the container loading problem is known by optimized the load that it will be carried in a container, taking in consideration the maximum of volume occupied by the load.

This problem has several variants for specific cases. Cases exist where the load is homogeneous or heterogeneous, where the load can be rotated in whole its dimensions, where a profit associated to each loaded box exists, among other variants, where the subject is not the load, but the container. The classification of the problem is directly tied up to its restrictions.

The study of the container loading problem here in Brazil it began to be accomplished with more emphasis at little time, for having wakened up financial interests in public and private companies, since the transport using containers is onerous and collected by rented container and not for the amount of items that you will be loaded. That the advantage of taking advantage of the volume of the container to the maximum.

In the literature it can be found several proposed of solution for each variant of the problem. Being these proposed deterministics or using heuristics and metaheuristics.

The study accomplished for the presentation of this dissertation brings in a wide way the heuristics that you are being used in the resolution of the problem, as well as it proposes a new heuristic specialized for the resolution of the container loading problem.

The work here presented he still brings a metaheuristic specialized for the resolution of the problem, the Chu-Beasley genetic algorithm. Therefore, two algorithms were developed: a heuristic and a metaheuristic. These algorithms simulated the shipment of a container with rectangular boxes and of different sizes, being in the compared end the obtained results for the two algorithms.

The two algorithms here proposed they work in way different from the found in the literature. In this study one of the methods of resolution, the heuristic, proposes a division of the container in four parts: It leaves Main or Main Body of the Container, It Residual Lateral Space, It Space Superior Residual and It Space Residual Frontal and each one of those parts is faced as a new container loading problem. Already the other resolution method told in this study, the genetic algorithm Chu-Beasley, proposes the container

loading through towers formed by boxes rectangular. The towers are allocated in the container in a systematic way following rules of decompose of the space of the container.

Being the accomplished study has like this as main objective to contribute with researches to ends, defining concepts about the container loading problem and bringing a new heuristic for the resolution of the problem.

Keywords: Container Loading Problem, Heuristics, Genetic Algorithm.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução Geral

O problema de Carregamento de *Container* é clássico em pesquisa operacional, cujo objetivo geral é determinar uma configuração de carga tal que o volume utilizado em relação ao volume total disponível do *container* seja maximizado. Em outras palavras o principal objetivo é ocupar ao máximo o volume do *container*.

Na literatura ele é diferenciado entre aqueles problemas em que a carga completa tem que ser armazenada, podendo usar mais de um *container* (conhecido como Problema *Bin-Packing*) e aqueles que toleram que alguns itens sejam deixados para trás, utilizando somente um *container* (conhecido como problema *Knapsack*). Outro tipo de suposta diferenciação é a definição de tipos de carga que será alocada no *container*, a carga pode ser homogênea (um tipo de caixa somente) e carga heterogênea (muitos tipos de caixa com poucas caixas de cada tipo).

O problema também pode ser diferenciado através do tipo de item que será carregado, se este item poderá ser rotacionado em suas três dimensões ou se o item exige que alguma dimensão seja fixada, como a altura, por exemplo.

Outros objetivos do problema são: Maximização do peso da carga, do equilíbrio (centro de gravidade) da carga, do valor monetário associado à carga, ou quando o *container* considerado é transportado por vias rodoviárias (caminhões de carga), tem-se o objetivo de minimizar a diferença entre os pesos da parte frontal e traseira, esquerda e direita do suposto *container* proporcionando sua estabilidade. A classificação do tipo de problema está diretamente relacionada às suas restrições.

Todos os esforços são usados para que os itens de carga sejam adequados dentro do *container* da melhor maneira possível, visando os objetivos citados acima.

A importância da abordagem deste problema está na prática constante do transporte internacional de mercadorias através de *containers*. Ele pode ser feito tanto por meio rodoviário, como por marítimo, ferroviário ou aéreo. As exportações para outros continentes, na sua maioria são feitas por meio marítimo, por ser mais cômodo, seguro e

barato, fazendo ainda com que a carga tenha um maior aproveitamento.

O transporte de mercadorias por *container* proporciona as empresas que o utiliza maior segurança, rapidez, garantia de qualidade do produto transportado e principalmente formas de minimizar o custo de transporte.

Este tipo de transporte é oneroso e cobra-se pelo *container* alugado e não pela quantidade de produto que será carregada. Por isso a vantagem de aproveitar o volume do *container* ao máximo.

Se o carregamento do *container* não for bem planejado, pode comprometer o valor final da mercadoria transportada ou até ser encarado como prejuízo para quem vendeu o produto que está sendo transportado.

O *container* do qual escreveu-se até agora nada mais é do que uma caixa inviolável, podendo ser fabricado em madeira, alumínio ou aço, uma vez carregado, só será aberto no seu destino. Por este motivo desperta tamanho interesse às empresas públicas e privadas.

Os *containers* têm suas medidas em "pés" ou "polegadas", sendo padronizados nas medidas de 20' e 40'. As medidas têm origem inglesa, devido ao desenvolvimento naval que essa nação teve ao longo do tempo.

Como durante um longo período os ingleses foram os maiores transportadores marítimos mundiais, a terminologia por eles adotada acabou sendo utilizada por quase todos os países. Atualmente, além das medidas inglesas, os *containers* já são identificados pelo sistema métrico-decimal, apresentando as duas classificações para facilitar o transporte.

Os dados do *container* de 20' são: capacidade volumétrica de $33m^3$, capacidade de peso de 27 toneladas, largura igual a $2,37$ metros, comprimento igual a $5,94$ metros e altura de $2,28$ metros.

Já os dados do *container* de 40' são: capacidade volumétrica de $67m^3$, capacidade de peso de 27 toneladas, largura igual a $2,32$ metros, comprimento igual a $12,04$ metros e altura de $2,38$ metros.

Antes de escolher um *container* é necessário verificar qual o melhor tamanho e a melhor capacidade a ser utilizada, que varia de acordo com o produto. O *container* de 20', normalmente é utilizado para cargas mais pesadas e menos volumosas como metais e ferros. Já o *container* de 40', é mais adequado para cargas mais leves e volumosas.

O problema que será abordado neste trabalho é do tipo *knapsack* com cargas testadas tanto fortemente homogêneas como fortemente heterogêneas. Este problema é considerado

NP-difícil e portanto complexo de ser resolvido matematicamente e deterministicamente. Por não se tratar de algo trivial, fatores como várias restrições com relação ao empilhamento de cargas, pesos, restrições dimensionais, centro de gravidade, quanto à orientação do posicionamento das cargas, valores monetários e outros, dificultam a determinação de uma solução ótima. A solução do problema é extremamente difícil de ser encontrada analiticamente, e, em termos computacionais, é pouco provável que exista um algoritmo de baixa complexidade, com base em uma abordagem matemática e determinística.

Neste tipo de problema justifica-se o emprego de técnicas heurísticas ou metaheurísticas de otimização para resolução aproximada, levando à determinação de uma solução ótima ou sub-ótima que possa ser utilizada.

O trabalho aqui exposto realiza uma ampla pesquisa na literatura apontando quais as heurísticas que estão sendo empregadas na resolução do problema de carregamento de *container* e propõe como resolução dois novos métodos.

Uma das propostas é uma heurística diferenciada, onde é proposta uma divisão do *container* em quatro partes: Parte Principal ou Corpo Principal do *Container*, Espaço Lateral Residual, Espaço Superior Residual e Espaço Frontal Residual e cada uma dessas partes são encaradas como um novo problema de carregamento de *container knapsack*.

O carregamento deve começar do ponto de origem do *container* que é o canto inferior esquerdo do fundo do *container*, partindo da esquerda para direita, que depois de completo, começa a subir uniformemente, logo após, o preenchimento é realizado para frente do *container*, sempre obedecendo à ordem das caixas mais pesadas por baixo das mais leves. A realização do carregamento pode ser tanto manual como por meio de empilhadeira e paleteira, mas sempre levando em consideração o peso e a uniformidade da carga.

Outra proposta deste trabalho é o algoritmo genético. O algoritmo proposto é especializado para resolver o problema deste estudo, e trabalha segundo as idéias de Chu-Beasley, onde se tem o maior controle sobre a diversidade da população.

A proposta usando o algoritmo genético como método de resolução, além de seguir as etapas do algoritmo genético de Chu-Beasley, também utiliza novas etapas, trabalhando com os conceitos de torres, vértices de alocação, largura residual e *container* virtual para a alocação das caixas no interior do *container*. A decomposição do espaço do *container* é realizada de maneira sistemática seguindo regras de alocação.

Algumas características do problema que estão relacionadas com o carregamento de *container*, como é o caso do problema de unitização de carga através de *pallets* não serão

comentadas por fugir do escopo deste trabalho.

O principal objetivo do estudo realizado é contribuir com pesquisas afins, definindo conceitos sobre o problema e trazendo dois novos métodos de resolução do problema em questão: uma nova heurística para a resolução do problema de carregamento de *container* e ainda uma metaheurística, muito conhecida na área de pesquisa operacional, o Algoritmo Genético.

Este trabalho está dividido em 5 capítulos onde a pesquisa realizada foi de uma forma geral apresentada neste primeiro capítulo. No capítulo 2 serão apresentados com mais detalhes o problema de carregamento de *container* e as heurísticas que estão sendo utilizadas para a resolução do mesmo, abordando também a modelagem matemática do problema. No capítulo 3 serão exibidos: a heurística proposta pelo estudo, os testes e resultados obtidos. No capítulo 4 o trabalho traz ainda uma proposta de algoritmo genético Chu-Beasley especializado para o problema de carregamento de *container*. Mostra também os testes realizados e resultados obtidos com a aplicação desta metaheurística na resolução do problema.

No capítulo 5 o trabalho é finalizado com as conclusões obtidas após a análise da pesquisa e seus resultados. A bibliografia pesquisada para a realização deste trabalho é apresentada logo na seqüência. Também o apêndice trazendo os dados dos sistemas testados é mostrado no final da dissertação.

2 *Análise do Problema de Carregamento de Container*

Este capítulo apresenta de maneira formal o problema de carregamento de *container*, trazendo sua formulação matemática e ainda as propostas mais relevantes sobre o assunto. São abordados de maneira ampla os estudos realizados para a obtenção da solução ótima ou quase ótima do problema de carregamento de *container* e suas variantes. O objetivo é mostrar a complexidade do problema aqui abordado em termos de programação matemática, para justificar a aplicação de um método heurístico ou metaheurístico, como é o caso do algoritmo genético, para a resolução do problema a ser proposto nos próximos capítulos.

Inicia-se este capítulo abordando o problema estudado neste trabalho e suas variantes, definindo formalmente o problema de carregamento de *container* e os conceitos envolvidos; logo em seguida é descrita a modelagem matemática do problema; segue-se com uma breve revisão bibliográfica, através da abordagem das técnicas de solução encontradas na literatura; finalizando com o estudo mais detalhado das propostas que ainda estão em discussão por sua importância e complexidade.

2.1 Introdução

O problema de carregamento de *container* tem numerosas aplicações no corte e empacotamento industrial. Por exemplo, quando se necessita aproveitar ao máximo uma peça de tecido a ser cortada para produção de roupas, aproveitamento de chapas de madeira, alumínio, borracha e tantos outros produtos industrializáveis que têm suas matérias prima em dimensões maiores a serem cortadas mais tarde na industrialização, o problema tem aplicação também no carregamento de objetos em *pallets*, ou no carregamento de *containers* de carga.

O problema foi inicialmente estudado por Gilmore e Gomory em (GILMORE; GOMORY,

1961), (GILMORE; GOMORY, 1963) e (GILMORE; GOMORY, 1965) nos anos sessenta, e a partir de então, foram apresentados numerosos documentos e algoritmos para sua solução.

Na literatura técnica, existem várias definições sobre o problema de carregamento de *container*. Encontram-se até artigos onde seus autores se preocuparam em enumerar os nomes encontrados na literatura para cada tipo de variante do problema, como é o caso do artigo escrito por Dyckhoff (DYCKHOFF, 1990). Para definir o que caracteriza este problema basta tentar armazenar de maneira eficiente objetos em meios de transporte. Este tipo de problema pode ser tranqüilamente modelado como um problema do carregamento de *container*. Existem relatos de variantes deste problema que também serão descritas neste trabalho.

Uma das definições encontradas na literatura sobre o problema de carregamento de *container* foi descrita por Bortfeldt e Gehring em (BORTFELDT; GEHRING, 2001) onde o problema de carregamento de *container* é descrito como um conjunto de pacotes retangulares (caixas) que devem ser arranjados em um ou mais *containers* retangulares de tal maneira que o melhor uso do espaço disponível do *container* seja feito para o armazenamento da carga. O carregamento ótimo de um *container* reduz o custo do transporte como também aumenta a estabilidade e apoio da carga.

Há, porém, diversas variantes do problema de carregamento de *container* que dependem da maneira como o padrão de carregamento será avaliado e/ou restrições apresentadas.

Segundo Bortfeldt e Gehring em (BORTFELDT; GEHRING, 2001) o problema de carregamento de *container* é diferenciado por várias características.

Primeiro, os problemas em que a carga completa tem que ser armazenada são diferenciados daqueles que toleram que alguns bens sejam deixados para trás.

Os problemas do primeiro tipo são conhecidos na literatura como (problema tridimensional) Problema *Bin-Packing*. O segundo tipo, (problema tridimensional) Problema *Knapsack*. Enquanto os problemas *Bin-Packing* pretendem, por exemplo, minimizar os custos dos *containers* requeridos, o alvo dos problemas *Knapsack* deve ser geralmente maximizar o volume armazenado.

Um outro tipo de suposta diferenciação ainda proposta por Bortfeldt e Gehring em (BORTFELDT; GEHRING, 2001) é a diferenciação do tipo de carga. Este problema diferencia-se entre carga homogênea (um tipo de caixa somente) e carga heterogênea (muitos tipos de caixas e poucas caixas de cada tipo). Duas caixas são exatamente do

mesmo tipo se, dada uma orientação espacial apropriada, coincidirem em todas as três dimensões laterais.

Já a questão das variantes do problema de carregamento de *container*, segundo Pisinger em (PISINGER, 2002) estão divididas da seguinte maneira:

Empacotamento de Pilhas: Nesta variante, o *container* fixou a largura e a altura, mas a profundidade é ilimitada. O problema consiste em empacotar todas as caixas tal que a profundidade do *container* seja minimizada. O problema de empacotamento de pilhas tem aplicações em situações conhecidas como "multi-rota", onde a carga deveria ser dividida em seções distintas que correspondem aos destinos diferentes.

Carregamento *Knapsack*: No carregamento *Knapsack* de um *container* cada caixa tem um lucro associado, e o problema é escolher um subconjunto das caixas que ajustam em um único *container* de forma que o máximo de lucro esteja carregado. Se fixado o lucro de uma caixa a seu volume, este problema corresponde ao de minimização de espaço perdido.

Carregamento *Bin-Packing*: Neste problema, todos os *containers* fixaram dimensões, e todas as caixas serão empacotadas em um número mínimo de *containers*.

Carregamento Multi-*Container*: Este problema é semelhante ao *Bin-Packing* exceto pelos *containers* poderem ter dimensões variadas e o objetivo é escolher um subconjunto dos *containers* que resultam no menor custo de envio.

Pisinger em (PISINGER, 2002) ainda diz que podem ser impostas várias outras restrições aos problemas acima: Somente rotações específicas podem ser permitidas, soluções factíveis poderiam ser restringidas, como por exemplo, pode ser exigido que o peso do *container* seja equilibrado, pode haver restrições nas quais quantas caixas podem ser postas em cima uma das outras, ou pode ser necessário colocar lotes de um tipo de caixa, próximo uns dos outros. Também deve ser levado em conta o suporte das caixas, entre outras exigências práticas que podem ser impostas ao problema.

O problema considerado neste trabalho, como já foi dito, é o Problema de Carregamento de *Container Knapsack* onde será associado a cada tipo de caixa um lucro. As caixas podem ser giradas em qualquer direção. O objetivo dos algoritmos propostos nos próximos capítulos para a resolução do problema de carregamento de *container* será carregar o *container* o máximo possível, sem levar o apoio dos itens em consideração. Em princípio, os espaços vazios poderão ser preenchidos com borracha ou espuma para assegurar um apoio formal das caixas.

Chen, Lee e Shen em (CHEN; LEE; SHEN, 1995) desenvolveram um modelo analítico para o problema de carregamento de *container* utilizando variáveis inteiras e binárias, onde são consideradas caixas retangulares e de tamanhos não uniformes e é previsto um carregamento em diferentes *containers* de dimensões possivelmente diferentes entre si. No modelo matemático proposto, o objetivo é encontrar uma solução que minimize o espaço desperdiçado do *container* e minimize o número de *containers* necessários para carregar todas as caixas. O modelo será descrito na próxima seção.

2.2 Modelagem Matemática

Apresenta-se a seguir o modelo matemático proposto por Chen, Lee e Shen em (CHEN; LEE; SHEN, 1995), considerando o carregamento em apenas um *container* e permissão para rotacionar as caixas em quase todos os eixos.

A largura será a dimensão horizontal de maior valor e o comprimento, a dimensão horizontal de menor valor.

A Figura 1, abaixo, auxilia no entendimento da descrição acima.

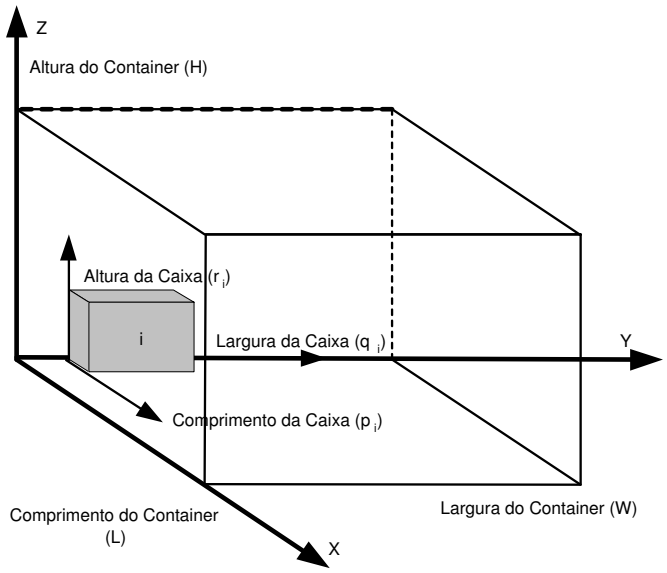

Figura 1: Ilustração do posicionamento da caixa, de acordo com suas dimensões e as dimensões do *container*.

Neste modelo, supõe-se que as dimensões das caixas e do *container* são conhecidas, bem como a quantidade de caixas a serem carregadas. As caixas são independentes e

são posicionadas ortogonalmente no *container*, ou seja, são posicionadas em paralelo ou perpendicular aos eixos. As caixas podem sofrer rotações de 90^o no posicionamento, de tal forma que a largura e o comprimento da caixa não sejam posicionados obrigatoriamente em paralelo à largura e ao comprimento do *container*, respectivamente, contudo, a altura da caixa sempre será posicionada em paralelo ao eixo da altura do *container*.

As notações listadas abaixo são necessárias para o entendimento do modelo matemático:

$\boldsymbol{N}$ - número de caixas disponíveis para carregamento;

$\boldsymbol{s_i}$ - variável binária que indica se a caixa foi posicionada no *container*. Quando isso ocorre, $s_i = 1$, caso contrário, $s_i = 0$;

$\boldsymbol{(L, W, H)}$ - triplo que indica comprimento, largura e altura do *container*, respectivamente;

$\boldsymbol{(p_i, q_i, r_i)}$ - triplo que indica comprimento, largura e altura da caixa, respectivamente;

$\boldsymbol{(x_i, y_i, z_i)}$ - triplo que indica a locação da caixa pelo canto inferior esquerdo traseiro;

$\boldsymbol{(l_{xi}, l_{yi}, l_{zi})}$ - triplo binário que indica para qual eixo o comprimento da caixa está em paralelo. Como a altura da caixa sempre está em paralelo com a altura do *container*, podemos trabalhar com o triplo $(l_{xi}, l_{yi}, 0)$;

$\boldsymbol{(w_{xi}, w_{yi}, w_{zi})}$ - variável binária que indica para qual eixo a largura da caixa está em paralelo. Como a altura da caixa sempre está em paralelo com a altura do *container*, essa variável pode ser modificada para $(w_{xi}, w_{yi}, 0)$;

$\boldsymbol{(h_{xi}, h_{yi}, h_{zi})}$ - triplo binário que indica para qual eixo a altura da caixa está em paralelo. Como a altura da caixa sempre estará em paralelo com a altura do *container*, esse triplo será fixo: $(0, 0, 1)$;

Ainda existem outras variáveis que são usadas para indicar o posicionamento das caixas em relação a outras caixas:

a_{ik} - caso seja 1, indica que a caixa i está à esquerda da caixa k;

b_{ik} - caso seja 1, indica que a caixa i está à direita da caixa k;

c_{ik} - caso seja 1, indica que a caixa i está atrás da caixa k;

d_{ik} - caso seja 1, indica que a caixa i está à frente da caixa k;

e_{ik} - caso seja 1, indica que a caixa i está abaixo da caixa k;

f_{ik} - caso seja 1, indica que a caixa i está acima da caixa k;

O problema então é formulado conforme abaixo, onde se observa que o objetivo é minimizar o espaço não preenchido do *container*:

min $\quad L \cdot W \cdot H - \sum_{i=1}^{N} p_i \cdot q_i \cdot r_i \cdot s_i$

Sujeito a

(Evitar sobreposições de caixas no *container*)

$$x_i + p_i \cdot l_{xi} + q_i \cdot w_{xi} + r_i \cdot h_{xi} \leq x_k + (1 - c_{ik}) \cdot M, \forall i, k, i < k \tag{2.1}$$

$$x_k + p_k \cdot l_{xk} + q_k \cdot w_{xk} + r_k \cdot h_{xk} \leq x_i + (1 - d_{ik}) \cdot M, \forall i, k, i < k \tag{2.2}$$

$$y_i + p_i \cdot l_{yi} + q_i \cdot w_{yi} + r_i \cdot h_{yi} \leq y_k + (1 - a_{ik}) \cdot M, \forall i, k, i < k \tag{2.3}$$

$$y_k + p_k \cdot l_{yk} + q_k \cdot w_{yk} + r_k \cdot h_{yk} \leq y_i + (1 - b_{ik}) \cdot M, \forall i, k, i < k \tag{2.4}$$

$$z_i + p_i \cdot l_{zi} + q_i \cdot w_{zi} + r_i \cdot h_{zi} \leq z_k + (1 - e_{ik}) \cdot M, \forall i, k, i < k \tag{2.5}$$

$$z_k + p_k \cdot l_{zk} + q_k \cdot w_{zk} + r_k \cdot h_{zk} \leq y_i + (1 - b_{ik}) \cdot M, \forall i, k, i < k \tag{2.6}$$

(Garantir que o par de caixas avaliados nas equações (2.1) a (2.6) está no *container*)

$$a_{ik} + b_{ik} + c_{ik} + d_{ik} + e_{ik} + f_{ik} \geq s_i + s_k - 1, \forall i, k, i < k \tag{2.7}$$

(Garantir que o posicionamento das caixas obedeça às limitações físicas dimensionais do *container*)

$$x_i + p_i \cdot l_{xi} + q_i \cdot w_{xi} + r_i \cdot h_{xi} \leq L + (1 - s_i) \cdot M, \forall i \tag{2.8}$$

$$y_i + p_i \cdot l_{yi} + q_i \cdot w_{yi} + r_i \cdot h_{yi} \leq W + (1 - s_i) \cdot M, \forall i \tag{2.9}$$

$$z_i + p_i \cdot l_{zi} + q_i \cdot w_{zi} + r_i \cdot h_{zi} \leq H + (1 - s_i) \cdot M, \forall i \tag{2.10}$$

M é um número inteiro arbitrário e grande.

Os triplos binários que identificam o posicionamento das caixas com relação aos eixos são variáveis dependentes e se relacionam conforme abaixo:

$$l_{xi} + l_{yi} + l_{zi} = 1 \tag{2.11}$$

$$w_{xi} + w_{yi} + w_{zi} = 1 \tag{2.12}$$

$$z_{xi} + z_{yi} + z_{zi} = 1 \tag{2.13}$$

$$l_{xi} + w_{xi} + h_{xi} = 1 \tag{2.14}$$

$$l_{yi} + w_{yi} + z_{yi} = 1 \tag{2.15}$$
$$l_{zi} + w_{zi} + z_{zi} = 1 \tag{2.16}$$

Como já mencionado, o triplo $(h_{xi}, h_{yi}, h_{zi}) = (0, 0, 1)$, implica que $h_{xi} = h_{yi} = 0$, portanto, as equações (2.1) a (2.10) podem ser substituídas, e a formulação fica conforme abaixo:

min $\quad L \cdot W \cdot H - \sum_{i=1}^{N} p_i \cdot q_i \cdot r_i \cdot s_i$

Sujeito a

(Evitar sobreposição de caixas no *container*)

$$x_i + p_i \cdot l_{xi} + q_i \cdot w_{xi} \leq x_k + (1 - c_{ik}) \cdot M, \forall i, k, i < k \tag{2.17}$$
$$x_k + p_k \cdot l_{xk} + q_k \cdot w_{xk} \leq x_i + (1 - d_{ik}) \cdot M, \forall i, k, i < k \tag{2.18}$$
$$y_i + p_i \cdot l_{yi} + q_i \cdot w_{yi} \leq y_k + (1 - a_{ik}) \cdot M, \forall i, k, i < k \tag{2.19}$$
$$y_k + p_k \cdot l_{yk} + q_k \cdot w_{yk} \leq y_i + (1 - b_{ik}) \cdot M, \forall i, k, i < k \tag{2.20}$$
$$z_i + p_i \cdot l_{zi} + q_i \cdot w_{zi} + r_i \cdot h_{zi} \leq z_k + (1 - e_{ik}) \cdot M, \forall i, k, i < k \tag{2.21}$$
$$z_k + p_k \cdot l_{zk} + q_k \cdot w_{zk} + r_k \cdot h_{zk} \leq y_i + (1 - b_{ik}) \cdot M, \forall i, k, i < k \tag{2.22}$$

(Garantir que o par de caixas avaliados nas equações (2.17) a (2.22) esteja no *container*)

$$a_{ik} + b_{ik} + c_{ik} + d_{ik} + e_{ik} + f_{ik} \geq s_i + s_k - 1, \forall i, k, i < k \tag{2.23}$$

(Garantir que o posicionamento das caixas obedeça às limitações físicas dimensionais do *container*)

$$x_i + p_i \cdot l_{xi} + q_i \cdot w_{xi} \leq L + (1 - s_i) \cdot M, \forall i \tag{2.24}$$
$$y_i + p_i \cdot l_{yi} + q_i \cdot w_{yi} \leq W + (1 - s_i) \cdot M, \forall i \tag{2.25}$$
$$z_i + p_i \cdot l_{zi} + q_i \cdot w_{zi} + r_i \cdot h_{zi} \leq H + (1 - s_i) \cdot M, \forall i \tag{2.26}$$

Ainda que o problema tenha sido simplificado com a eliminação de algumas variáveis, continua sendo um problema complexo de solução não trivial, sendo classificado como um problema de classe NP-*Complete* (NP completo) (*Non-deterministic Polynomial-time Complete*), isso significa que o problema não apresenta uma solução polinomial determinística em tempo polinomial segundo Rodrigues em (RODRIGUES, 2005).

Chen, Lee e Shen em (CHEN; LEE; SHEN, 1995) afirmam também que ainda que a formulação matemática leve a uma solução ótima para o problema de carregamento de *container* utilizando um modelo de programação linear inteiro e binário mista, esta não se apresenta como uma solução eficiente para um problema de larga escala, com um número elevado de caixas, já que o número de variáveis e de restrições torna-se muito grande.

A inclusão de novas restrições, tais como peso de carga, valores e outros, aumentaria ainda mais a complexidade do problema, o que, por sua vez, aumentaria a dificuldade de se encontrar uma solução.

Considerando a complexidade da formulação apresentada acima e o tempo para que se encontre a solução com esta formulação, Rodrigues em (RODRIGUES, 2005) afirma que técnicas de otimização heurísticas e metaheurísticas, apresentam-se como métodos mais eficientes, simples e possíveis de serem aplicados em problemas complexos de carregamento de *container*.

2.3 Técnicas de Solução

Grande parte dos problemas práticos de otimização combinatória são classificados como problemas NP-Completo ou problemas NP-Difícil. Esses problemas não podem ser resolvidos por algoritmos simples em tempo polinomial segundo Rodrigues em (RODRIGUES, 2005).

Tomam-se como técnicas para solução de problemas práticos de otimização combinatória métodos exatos e métodos aproximados. Os métodos exatos têm como exemplo o algoritmo de *Branch-and-Bound*, baseado na técnica de dividir o problema em vários problemas menores entre outros. Já os métodos aproximados podem contar com vários tipos de heurísticas, metaheurísticas e técnicas especiais para os representar. Entre as heurísticas pode-se citar a heurística de construção, partição e gulosa para a resolução do problema, já entre as metaheurísticas têm-se os algoritmos genéticos, busca tabu, *simulated annealing* e *GRASP*.

Rodrigues em (RODRIGUES, 2005) resume os métodos de resolução dos problemas clássicos de otimização combinatória no organograma, conforme Figura 2.

O problema de carregamento de *container* é considerado NP-difícil e por isso complexo de ser resolvido matematicamente e deterministicamente, por este motivo têm-se como técnicas de solução do problema de carregamento de *container* heurísticas e metaheurísticas, bem como métodos aproximados especiais.

Segundo Romero e Mantovani em (ROMERO; MANTOVANI, 2004), uma heurística realiza um conjunto de transições através do espaço de busca do problema, iniciando o processo de um ponto do espaço de busca e terminando em um ponto de ótimo local.

O algoritmo heurístico construtivo consiste em ir adicionando, geralmente um a um,

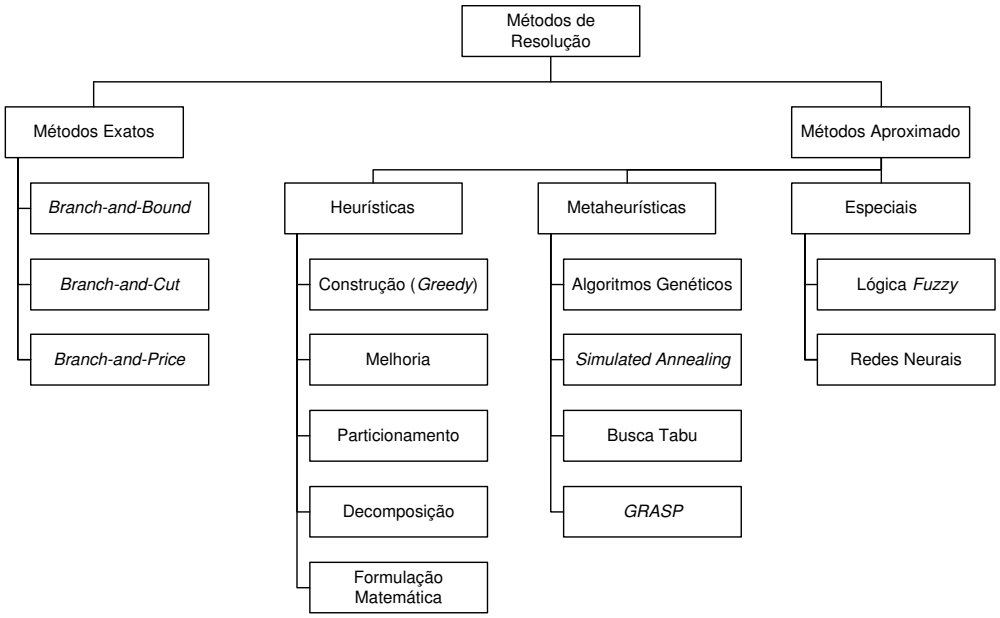

Figura 2: Métodos de Resolução de Problemas de Otimização.

componentes individuais da solução até encontrar uma solução factível. O mais popular entre os algoritmos heurísticos construtivos é o *greedy* ou guloso. O algoritmo heurístico de melhoria se esforça para que a cada passo uma melhora aconteça em sua proposta de solução.

Os algoritmos heurísticos de particionamento e decomposição, segundo Romero e Mantovani em (ROMERO; MANTOVANI, 2004), consistem em decompor o problema em vários problemas menores a fim de simplificar o processo de resolução (algoritmo de particionamento), já o algoritmo de decomposição é ligeiramente diferente e consiste em separar em vários subproblemas independentes do problema geral.

Metaheurísticas de otimização apresentam como característica principal uma componente probabilística, quando comparado às técnicas clássicas, e propõem uma heurística geral que independe da natureza do problema específico a ser tratado. De uma forma geral, essas técnicas conseguem evitar ótimos locais, por realizarem a procura do ótimo em diversas regiões do espaço, utilizando uma aleatoriedade "direcionada" pela heurística codificada na função objetivo.

Algumas das técnicas classificadas como metaheurística, desenvolvidas e utilizadas nos últimos anos são: Algoritmos Genéticos, Busca Tabu, *Simulated Annealing*, *Grasp*, VNS e outros.

Segundo Rodrigues em (RODRIGUES, 2005) as principais metaheurísticas citadas acima

podem ser resumidas conforme abaixo:

- *Simulated Annealing* é uma técnica de busca local probabilística, que se fundamenta em uma analogia com a termodinâmica, simulando resfriamento de um conjunto de átomos aquecidos. A técnica utiliza-se de uma solução inicial qualquer para buscar a solução final. A cada iteração é escolhido o vizinho mais interessante da topologia corrente através de uma lógica baseada no processo de *annealing*.

- Busca Tabu é um método de pesquisa que através de um procedimento adaptativo, utiliza uma estrutura de memória como guia, que por sua vez, armazena os últimos movimentos realizados, e é chamada de lista tabu. Essa lista funciona como uma fila, onde um novo movimento é adicionado e o mais antigo retirado e permite que se decida pela continuidade da exploração do espaço de soluções, mesmo na ausência de movimentos de melhora, evitando que haja retorno a um ótimo local previamente visitado.

- Algoritmos Genéticos, por sua vez, utilizam a analogia baseada no processo natural da evolução das espécies, onde as características de uma possível solução são medidas por uma função de aptidão, determinando assim que os indivíduos mais aptos têm maiores chances de sobreviver e reproduzir. Para reproduzir ou criar uma nova geração, novamente em analogia com a biologia, operações de recombinação, mutação e seleção são utilizadas. Quando se atinge um critério de parada, o método termina e o melhor indivíduo (cromossomo) que apresenta características mais adequadas à solução esperada (maximização ou minimização) é selecionado como solução ótima ou quase ótima.

- *GRASP*: *Greedy Randomized Adaptive Search Procedure* ou Procedimento de Busca Adaptativa Gulosa e Aleatória é um método iterativo de duas etapas. A primeira etapa consiste na construção de soluções, elemento a elemento, e em uma segunda etapa, faz-se uma busca local para determinar um ótimo local na vizinhança de uma solução construída. O processo de seleção de soluções é baseado em uma função adaptativa gulosa. Em cada iteração são geradas diferentes soluções.

2.4 Análise do Estado da Arte em Carregamento de *Container*

O problema abordado no trabalho, como foi discutido na seção anterior, é considerado NP-difícil e por isso complexo de ser resolvido matemática e deterministicamente, justificando o emprego de heurísticas e metaheurísticas para sua resolução.

Não existe um algoritmo geral que garanta encontrar a solução ótima para o problema de carregamento de *container*; o que existe são várias heurísticas e metaheurísticas propostas na literatura que são utilizadas como técnicas de solução para o problema de carregamento de *container*. O único modo de determinar qual o conjunto de técnicas alternativas que encontra a melhor solução para um problema específico é geralmente experimentar trocas de técnicas entre as heurísticas já testadas.

Realizou-se um estudo amplo sobre as técnicas de soluções e variantes que estão sendo aplicadas para resolver o problema de carregamento de *container* entre estas técnicas se encontram: (DAVIES; BISCHOFF, 1999), (GEHRING; MENSCHNER; MEYER, 1990), (GOMES; OLIVEIRA, 2002), (ELEY, 2002). Algumas heurísticas, senão as mais importantes, serão descritas a seguir.

Segundo Bischoff e Marriott em (BISCHOFF; MARRIOTT, 1990) existem duas técnicas heurísticas que serviram de base para outras heurísticas propostas na literatura, a saber, a busca de George e Robinson em (GEORGE; ROBINSON, 1980) e a busca de Bischoff e técnica de Dowsland em (BISCHOFF; DOWSLAND, 1982). Estas heurísticas consideram o problema *Knapsack* e trabalham com instâncias de carga fortemente heterogêneas.

A busca de George e Robinson em (GEORGE; ROBINSON, 1980) trabalha com uma heurística construtiva onde o procedimento de carregar *container* é realizado em camadas. Eles propõem o preenchimento através de camadas preenchendo primeiro a dimensão da largura do *container* (preenchimento na horizontal primeiramente) e posteriormente o preenchimento da altura do *container* (preenchimento vertical), fazem com que o preenchimento de uma camada não invada o espaço da próxima camada que estaria a sua frente (preenchimento do fundo do *container* para frente). Os itens a serem carregados foram chamados por George e Robinson em (GEORGE; ROBINSON, 1980) de "caixas" e o termo é empregado até hoje, elas são divididas em duas classes: tipo aberto e sem abrir, com a diferença entre estes tipos que caixas abertas foram usadas e caixas sem abrir não foram utilizadas ainda no carregamento do *container*. Tipos abertos na verdade são as caixas que tem preferência e deveriam ser usadas para começar uma camada. Esta primeira caixa

em uma camada é crucial, ela determina a profundidade que será seguida pelas outras caixas que farão parte da camada. Procura-se utilizar em uma camada, caixas do mesmo tipo para evitar que uma caixa invada o espaço correspondente ao da próxima camada. Outros tipos de caixas só são considerados para preencher espaços deixados dentro da camada atual ou para preencher buracos que combinam com o volume da caixa em camadas anteriores.

No início do procedimento não existe tipo de caixa aberta. Sempre que esta situação acontece, a caixa escolhida para começar a próxima camada é determinada com base nos critérios de prioridade no qual o critério primário é o tamanho da menor dimensão de uma caixa. As caixas são ordenadas pelo maior tamanho da menor dimensão entre as caixas. O maior tamanho entre os tamanhos da menor dimensão é escolhido, a razão é simples, aquela caixa que tem o maior tamanho entre os tamanhos da menor dimensão de uma caixa pode ser mais difícil de ser acomodada em um *container* no procedimento de carregamento. No caso de empate deste primeiro critério, a escolha é feita com base no número de caixas do mesmo tipo particular, uma quantidade grande da mesma caixa é mais provável que preencha uma camada. O segundo desempate proposto é o tamanho da maior dimensão entre as caixas empatadas, se este é grande (longo), a caixa pode ser desajeitada para ser acomodada no *container* e o carregamento pode ser facilitado tentando acomodar tal caixa mais cedo.

A busca de George e Robinson em (GEORGE; ROBINSON, 1980) segundo Bischoff e Marriott em (BISCHOFF; MARRIOTT, 1990), traz um parâmetro k que é usado para restringir o tamanho da profundidade de uma camada. Nenhuma razão para escolher k é determinada pelos autores, mas tentativas levaram a crer que isso pode ter um efeito significante na eficiência do carregamento do *container*.

Outra heurística sofisticada para o problema de carregamento de *container* foi proposta por Bischoff e Dowsland em (BISCHOFF; DOWSLAND, 1982). A descrição desta heurística segue abaixo.

A busca, segundo Bischoff e a técnica de Dowsland em (BISCHOFF; DOWSLAND, 1982), é uma busca que também está baseada no princípio de preenchimento do *container* construindo camadas começando com a largura. Porém, há duas diferenças principais em relação ao procedimento de George e Robinson em (GEORGE; ROBINSON, 1980): Primeiramente, cada camada só é construída de um único tipo de caixa; e secundariamente, o arranjo de caixas dentro de uma camada é determinado por uma heurística de empacotamento bidimensional. A busca de Bischoff e Dowsland tem o objetivo de maximizar

a utilização da área de uma camada (isto é, do retângulo formado pela largura e altura do *container*). Depois de estabelecer as diferenças entre as propostas, resta dizer que a heurística foi proposta originalmente como uma busca para calcular padrões de plano (*layout*) segundo Bischoff e Marriott em (BISCHOFF; MARRIOTT, 1990).

O preenchimento do espaço de uma camada não é considerado na heurística proposta, pois a camada é composta por caixas do mesmo tipo e a ordem na qual as caixas são colocadas dentro das camadas não tem nenhuma influência na eficiência do carregamento alcançado.

O critério usado para decidir a dimensão de profundidade de uma camada, porém, é de crucial importância. Cada um dos três lados de uma caixa é examinado trocando a caixa de posição e verificando qual dimensão da caixa se enquadra com uma profundidade potencial para uma camada. Com esta dimensão fixada, o número máximo de caixas que podem ser acomodadas na camada é então determinada por meio de uma rotina de empacotamento bidimensional.

Em outras palavras, se a largura e a altura do *container* são denotadas por W e H, e as três dimensões da caixa são w (profundidade), l (largura) e h (altura), como sendo w considerado atualmente a profundidade da camada, é calculado o número de retângulos $l \times h$ que ajusta no retângulo $W \times H$. Se o resultado é menor que o número de caixas daquele tipo particular disponíveis para serem carregadas, uma camada cheia não pode ser formada e a dimensão de profundidade referida é conseqüentemente rejeitada. Se, por outro lado, há mais que uma possível orientação de profundidade que complete uma camada, uma escolha necessita ser feita entre elas.

Um critério para fazer esta escolha é a porcentagem de preenchimento da camada (em termos de volume ou área). Seguindo outra razão, porém, poderia ser desejável tentar maximizar o número de caixas que são acomodadas em camadas completas.

Em alguma fase da heurística é alcançado um ponto onde ou todas as caixas são alocadas em camadas completas ou, como é mais provável, não é possível formar mais adiante camadas completas de qualquer tipo de caixa. Neste caso as caixas restantes são alocadas usando a busca de George e Robinson (GEORGE; ROBINSON, 1980).

Uma outra heurística encontrada na literatura, baseada na busca citada acima foi a heurística proposta por Pisinger em (PISINGER, 2002) para a resolução do problema de carregamento de *container*.

A heurística é baseada na busca chamada por ele de “Construindo-Camadas”, onde é

proposto que o *container* seja decomposto em camadas que novamente são divididas em filas (se o preenchimento começar na horizontal) ou pilhas (se o preenchimento começar na vertical). A profundidade das camadas é escolhida depois de um estudo detalhado para ver qual seria a melhor profundidade para a camada e então todas as caixas que fizerem parte dela teriam que respeitar sua profundidade, não podendo exceder a medida imposta pela primeira caixa alocada nesta camada. O mesmo acontece com o preenchimento das filas ou pilhas. As caixas não devem exceder a altura da primeira caixa alocada na camada com o preenchimento começando na horizontal e também não deve exceder a largura quando o preenchimento começar na vertical. O empacotamento das filas ou pilhas pode ser formulado e resolvido otimamente como um Problema *Knapsack*.

A profundidade da camada como também a espessura de cada fila, segundo Pisinger em (PISINGER, 2002), é decidida pela técnica de *Branch-and-Bound* onde para cada camada é explorado somente um subconjunto de tamanhos para sua profundidade.

São apresentadas várias regras de posição para a seleção das profundidades de camadas mais promissoras e larguras de filas. A melhor regra de posição é então usada em estudos computacionais envolvendo instâncias com carga homogênea e heterogênea.

Outra proposta estudada foi a de Gehring e Bortfeldt em (GEHRING; BORTFELDT, 1997), onde estes introduziram o conceito de torres. A proposta dos pesquisadores foi a de utilizar o conceito de torres junto com uma metaheurística comum em pesquisa operacional, o algoritmo genético, dando origem a um algoritmo genético híbrido. Primeiramente gera-se as torres que posteriormente irão compor os cromossomos da população inicial.

Cada torre é gerada escolhendo aleatoriamente uma caixa, cujo nome adotado pelos autores para esta primeira caixa é “caixa base” e após este processo, procura-se por outra caixa disponível que tenha o maior volume possível, mas que sua área seja igual ou menor que a área da caixa base, isto acontece para que nenhuma torre invada o espaço da outra quando colocadas lado a lado no *container*. O processo de alocação de caixas se repete até que não seja mais possível alocar caixas em uma torre. Várias torres são formadas através deste processo.

A população inicial é gerada aleatoriamente utilizando as torres para compor cada cromossomo. Estes cromossomos passam por uma etapa de decomposição, onde as torres são alocadas no *container* usando regras de alocação. As regras de alocação consideram o comprimento do *container* infinito, portanto, pode-se alocar todas as torres no *container* e após o carregamento com todas as torres realiza-se uma verificação das torres que realmente estão por inteiro dentro do *container*, somente com estas torres é que

cada cromossomo é avaliado. O algoritmo genético híbrido proposto pelos autores considera que somente um operador genético (recombinação ou mutação) seja aplicado em um cromossomo a cada iteração.

Mais tarde em 2001, estes mesmos autores Bortfeldt e Gehring, relatam outra proposta estudada por eles. Bortfeldt e Gehring em (BORTFELDT; GEHRING, 2001) utilizaram como técnica de solução para o problema do *container* outro algoritmo genético. Estes pesquisadores propuseram um algoritmo genético híbrido para o problema de carregamento de *container* com caixas de tamanhos diferentes e um único *container* para carregar, ou seja, o problema considerado é o *Knapsack* com carga fortemente heterogênea. O algoritmo genético híbrido, segundo Bortfeldt e Gehring em (BORTFELDT; GEHRING, 2001), é usado para gerar plantas de armazenamento com uma estrutura de tipos de camada, ou seja, através do algoritmo genético híbrido é definido como as camadas são divididas para preencher o *container*, esta divisão é feita no espaço bidimensional, depois de aplicados os operadores genéticos, recombinação e mutação, escolhe-se a melhor divisão do espaço bidimensional do *container*. As camadas são preenchidas verticalmente, cada camada contém uma ou mais caixas que não se projetam em camadas adjacentes. As caixas usadas não são sobrepostas em camadas paralelas, tornando cada camada independente.

No início do procedimento, uma heurística básica é usada para fornecer plantas de armazenamento de boa qualidade para compor a população.

Também no trabalho de Rodrigues (RODRIGUES, 2005) aparece uma proposta de solução baseada no algoritmo genético proposto por Bortfeldt e Gehring em (BORTFELDT; GEHRING, 2001). Nesta versão o algoritmo genético é adequado às características e ao contexto do problema real, com possibilidade de escolha entre dois métodos de preenchimento: Lateral, Superior e Frontal e o outro método de preenchimento: Superior, Lateral e Frontal.

Em outras palavras, Rodrigues em (RODRIGUES, 2005) propõe dois métodos de preenchimento baseados em pesquisas já realizadas, unificando propostas e acrescentando a maneira de avaliar os indivíduos da população corrente, o objetivo de maximizar o valor monetário final da carga, considerando um limite dado. É associado a cada item que será carregado um valor monetário. Ele propõe que a população seja gerada aleatoriamente e após a aplicação dos operadores genéticos, recombinação e mutação, seja aplicado na população um processo chamado por ele de "Decodificação para Obtenção do Padrão de Carregamento". As caixas são alocadas no *container* na seqüência que aparecem nos genes dos cromossomos da população. O processo decorre de um método sistemático de

preenchimento de subespaços até que todo o espaço do *container* esteja preenchido ou que não haja mais caixas que possam atender às restrições dimensionais do espaço restante.

O problema considerado por Rodrigues em (RODRIGUES, 2005) é o Problema *Knapsack*, e os testes foram realizados em instâncias fortemente homogêneas e fortemente heterogêneas.

Outra proposta encontrada na literatura é a heurística para a resolução de uma variante do problema de carregamento de *container*, ele é conhecido como Problema Tridimensional *Bin Packing*, a heurística é proposta por Lodi, Martello e Vigo em (LODI; MARTELLO; VIGO, 2002) e consiste em empacotar caixas através de camadas como vimos nas propostas acima, mas trabalhando no preenchimento da camada de maneira diferente.

A heurística funciona em duas fases, e foi denominada pelos autores de "Primeiro Altura - Segundo Área (HA)", isto porque na primeira fase da heurística, as caixas que serão empacotadas são ordenadas de maneira decrescente pelo tamanho de sua altura. As caixas são agrupadas em lotes de maneira que as alturas tenham o menor desnível possível e as caixas deste grupo são arranjadas no *container*.

Logo em seguida, as caixas são arranjadas no *container* pela ordem decrescente da área que estas caixas apresentam. O melhor resultado encontrado comparando as duas fases é escolhido e assim a primeira camada do *container* é carregada. Existe uma terceira fase onde é considerado que as caixas podem ser giradas para depois serem aplicadas as duas primeiras fases.

O ideal seria que a primeira caixa utilizada no empacotamento ocupasse todo o espaço da primeira camada, ou seja, a base da primeira camada coincidisse com a base da caixa e que a altura da primeira camada coincidisse com a altura da caixa. Se a altura da primeira caixa utilizada não coincide com a altura da primeira camada, a altura da camada subseqüente considera como altura limite à altura da caixa mais alta empacotada na camada anterior.

Para produzir um "efetivo" empacotamento em camadas, a heurística proposta por Lodi, Martello e Vigo em (LODI; MARTELLO; VIGO, 2002) considera dois pontos principais:

(*i*) obter um "bom preenchimento vertical" empacotando caixas semelhantes na mesma camada;

(*ii*) obter um "bom preenchimento horizontal" resolvendo efetivamente o empacotamento bidimensional das caixas em relação às bases das camadas.

Existem outros problemas encontrados na literatura que tem relações próximas com o problema do carregamento de *container*. Como podem ser vistos em (FRASER; GEORGE, 1994), (DOWSLAND; DOWSLAND, 1992), (HOLTHAUS, 2002), (HADJICONSTANTINOU; CHRISTOFIDES, 1995), (HAESSLER; SWEENEY, 1991), (FARLEY, 1990), (GOULIMIS, 1990). Pode-se citar aqui alguns, como exemplo: O Problema Bidimensional de Corte de Estoque e Empacotamento Bidimensional com formas retangulares. Por esta razão também foram abordadas propostas heurísticas para resolução destes tipos de problemas.

Lodi, Martello e Monaci em (LODI; MARTELLO; MONACI, 2002) realizaram uma pesquisa sobre o problema bidimensional de empacotamento de caixas e filas que estão rigidamente relacionado com o problema de carregamento de *container*. As definições dos problemas segundo os autores de (LODI; MARTELLO; MONACI, 2002) estão apontadas a seguir.

Em várias aplicações industriais é necessário alocar um conjunto de artigos retangulares em uma unidade maior de armazenamento, que também é retangular, unificando os artigos e minimizando o desperdício. Em indústrias de madeira ou vidro, componentes retangulares têm que ser cortados de folhas grandes, de sua matéria prima.

Em contextos de armazenamento de mercadorias, artigos têm que ser colocados em estantes. Em páginas de jornal, reportagens e anúncios têm que ser organizados nas páginas.

Nestas aplicações, as unidades de estoque unificadas são retângulos, e uma função objetivo comum é empacotar todos os artigos pedidos em um número mínimo de unidades de estoque: as resultantes dos problemas de otimização são conhecidas na literatura como problemas bidimensionais de empacotamento de caixas.

Em outros contextos, como papel ou indústrias de pano, tem-se uma única unidade unificada ao invés de várias (um rolo da matéria prima), e o objetivo é obter os artigos usando o comprimento de rolo mínimo: os problemas estão sendo referenciados como problema bidimensional de empacotamento de filas.

Estes problemas fazem parte de um subconjunto de problemas bidimensionais clássicos de empacotamento e de cortes, todos eles considerados NP-difícil.

Os dois problemas têm uma relação forte em quase todas as técnicas de buscas algorítmicas para sua solução, bem como uma relação estreita com o problema de carregamento de *container*, que é um problema tridimensional. A seguir apresentam-se algumas técnicas de solução encontradas na literatura para a resolução dos problemas relacionados

acima.

Wu e seus colegas de pesquisa em (WU et al., 2002) escreveram sobre o assunto e propuseram uma heurística para um dos problemas bidimensionais citados acima, o problema de empacotamento de caixas. A heurística proposta é baseada no senso comum e denominada por Wu e outros de uma heurística "quase-humana", o princípio de "Primeiro Menor Flexibilidade". A estratégia de busca foi desenvolvida por um princípio filosófico inspirado a 1000 anos atrás, conhecida por profissionais antigos chineses. Por milhares de anos, pedreiros chineses utilizaram a seguinte "regra do manusear" no trabalho de empacotamento. São atribuídas cores a partes de um espaço a ser preenchido. Dourados são os cantos; prateados são os lados; e palhas são os espaços centrais. Existe uma ordem de prioridade para preencher os espaços vazios dentro da área de trabalho. Sugere-se que deveriam ser preenchidos primeiro os cantos "vazios" dentro da área limitada. Inspirados por esta estratégia, os autores em (WU et al., 2002) propuseram assim um chamado Princípio do Primeiro Menos Flexível (LFF). A meta do problema de empacotamento de retângulos é empacotar um conjunto de determinados retângulos com tamanhos arbitrários em um espaço de trabalho tão pequeno quanto possível. Usando o princípio de "Primeiro Menos Flexibilidade", o espaço com "menos flexibilidade" do retângulo deveria ser ocupado mais cedo. Na realidade, a regra cantos dourados, lados prateados e o centro palha concorda com o princípio de Primeiro Menor Flexibilidade lidando com espaços vazios.

A flexibilidade de estados espaciais tem três graus diferentes:

(*i*) um canto tem a menor flexibilidade;

(*ii*) um lado tem flexibilidade média e;

(*iii*) uma área nula central tem a flexibilidade mais alta.

Um canto é fixo (limitado) por dois lados e um objeto empacotado ao canto, não pode mover-se em nenhuma direção.

Um objeto empacotado em um lado pode mover ao longo da linha limitante com somente um lado fixado. O grau de liberdade ou flexibilidade é unidimensional.

Finalmente, um objeto empacotado em uma área nula pode mover-se livremente para qualquer direção sem lado fixo. O grau de liberdade ou flexibilidade é bidimensional.

A ordem de flexibilidade segundo Wu e outros pesquisadores em (WU et al., 2002) pode ser derivada como segue:

flexibilidade de canto < flexibilidade de lado < flexibilidade de área nula.

Além disso, o princípio pode ser estendido para selecionar os retângulos candidatos a serem empacotados.

Segundo os autores deve-se empacotar uma caixa menos flexível (maior, mais longa) primeiro em um canto, para ter um resultado de empacotamento melhor. Além disso, é mais favorável ocupar mais de um canto. Deve-se também evitar caixas nas áreas nulas, que não são adjacentes a qualquer outra caixa. É considerado que tal ato não é razoável, desde que as caixas empacotadas desta maneira podem ser obstáculos para futuros empacotamentos.

Portanto a heurística proposta por Wu e outros em (WU et al., 2002) combina todos os candidatos com um canto da área de trabalho e em todas as posições possíveis, ordena esta lista por ordem lexicográfica e vai preenchendo a área de trabalho começando do canto inferior esquerdo, utilizando a primeira caixa da lista, depois é calculado o valor da função de aptidão de custo utilizada pelos autores para avaliar o preenchimento e é gerado logo em seguida um pseudocanto, onde o preenchimento começa dele. O preenchimento segue o mesmo processo alocando as caixas primeiramente da esquerda para a direta e de baixo para cima. O processo se repete até que todos os retângulos sejam utilizados ou até que não haja mais nenhuma área vazia para ajustar caixas.

A proposta de Morabito e Arenales em (MORABITO; ARENALES, 1992) é diferenciada por trazer um método de aproximação especial para a resolução do problema bidimensional de corte por guilhotina. O método foi chamado por eles de "E/OU Gráficos" e é baseado em Inteligência Artificial, mais especificamente em Redes Neurais. A aproximação consiste em um processo de busca de gráficos dirigida, na qual cada nó representa um subproblema e cada arco representa uma relação entre os nós.

É possível definir um problema como uma busca em um gráfico, especificando o nó inicial, os nós finais e as regras que definem os movimentos legais.

No problema bidimensional de corte, a folha retangular de uma matéria prima é representada pelo nó inicial e os pedaços retangulares pequenos são representados pelos nós finais. As regras que definem os movimentos legais são os possíveis cortes em um retângulo (cortes de guilhotina, cortes artificiais que deixam os retângulos intactos, e regras que evitam duplicação de padrões como corte ordenado, restrições de simetria, e outros). Então, um subconjunto de cortes pode ser aplicado à folha retangular, correspondendo ao grau de ramificação do nó inicial. Depois de cada corte, os dois retângulos sucessivos

correspondem aos nós intermediários. Podem ser aplicados outros subconjuntos de cortes a retângulos intermediários e assim por diante, até os pedaços retangulares determinados se encontrarem como os nós finais.

No artigo escrito por Morabito e Arenales, (MORABITO; ARENALES, 1992), a proposta é definida ainda como um tipo de estrutura útil por representar a solução de problemas que podem ser resolvidos decompondo-os em um conjunto de problemas menores. Esta decomposição, ou redução gera os arcos. Um arco E pode apontar para um número qualquer de nós sucessores, todos os quais devem ser resolvidos para que o arco aponte uma solução. Da mesma maneira que um gráfico OU, vários arcos podem emergir de um único nó, enquanto indicando uma variedade de caminhos nos quais o problema original poderia ser resolvido.

Existem tantas outras propostas na literatura para a resolução do problema de carregamento de *container* e problemas relacionados que também poderiam ser descritas aqui, tais como (MOHANTY; MATHUR; IVANCIC, 1994), (BISH, 2003), (CARVALHO, 2002), (ELEY, 2002), (ARMBRUSTER, 2002), (LINS; LINS; MORABITO, 2002), mas a pequena amostra acima é suficiente para os objetivos deste trabalho.

3 *Heurística Proposta para o Problema de Carregamento de Container*

Este capítulo relata a heurística proposta neste trabalho, para a solução do problema de carregamento de *container*, tendo como objetivo contribuir com uma proposta de solução do problema de otimização aqui exposto de modo diferenciado dos encontrados na literatura.

A heurística é detalhadamente exibida, mostrando passo a passo os procedimentos que realiza para a obtenção da solução do problema. Os testes que foram simulados e seus resultados também constam no conteúdo deste capítulo.

O capítulo inicia-se com uma introdução abordando conceitos de heurística; segue com a apresentação da heurística proposta neste trabalho, para a resolução do carregamento de *container*; logo em seguida são relatados os testes que foram realizados, a simulação do carregamento de *container* usando a heurística proposta e comentários sobre como o algoritmo heurístico reagiu; finalizando com uma análise dos resultados obtidos.

3.1 Introdução

Uma heurística segundo Romero e Mantovani em (ROMERO; MANTOVANI, 2004) é uma técnica de otimização que através de passos muito bem definidos encontra uma solução de boa qualidade para um problema complexo.

Como se pode observar na definição de heurística é perfeitamente justificável o emprego de um algoritmo heurístico para a resolução do problema de carregamento de *container*.

Os comentários do capítulo anterior deixam claro que o problema estudado neste trabalho é de extrema complexidade para ser resolvido matematicamente, sendo conside-

rado NP-Difícil, ou seja, problemas para os quais não se conhecem algoritmos de esforço polinomial para encontrar sua solução ótima.

Portando, a heurística proposta neste trabalho procede de maneira sistemática, como manda a definição de heurísticas.

Depois de realizado um amplo estudo sobre as heurísticas que estão sendo empregadas na resolução do problema de carregamento de *container*, o trabalho propõe uma nova heurística para encontrar soluções de boa qualidade para o problema já especificado.

A heurística de um modo geral trabalha dividindo o *container* em quatro partes onde a Parte Principal ou o Corpo Principal do *Container* é formado por caixas de dimensões similares e as outras partes do *container* são preenchidas nesta ordem: primeiro a parte chamada de Espaço Lateral Residual; em seguida a parte denominada Espaço Superior Residual e por fim é preenchido o Espaço Frontal Residual.

O algoritmo tem a preocupação de colocar as caixas mais leves em lugares sem risco de esmagamento por cargas mais pesadas. A heurística proposta será detalhada na seção a seguir.

3.2 Algoritmo Heurístico Proposto

O algoritmo heurístico proposto trabalha de maneira sistemática, onde as caixas que estão disponíveis para o carregamento do *container* são de vários tipos. A quantidade de caixas de cada tipo será contabilizada no início da aplicação da heurística, e caixas com mesmas dimensões, mas de tipos diferentes serão contabilizadas como caixas de um único tipo. Por exemplo, se for encontrada caixas com dimensões iguais, mas com pesos diferentes serão contabilizadas como caixas do mesmo tipo.

O algoritmo heurístico tem como principal objetivo colocar o maior número de caixas dentro do *container* preenchendo seu volume até que as caixas disponíveis para o carregamento se acabem ou até que as caixas que ainda se encontram disponíveis não satisfaçam às restrições de dimensão e volume do *container*.

A ordem que as caixas serão carregadas no *container* ao final da aplicação da heurística será chamada de padrão de carregamento. Este padrão de carregamento será armazenado em um vetor onde o índice do vetor indica a ordem que a caixa será alocada no *container* e o conteúdo do vetor naquele índice (posição) mostra qual caixa será alocada no *container* naquela ordem.

O cálculo do volume ocupado pela carga carregada, além dos cálculos do peso da carga, do equilíbrio e valor da carga carregada será realizado após o processo heurístico aplicado no problema, processo que será chamado de decomposição dos espaços do *container*.

A heurística proposta por este trabalho possibilita através da decomposição dos espaços do *container* o preenchimento do mesmo dividindo-o em quatro partes.

São elas: Parte Principal ou Corpo Principal do *Container*, Espaço Lateral Residual, Espaço Superior Residual e Espaço Frontal Residual.

A Figura 3 abaixo ilustra a divisão do *container* proposta pela heurística.

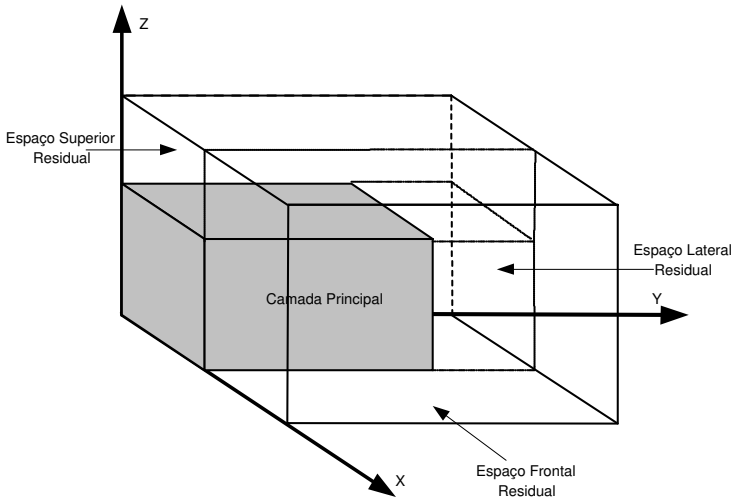

Figura 3: Divisão do *Container* em Parte Principal ou Corpo Principal do *Container*, Espaço Lateral Residual, Espaço Superior Residual e Espaço Frontal Residual.

O preenchimento dos quatro espaços do *container* obedece a uma ordem de preenchimento partindo primeiramente da lateral esquerda para a direita, preenchendo o espaço horizontal, logo após segue o preenchimento do espaço vertical de baixo para cima e em seguida do fundo do *container* para frente.

A heurística procura também, alocar caixas mais pesadas abaixo e em camadas específicas, onde o risco de esmagamento da carga por outra mais pesada se anula.

Todo o processo heurístico realizado para a obtenção da solução do problema de carregamento do *container* é apresentado de maneira simplificada no fluxograma da Figura 4.

Em seguida é descrita detalhadamente a codificação do padrão de carregamento que será obtido ao final do processo heurístico aqui estudado. Além da codificação do padrão

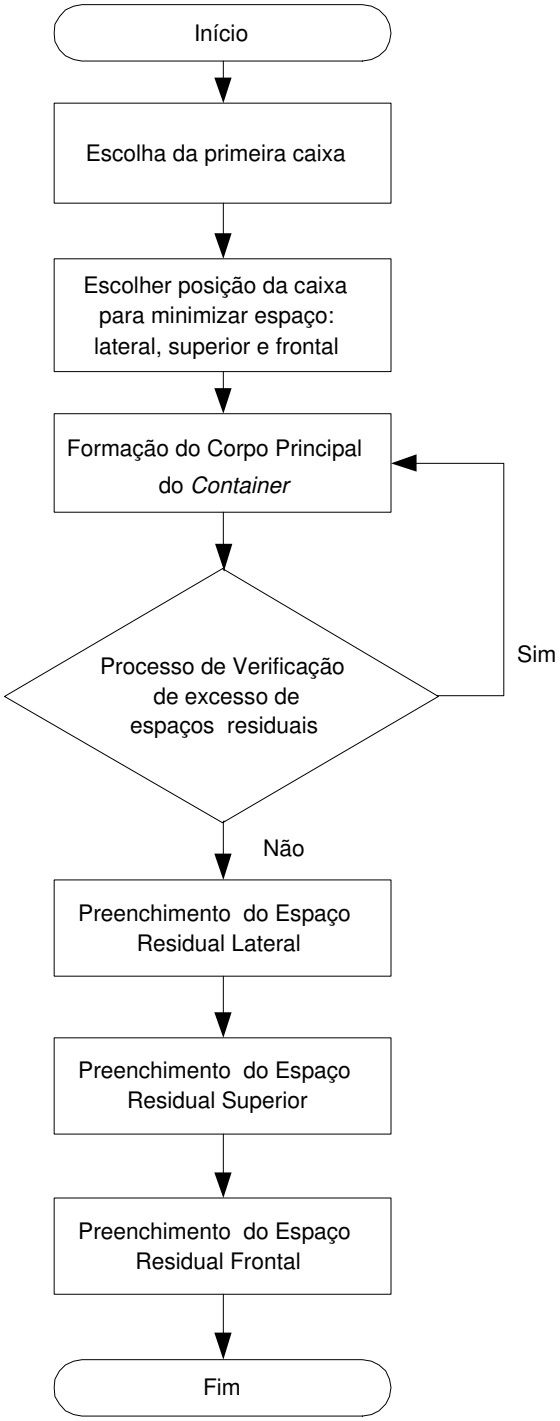

Figura 4: Fluxograma do Algoritmo Heurístico proposto para a determinação da solução do problema de carregamento de *container*.

de carregamento, será esclarecido como o padrão de carregamento deve ser avaliado, para fornecer a informação de que a solução encontrada é de boa qualidade ou simplesmente encontrou-se uma solução menos favorável. É explicado ainda, como o padrão de carregamento é formado, ou seja, como se dá a decomposição dos espaços do *container*.

3.2.1 Codificação do Padrão de Carregamento

A representação do padrão de carregamento (P) é formada por índices de caixas b_i: $P = b_1, b_2, b_3, ..., b_n$, onde $i = 1, ..., n$ são índices das caixas.

A seqüência b_1, b_2, ..., b_n representa a ordem com que as caixas devem ser posicionadas no *container*, seguindo as regras de preenchimento previamente definidas.

A seguir, a Figura 5 mostra como seria um possível padrão de carregamento, com quatro caixas disponíveis.

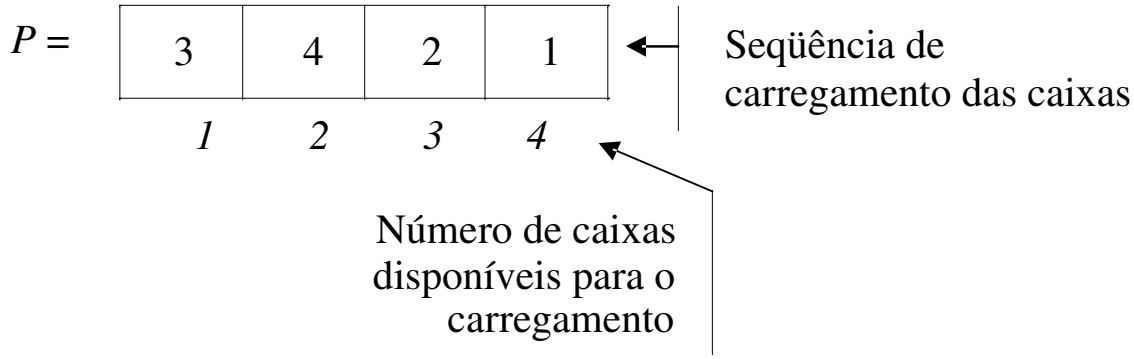

Figura 5: Exemplo de Padrão de Carregamento.

Os índices das caixas também representam os P tipos de caixas, pois se temos x tipos de caixas, e cada tipo de caixa i tem m_i caixas representando-o, então as caixas com índices de 1 a m_1 representam as caixas do tipo 1, as caixas com índices de $m_1 + 1$ a $m_1 + m_2$ representam as caixas do tipo 2 e assim sucessivamente, até completar os índices das caixas disponíveis para o carregamento.

A Tabela 1 logo a seguir, traz de maneira simples e didática como identificar o tipo de cada caixa.

Além do vetor onde é representado o padrão de carregamento, a codificação do problema estudado necessita de um vetor auxiliar onde é sinalizado como as caixas foram rotacionadas e alocadas no *container*. Para tanto, a variação da orientação da caixa é identificada através dos seguintes índices:

Tabela 1: Exemplo de Identificação de Tipo de caixa.

Tipo	Qtde.Caixas	Intervalo de Tipos de Caixas
1	2	1 a 2
2	3	3 a 5
3	5	6 a 10

- Se a largura da caixa está apoiada na base, e a altura e a profundidade não variam, então o índice que representa esta posição de caixa é igual a 1.
- Se a altura da caixa está apoiada na base, e a largura da caixa no lugar da altura, com profundidade sem variar, então o índice que representa esta posição de caixa é igual a 2.
- Se a profundidade está apoiada na base, a altura sem variar e a largura da caixa está no lugar da profundidade, então o índice que representa esta posição de caixa é igual a 3.
- Se a largura está apoiada na base, e a altura e a profundidade trocam de lugar entre si, então o índice que representa esta posição de caixa é igual a 4.
- Se a altura está apoiada na base, a profundidade está no lugar da altura e a largura no lugar da profundidade, então o índice que representa esta posição de caixa é igual a 5.
- Se a profundidade está apoiada na base, a largura está no lugar da altura e a altura está no lugar da profundidade, então o índice que representa esta posição de caixa é igual a 6.

Assim depois de aplicar o processo de decomposição de espaços é gerado um outro vetor complementar ao padrão de carregamento onde é identificado de que maneira a caixa foi alocada no *container*.

A Figura 7 exemplifica como ficaria o vetor complementar ao padrão de carregamento quando analisado o carregamento do *container* da Figura 6.

3.2.2 Cálculos para a avaliação do Padrão de Carregamento

Temos duas propostas para a avaliação do padrão de carregamento. São elas: a avaliação com o objetivo de maximizar somente o volume da carga alocada no *container*

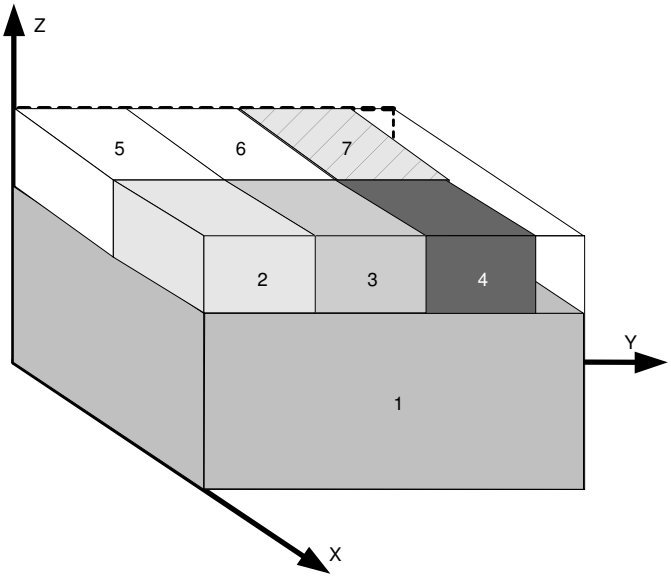

Figura 6: *Container* carregado após o processo heurístico.

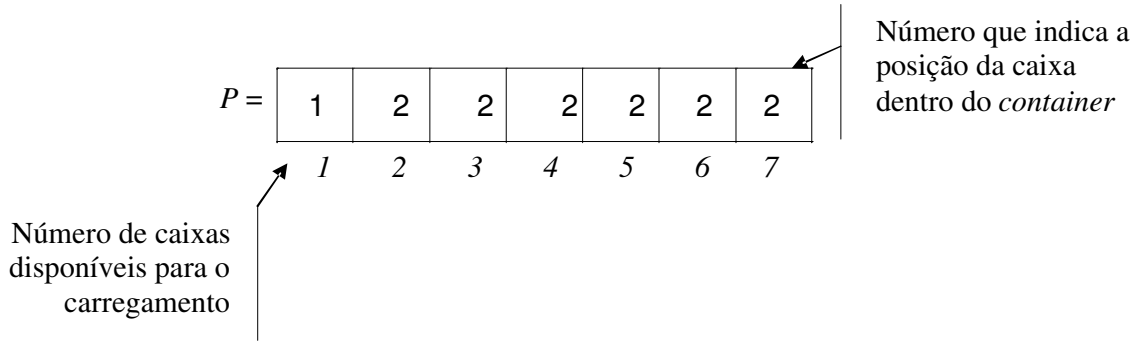

Figura 7: Vetor complementar ao padrão de carregamento do *container* analisado.

(avaliação mais comum na literatura) e a avaliação com o objetivo de maximizar volume, peso, centro de gravidade e valor da carga carregada. Este segundo tipo de avaliação foi proposto por Rodrigues em (RODRIGUES, 2005) e será aplicada no trabalho para fins de comparação.

3.2.2.1 Maximizar somente volume da carga

O cálculo mais comum no meio científico para a avaliação do padrão de carregamento é calcular apenas a porcentagem do volume ocupado pela carga alocada no *container* em relação ao volume do *container*. O objetivo é obter o valor máximo de 100% na ocupação do volume do *container*, mas como muitas vezes isso não é possível, procura-se o valor mais próximo.

A função usada para maximizar o volume da carga está descrita logo abaixo.

$$\omega = \frac{\sum_{i=1}^{k} VL_{bi}}{VL_c} \times 100 \tag{3.1}$$

Em que VL_{bi} é o volume de cada caixa carregada, i determina o índice da caixa, VL_c é o volume disponibilizado pelo *container* e k o número de caixas alocadas no *container*.

3.2.2.2 Maximizar volume, peso, centro de gravidade e valor da carga

Os cálculos para a avaliação do padrão de carregamento após a aplicação do processo de decomposição dos espaços são realizados de maneira separada, pois serão calculados vários interesses e só depois os resultados serão reunidos para um valor final. Segundo Rodrigues em (RODRIGUES, 2005) avalia-se a configuração da carga carregada conforme o volume ocupado, o peso dos produtos carregados, o centro de gravidade e o valor total dos produtos. Cada um desses aspectos avaliados demonstra o atendimento do padrão de carregamento em relação às restrições impostas pelo problema: volume disponível, peso permitido de carregamento, centro de gravidade da carga e valor máximo das cargas.

Cada sub-função para a avaliação do padrão de carregamento possui um peso associado que prioriza os objetivos mais importantes em relação a outros objetivos.

O objetivo principal é obter o valor máximo da função geral que rege este problema. A função geral está descrita logo abaixo.

$$\omega = \frac{k_1 \times VL + k_2 \times P + k_3 \times G + k_4 \times V}{k_1 + k_2 + k_3 + k_4} \tag{3.2}$$

As sub-funções são as seguintes:

VL - Sub-função para o cálculo do volume;

P - Sub-função para o cálculo do peso;

G - Sub-função para o cálculo do centro de gravidade e;

V - Sub-função para o cálculo do máximo valor dos produtos carregados.

Os pesos são determinados por k_1, k_2, k_3 e k_4, que estão associados às sub-funções do volume (VL), do peso (P), do centro de gravidade (G) e do valor da carga (V), respectivamente.

As sub-funções são encontradas em Rodrigues (RODRIGUES, 2005) e serão descritas abaixo.

Sub-função do Volume (*VL*)

Essa sub-função avalia o padrão de carregamento conforme o volume ocupado em relação ao volume disponibilizado pelo *container* e o resultado é uma relação percentual entre a somatória dos volumes das caixas carregadas pelo volume disponibilizado pelo *container*, conforme expressão matemática abaixo.

$$VL = \frac{\sum_{i=1}^{k} VL_{bi}}{VL_c} \times 100 \tag{3.3}$$

Nesta expressão, VL_{bi} é o volume de cada caixa carregada, em que i determina o índice da caixa, e VL_c é o volume disponibilizado pelo *container* e k o número de caixas alocadas no *container*.

O objetivo é ocupar o máximo do volume disponível do *container* com as caixas. Como na obtenção do padrão de carregamento, através da decomposição do espaço, já leva em consideração o volume máximo disponibilizado pelo *container*, conforme suas dimensões, o valor que a função VL pode atingir está entre 0 e 100.

Sub-função do Peso (*P*)

A sub-função do peso (P) avalia o peso total das caixas do padrão de carregamento em relação ao peso máximo permitido pelo *container*.

Considerando que o peso da carga não pode ultrapassar o peso máximo permitido do *container*, a sub-função apresenta duas possibilidades de resultados. Caso o peso das caixas carregadas no *container* ultrapassar o peso máximo do *container*, a sub-função recebe o valor "0" (zero). Se a somatória dos pesos das caixas não ultrapassar o valor do peso máximo do *container*, então, calcula-se a relação entre o peso total da carga pelo peso máximo de carregamento do *container*.

A fórmula abaixo faz uma descrição matemática da sub-função do peso:

$$P = \begin{cases} 0, & \text{se } \sum_{i=1}^{k} P_{bi} > P_c \\ \frac{\sum_{i=1}^{k} P_{bi}}{P_c} \times 100, & \text{se } \sum_{i=1}^{k} P_{bi} \leq P_c \end{cases} \tag{3.4}$$

P_c é o peso máximo suportado pelo *container*, P_{bi} é o peso da caixa de índice i e k representa o número de caixas alocadas no *container*.

O objetivo é maximizar o peso de caixas carregadas no *container* sem ultrapassar o limite máximo de peso suportado pelo *container*.

Sub-função do Centro de Gravidade (G)

O carregamento das caixas também deve ser avaliado com relação à estabilidade da carga dentro do *container* após as caixas serem posicionadas no mesmo. O objetivo é que o centro de gravidade seja o mais baixo possível. Para tanto, a avaliação da sub-função é realizada conforme a fórmula abaixo.

$$G = (\frac{1}{H_c}) \times [1.5H_c - \frac{\sum_{i=1}^{k} P_{bi} \times G_{bi}}{\sum_{i=1}^{k} P_{bi}}] \times 100 \tag{3.5}$$

P_{bi} representa o peso da caixa carregada, G_{bi} representa a distância do centro de gravidade de cada caixa para com a base do *container*, conforme índice i, e assume-se que o valor médio da altura da caixa é o seu centro de gravidade, e H_c é a altura do *container*.

Considera-se como adequado um centro de gravidade no centro geométrico do *container* (metade da altura), ou seja, a relação dada pela somatória dos produtos do centro de gravidade da caixa pelo peso da caixa, deverá resultar no valor $0,5 \cdot H_c \cdot \sum_{i=1}^{k} P_{bi}$. Nesse caso, o valor da sub-função G é 100. Caso o centro de gravidade esteja abaixo do

valor médio da altura do *container*, o valor de G será maior que 100 que representa uma proposta ainda melhor e, caso contrário, G assumirá um valor abaixo de 100.

Sub-função do Valor (V)

A função geral também deve avaliar a configuração de carregamento conforme o valor máximo a ser permitido para a carga do *container*. A somatória dos valores atribuídos a cada caixa deve ser igual ou inferior ao valor máximo previamente determinado.

Assim como a sub-função do peso, a sub-função do valor recebe zero quando o valor total dos produtos carregados for superior ao limite estabelecido. Quando o valor dos produtos carregados está dentro do limite estipulado, é feita a relação entre o valor da carga em relação ao valor máximo estipulado. A fórmula abaixo demonstra como esta sub-função determina seu valor com relação ao atendimento das restrições.

$$V = \begin{cases} 0, & \text{se } \sum_{i=1}^{k} V_{bi} > V_c \\ \frac{\sum_{i=1}^{k} V_{bi}}{V_c} \times 100, & \text{se } \sum_{i=1}^{k} V_{bi} \leq V_c \end{cases} \tag{3.6}$$

V_{bi} é o valor associado a cada produto, V_c é o valor monetário máximo que a carga do *container* deve possuir e k o número de caixas alocadas no *container*.

3.2.3 Decomposição dos Espaços do *Container*

A decomposição dos espaços do *container* para obtenção da seqüência de caixas que permitirá conhecer o padrão de carregamento do *container* é realizada de maneira sistemática.

O processo respeita as restrições que as caixas e o próprio *container* apresentam, como limites dimensionais. As caixas poderão ser rotacionadas (mudança de orientação) em até seis variações.

As caixas preencherão o *container* seguindo a ordem de preenchimento da esquerda para a direita, de baixo para cima e do fundo do *container* para frente.

No processo de decomposição dos espaços do *container* temos uma seqüência de passos a seguir e que deve ser respeitada sua ordem.

1. Primeiramente, verifica-se qual a caixa que tem o maior número de similares, ou seja, qual a caixa que tem o maior número de caixas com dimensões idênticas a ela.

2. A caixa escolhida é alocada no ponto de partida que é a origem do *container*, localizada no canto inferior esquerdo do fundo do *container*.

3. A caixa é rotacionada para que minimize primeiramente o espaço restante na lateral direita do *container*, o ideal seria que não houvesse sobra. Então a dimensão que retornar a menor sobra é fixada como largura da caixa.

4. Depois, nesta ordem de prioridade, rotacionam-se as duas dimensões livres para que minimize o espaço restante superior. Então a dimensão que retornar a menor sobra é fixada como altura da caixa.

5. E por último, na ordem de prioridade, a única dimensão livre da caixa é atribuída ao comprimento da caixa, tendo também uma sobra no espaço frontal do *container*.

6. A partir desta posição fixa, outras caixas com dimensões iguais serão colocadas uma do lado da outra e uma em cima da outra até que o espaço na lateral direita e acima sejam minimizados, formando assim uma camada.

7. Camadas iguais à descrita logo acima preencherão o *container* até que não existam mais caixas disponíveis e iguais às utilizadas na camada anterior ou que o espaço restante na parte frontal do *container* não consiga alocar mais uma camada.

8. A parte do *container* que foi preenchida pelo conjunto de camadas iguais será chamada de Parte Principal ou Corpo Principal do *Container*, e os espaços restantes serão chamados de Espaço Lateral Residual, Espaço Superior Residual e Espaço Frontal Residual.

Portanto, houve uma divisão de quatro partes no *container* para melhor preenchê-lo.

Logo após o Corpo Principal do *Container* ser encontrado, é verificado se os espaços residuais são excessivos. Será considerada sobra excessiva se esta ultrapassar cerca de 4% do tamanho da dimensão do *container* relacionada com esta sobra. Se não for constatado sobra excessiva deve-se ignorar os passos relacionados a ela.

9. Verifica-se a largura do Espaço Lateral Residual, se esta largura for maior que 4% da largura do *container* é verificado se esta largura é maior ou igual ao tamanho da menor dimensão de caixa disponível para carregamento. Se esta condição for satisfeita não serão executados os passos a partir do passo 10.

10. Para um resultado negativo da condição acima este passo deve ser executado, se a largura do Espaço Lateral Residual for maior que 4% da largura do *container*, mas o tamanho da sobra é menor que o tamanho da menor dimensão de caixa encontrada entre as caixas disponíveis, deve-se escolher a dimensão que resulta na segunda menor sobra para o espaço lateral.

11. Verifica-se a largura do Espaço Lateral Residual, se esta largura for maior que 4% da largura do *container* é verificado se a largura do Espaço Lateral Residual é maior ou igual ao tamanho da menor dimensão de caixa disponível para o carregamento. Se esta condição for satisfeita não serão executados os passos a partir do passo 12.

12. Para um resultado negativo da condição acima este passo deve ser executado, se a largura do Espaço Lateral Residual for maior que 4% da largura do *container*, mas o tamanho da sobra é menor que o tamanho da menor dimensão de caixa encontrada entre as caixas disponíveis, deve-se escolher a dimensão que resulta na terceira menor sobra para o espaço lateral.

13. Verifica-se a largura do Espaço Lateral Residual, se esta largura for maior que 4% da largura do *container* é verificado se a largura do Espaço Lateral Residual é maior ou igual ao tamanho da menor dimensão de caixa disponível para o carregamento. Se esta condição for satisfeita não será executado o passo 14.

14. Se nenhuma das tentativas acima foi satisfeita, a primeira caixa alocada na origem do *container* deve ser substituída pela caixa que detém o segundo maior número de caixas similares a ela, se mesmo esta não satisfazer a condição dos 4%, deve ser testada a caixa que detém o terceiro maior número de caixas similares a ela e se mesmo assim esta terceira caixa não satisfazer a condição imposta, deve-se optar pela melhor entre elas, ou seja, a caixa que resultou na menor sobra, mesmo não satisfazendo a condição imposta será a escolhida. Volta-se ao passo 4.

O mesmo deve acontecer com o Espaço Superior Residual.

15. Verifica-se a altura do Espaço Superior Residual, se esta altura for maior que 4% da altura do *container* é verificado se esta altura é maior ou igual ao tamanho da menor dimensão de caixa disponível para o carregamento. Se esta condição for satisfeita, não serão executados os passos a partir do passo 16.

16. Para um resultado negativo da condição acima, este passo deve ser executado, se a altura do Espaço Superior Residual for maior que 4% da altura do *container*, mas o

tamanho da sobra é menor que o tamanho da menor dimensão de caixa encontrada, avaliando os tipos de caixas disponíveis, deve-se escolher a outra dimensão que está livre e que resulta na segunda menor sobra para o espaço superior.

17. Verifica-se a altura do Espaço Superior Residual, se esta altura for maior que 4% da altura do *container* é verificado se a altura do Espaço Superior Residual é maior ou igual ao tamanho da menor dimensão de caixa disponível para o carregamento. Se esta condição for satisfeita não será executado o passo 18.

18. Caso as tentativas anteriores não forem satisfeitas, deve-se retirar a última caixa colocada acima das outras e voltar ao passo 7.

Com o Corpo Principal do *Container* definido, os espaços residuais serão preenchidos procurando seguir as mesmas orientações que a parte principal.

A ordem de preenchimento dos espaços residuais é primeiramente o Espaço Lateral Residual, em seguida Espaço Superior Residual e por último Espaço Frontal Residual.

19. Procurando-se caixas de dimensões idênticas e capazes de preencher ao máximo os espaços residuais, a partir do momento que a caixa foi escolhida para preencher o espaço livre do *container*, ela não poderá mais ser utilizada para preenchimentos posteriores.

Dentro do processo de decomposição, os espaços do *container* são analisados separadamente, verificando a existência de caixas de mesmas dimensões, mas pesos diferentes. Estas caixas serão colocadas dentro do espaço de maneira que se encontre abaixo das outras ou em camadas específicas para não ocorrer esmagamento de carga.

A Figura 8 traz o fluxograma mais detalhado do Processo de Decomposição do Corpo Principal do *Container*.

A Figura 9 traz o fluxograma detalhado do Processo de Decomposição do Espaço Lateral Residual do *Container*.

A Figura 10 traz o fluxograma do Processo de Decomposição do Espaço Superior Residual do *Container*.

A Figura 11 traz o fluxograma do Processo de Decomposição do Espaço Frontal Residual do *Container*.

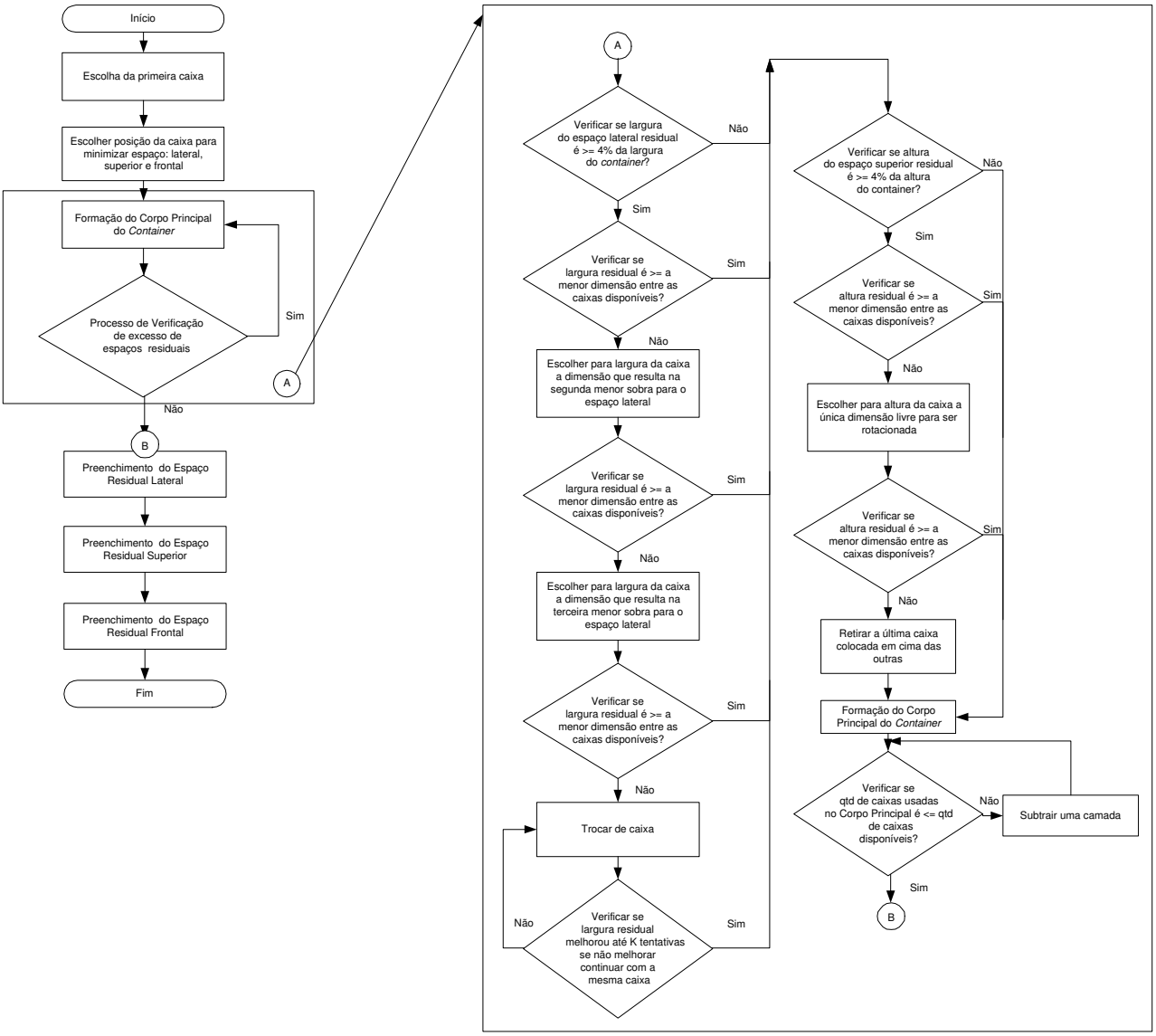

Figura 8: Fluxograma detalhado do Processo de Decomposição do Corpo Principal do *Container*.

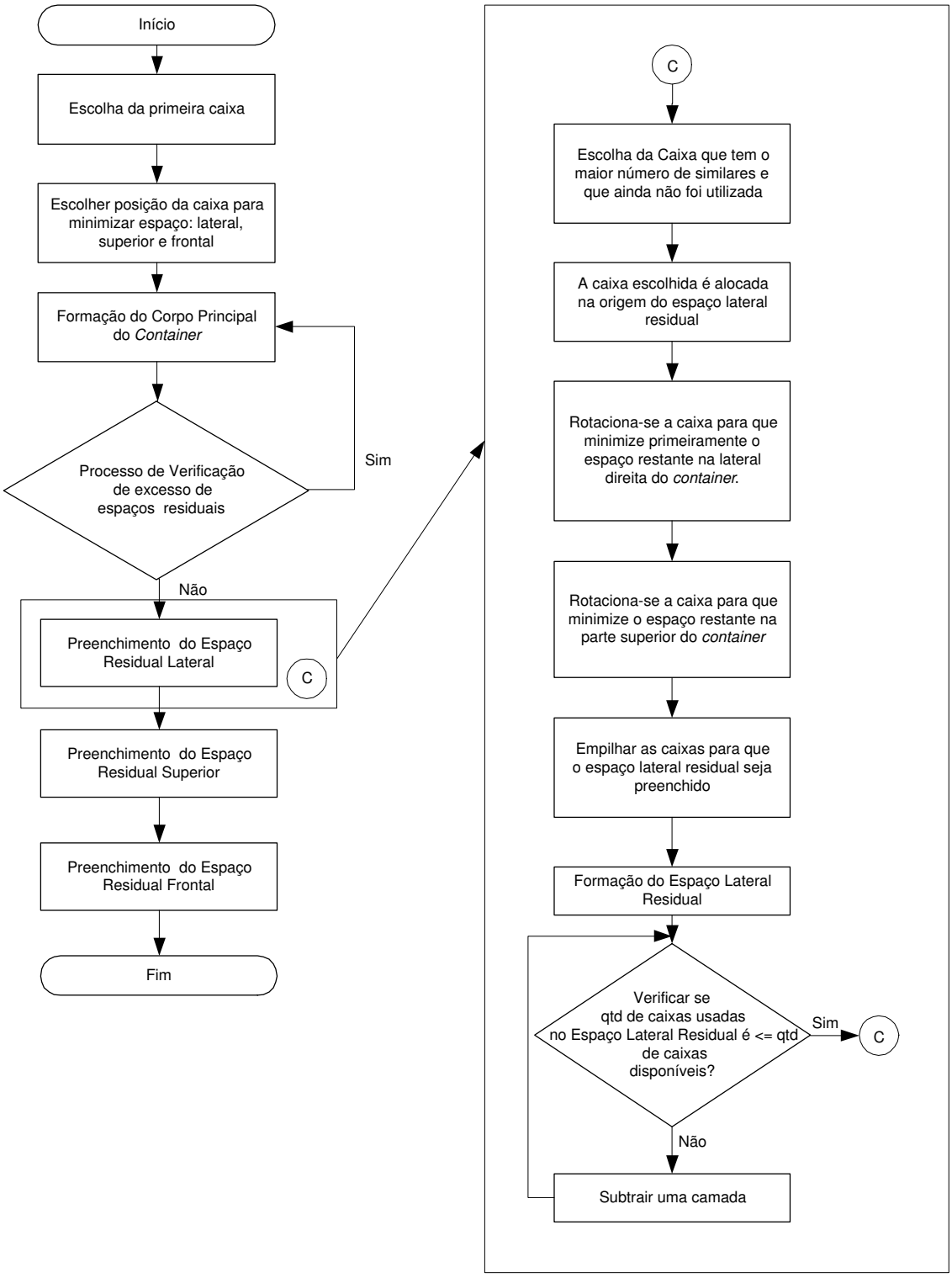

Figura 9: Fluxograma do Processo de Decomposição do Espaço Lateral Residual do *Container*.

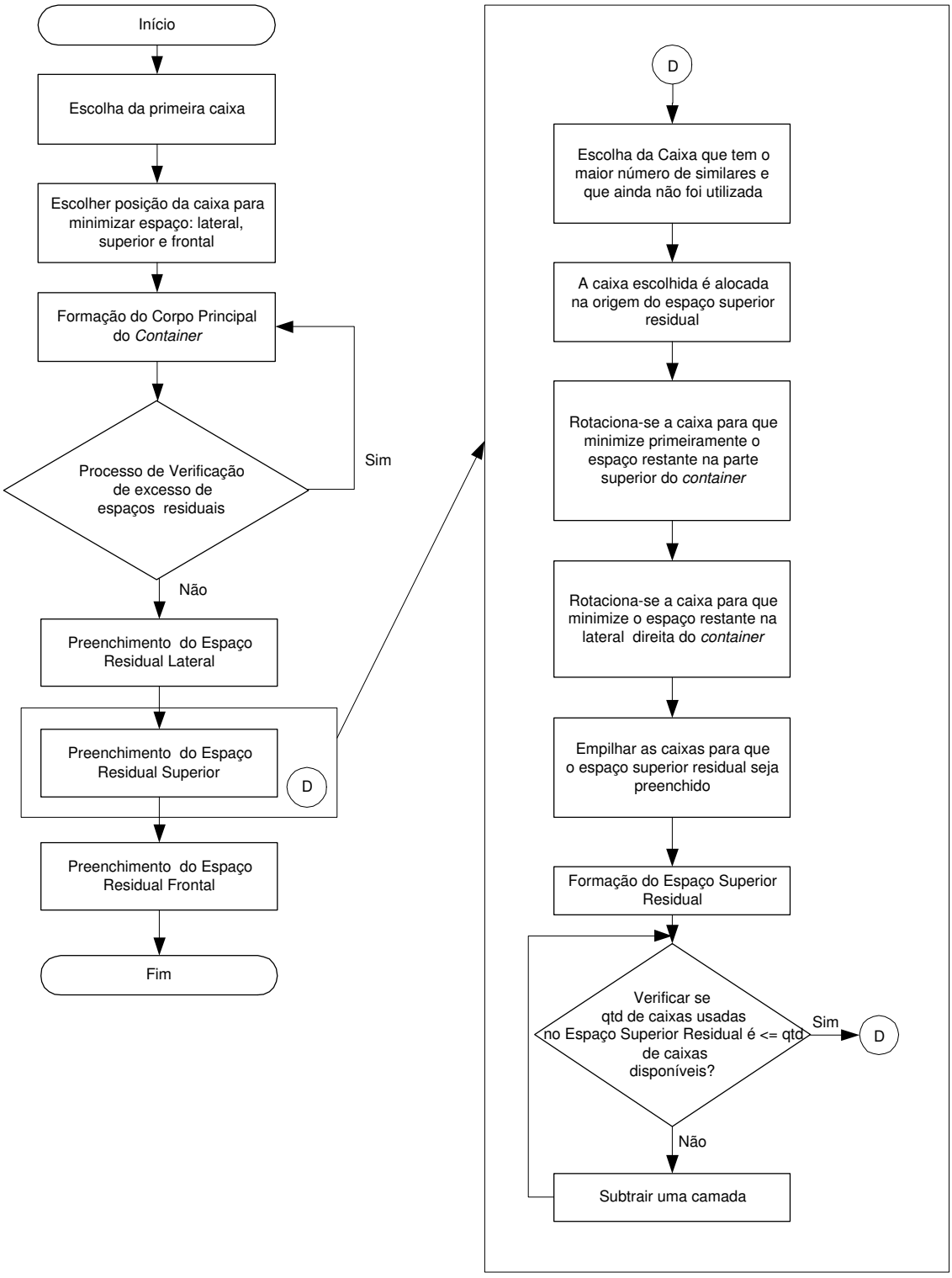

Figura 10: Fluxograma do Processo de Decomposição do Espaço Superior Residual do *Container*.

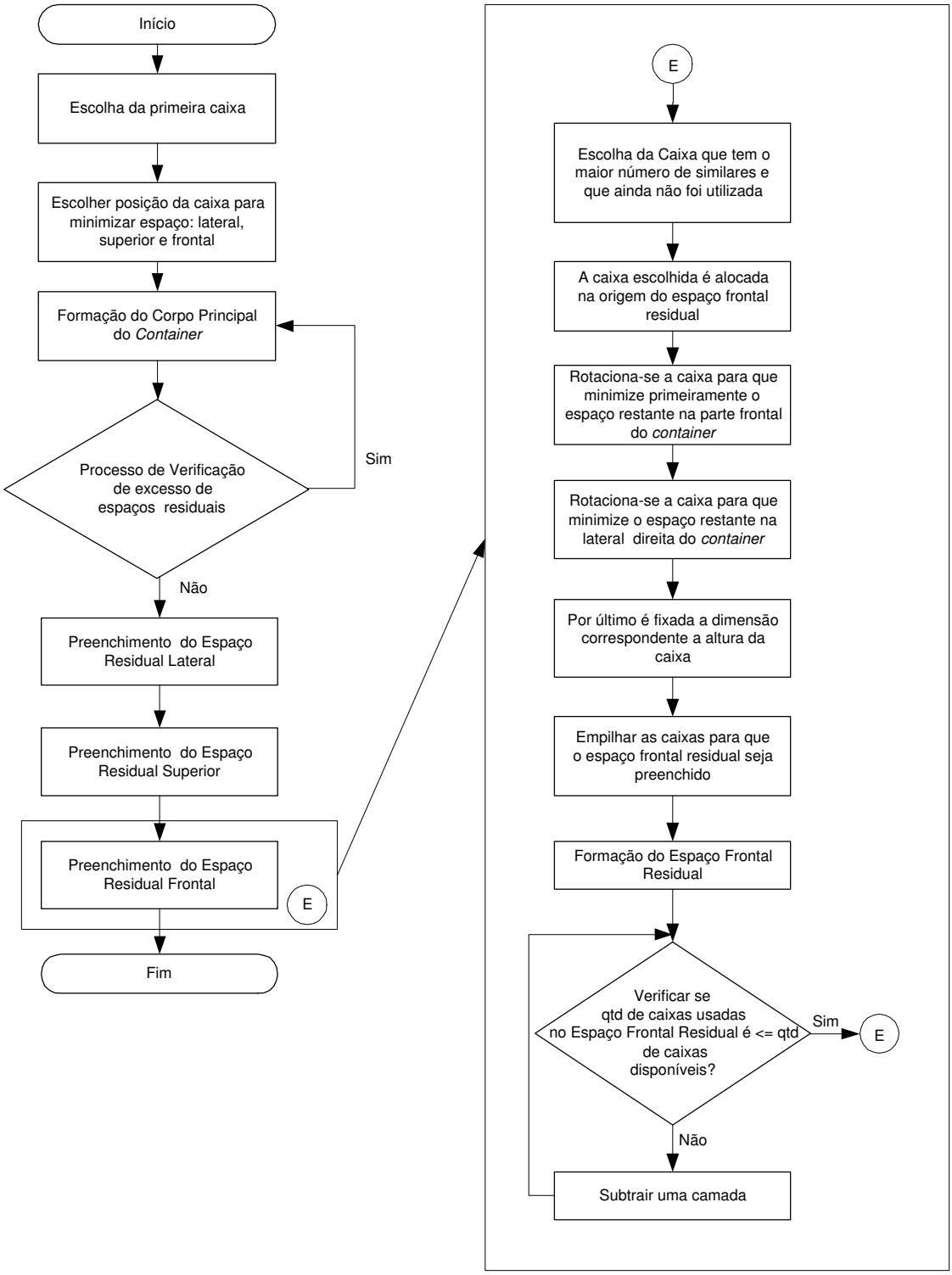

Figura 11: Fluxograma do Processo de Decomposição do Espaço Frontal Residual do *Container*.

3.3 Testes

A heurística desenvolvida foi testada com diferentes dimensões de caixas disponíveis para o carregamento e com diferentes tipos de carga (fortemente homogênea e fortemente heterogênea) com quantidades diferentes de caixas também.

O primeiro teste realizado, utilizou uma carga fortemente homogênea com 100 caixas disponíveis (**Sistema I**). Os dados dimensionais das caixas e do *container* podem ser encontrados no apêndice deste trabalho.

O segundo teste realizado utilizou uma carga fortemente heterogênea com 285 caixas disponíveis para o carregamento (**Sistema II**). Os dados dimensionais das caixas e do *container* também podem ser encontrados no apêndice deste trabalho.

A heurística foi desenvolvida em FORTRAN 4.0 e os testes foram realizados em um PC HP BRIO PentiumII Processador IntelMMX/ 256 MB de memória RAM, com sistema operacional Windows 98.

Nos testes realizados, foram utilizados dois tipos de função objetivo. O primeiro cálculo trabalha somente com o volume ocupado pela carga alocada no *container*, veja seção 3.2.2.1, e o segundo cálculo trabalha com uma função objetivo geral proposta por Rodrigues em (RODRIGUES, 2005), veja seção 3.2.2.2.

3.3.1 Teste com Sistema I

O teste do Sistema I, que trabalha com 100 caixas praticamente iguais (carga fortemente homogênea) teve os seguintes resultados:

(a) Considerando somente Volume como Função de Avaliação

O percentual de aproveitamento do volume do *container* foi de 79.63% da carga alocada em seu interior.

(b) Considerando a Função de Avaliação proposta por Rodrigues

Utilizando o cálculo proposto por Rodrigues em (RODRIGUES, 2005) alcançou neste estudo o valor de 71.41% para a Função Geral, 79.63% para o volume ocupado, 42.64% do valor admitido, 7.9% do peso permitido e 133.69% de equilíbrio alcançado.

O programa foi executado em menos de 1 min, e o padrão de carregamento resultante

é o seguinte:

- **Corpo Principal**: 48 caixas entre os tipos B, C, D e E. Os tipos de caixas, bem como suas dimensões, valor e peso podem ser encontrados no apêndice do trabalho.
- **Espaço Lateral Residual**: não foi possível preenchê-lo, espaço de sobra desprezível.
- **Espaço Superior Residual**: 18 caixas do tipo A.
- **Espaço Frontal Residual**: 4 caixas do tipo F.

O vetor auxiliar traz a posição de cada caixa a partir dos seguintes índices:

- **Corpo Principal**: 1.
- **Espaço Lateral Residual**: não foi possível preenchê-lo, espaço de sobra desprezível.
- **Espaço Superior Residual**: 1.
- **Espaço Frontal Residual**: 2.

3.3.2 Teste com Sistema II

O teste do Sistema II, que trabalha com 285 caixas, representando uma carga do tipo fortemente heterogênea, teve os seguintes resultados:

(a)Considerando somente Volume como Função de Avaliação

Neste segundo teste, foi utilizado para a avaliação do padrão de carregamento somente o volume da carga alocada no interior do *container* e o percentual de aproveitamento do volume do *container* foi de 94.5%.

(b)Considerando a Função de Avaliação proposta por Rodrigues

Utilizando a proposta de Rodrigues em (RODRIGUES, 2005) para a avaliação do padrão de carregamento, foi alcançado neste trabalho o valor de 78.6% para a Função Geral, 94.5% para o volume ocupado, 23.1% do valor admitido, 4.6% do peso permitido e 136.2% de equilíbrio alcançado. O melhor resultado entre as simulações.

O programa foi executado com um pouco mais de 2 min, e o padrão de carregamento resultante é o seguinte:

- **Corpo Principal**: 66 caixas entre os tipos E, F e G. Os tipos de caixas, bem como suas dimensões, valor e peso podem ser encontrados no apêndice do trabalho.
- **Espaço Lateral Residual**: não foi possível preenchê-lo, espaço de sobra desprezível.
- **Espaço Superior Residual**: 24 caixas do tipo A.
- **Espaço Frontal Residual**: 16 caixas do tipo B.

O vetor auxiliar traz a posição de cada caixa a partir dos seguintes índices:

- **Corpo Principal**: 4.
- **Espaço Lateral Residual**: não foi possível preenchê-lo, espaço de sobra desprezível.
- **Espaço Superior Residual**: 3.
- **Espaço Frontal Residual**: 3.

3.4 Análise de Resultados

Como observado a heurística proposta demonstrou ser uma técnica eficiente para o problema de carregamento de *container*.

Os resultados obtidos são altamente satisfatórios, tendo em vista que na literatura encontram-se resultados de boa qualidade como os aqui apresentados.

Em Wu e outros (WU et al., 2002), por exemplo, os resultados obtidos mostram que o percentual médio de área não utilizada no carregamento foi de 5% utilizando a heurística 2 e 3% utilizando a heurística 1, ou seja, os resultados obtidos com a aplicação das heurísticas apresentadas aqui neste trabalho têm valores próximos aos encontrados em (WU et al., 2002), com uma vantagem do tempo de execução da heurística. Neste trabalho o tempo médio de execução da heurística foi de 1.5 min, o tempo médio apresentado pelos autores em (WU et al., 2002), foi de 15 min.

Comparando os resultados deste trabalho com os resultados de Rodrigues em (RODRIGUES, 2005), fica claro o melhor desempenho do algoritmo aqui proposto em relação a carga fortemente heterogênea.

Em Rodrigues (RODRIGUES, 2005), os testes realizados com as mesmas 100 caixas fortemente homogêneas testadas neste trabalho, ele obteve um percentual médio de volume ocupado no *container* de 83.20%, com função geral no valor de 74.34%.

Já com a carga fortemente heterogênea, utilizando as mesmas 285 caixas testadas neste trabalho, o resultado obtido por ele foi de 70.19% de volume ocupado e função geral de 69.0%.

Como foi possível observar, os resultados obtidos neste trabalho são superiores aos resultados de Rodrigues em (RODRIGUES, 2005) quando se trata da carga fortemente heterogênea.

A heurística proposta se mostrou flexível ao se adaptar ao problema específico estudado, trabalhando de maneira aceitável com grupos de caixas fortemente homogêneas, mas o desempenho obtido com a aplicação da heurística proposta neste trabalho com caixas fortemente heterogêneas foi satisfatório e de boa qualidade.

4 *Algoritmo Genético Especializado para o Problema de Carregamento de Container*

Conforme mencionado nos capítulos anteriores, as técnicas metaheurísticas, em especial os algoritmos genéticos, são amplamente aplicados na resolução do problema de carregamento de *container* e nos problemas de otimização em geral.

Segundo Romero e Mantovani em (ROMERO; MANTOVANI, 2004) as metaheurísticas nada mais são do que um conjunto de técnicas de otimização adaptadas para lidar com problemas complexos e que apresentam a característica da explosão combinatória.

Como o problema do carregamento de *container* é considerado NP-Difícil é perfeitamente aceitável o emprego de heurísticas e metaheurísticas para sua resolução.

Os algoritmos genéticos normalmente associam à sua estrutura tradicional, um procedimento complementar para a resolução do problema aqui estudado.

Neste capítulo é mostrado com detalhes o algoritmo genético proposto para a resolução do problema de carregamento de *container*, detalhando o tipo de algoritmo genético que foi a base para o proposta neste trabalho. Os testes e os resultados obtidos também serão relatados.

O capítulo inicia-se com uma introdução sobre algoritmo genético; na seção 4.2 mostra o funcionamento do algoritmo genético tradicional; na seção 4.3 é tratada a questão do algoritmo genético modificado, proposto por Chu-Beasley; já na seção 4.4 é apresentado o algoritmo genético especializado - Chu-Beasley modificado. Este algoritmo modificado é a proposta metaheurística deste trabalho para a resolução do carregamento de *container*; finalizando, na seção 4.5 são relatados os testes realizados e as simulações do carregamento de *container* usando o algoritmo genético proposto. O capítulo traz ainda na seção 4.6, uma análise dos resultados obtidos.

4.1 Introdução

O Algoritmo Genético segundo Soares em (SOARES, 1997) é uma técnica de busca através de propostas de solução para um determinado problema. Foi formulado inicialmente por John Holland nos anos 60 e desenvolvido na Universidade de Michigan, por ele e seus alunos, em meados dos anos 70.

Holland tinha como seu principal objetivo dedicar-se ao estudo informal do fenômeno da evolução, como ocorre na natureza e desenvolver maneiras para incorporá-los aos sistemas computacionais.

A evolução das espécies é determinada por um processo de seleção que leva à sobrevivência dos indivíduos geneticamente melhores adaptados para superar os problemas do meio ambiente em que vivem. Por exemplo: Quando uma determinada população tem que se adaptar às novas condições impostas pelo ambiente em que vivem, sejam elas falta de comida, espaço ou outro recurso essencial, os indivíduos que mais se adaptarem a essas novas condições sobrevivem, enquanto os mais fracos são descartados e extintos do meio. Isso acontece porque, dentre todas as características imprescindíveis à sobrevivência, esses seres possuem algumas dessas características mais acentuadas que as dos outros seres. Por herança essas características provavelmente passarão para seus descendentes e, assim, eles terão grandes chances de sobreviverem. Por outro lado, fortes indivíduos podem surgir da exploração de uma outra característica ainda não desenvolvida na população. Se a natureza tentasse descobrir essas novas características através da seleção dos mais aptos e da recombinação dentro de um mesmo grupo certamente não teria sucesso, visto que, depois de muitas gerações, todos os membros compartilhariam praticamente do mesmo código genético. Para contornar o problema a natureza insere material genético diferente através do processo conhecido como mutação. Se este indivíduo, que sofreu mutação, estiver tão capacitado à sobrevivência, quanto os atuais, suas chances são grandes em um futuro processo de seleção.

Com esta motivação os algoritmos genéticos nasceram seguindo uma metodologia baseada nos mecanismos da seleção natural e evolução, mecanismos estes que foram propostos na teoria de Charles Darwin.

A evolução ocorre quando novos indivíduos são introduzidos na população. Com ela, pretende-se em cada geração, encontrar soluções mais qualificadas. Para isso, o algoritmo genético realiza o chamado "ciclo geracional", ou seja, instauram-se os operadores de seleção, recombinação e mutação. Na seleção, os melhores indivíduos são privilegiados

com maior probabilidade de participar da nova população. Na recombinação, é feita a troca de material genético entre as configurações e na mutação, novos genes são incorporados à população. Portanto, para que a população possa evoluir é necessário definir uma representação adequada dos seus indivíduos. Esses indivíduos constituem as configurações e, no processo de otimização, as configurações representam as soluções candidatas dos problemas a serem resolvidos. Assim, a escolha da codificação está diretamente relacionada às características do problema.

Os algoritmos genéticos encontram de maneira adequada os ótimos locais presentes nos problemas que possuem elevada quantidade de alternativas, pois, por meio da população (conjunto de configurações) é possível avaliar, simultaneamente, várias regiões pertencentes ao mesmo espaço de busca. Esta característica entre outras se torna responsável pelo bom desempenho do método.

4.2 Funcionamento do Algoritmo Genético Tradicional

Antes de iniciar uma explicação mais detalhada do funcionamento do algoritmo genético tradicional é preciso ter o conhecimento dos conceitos de alguns termos comuns no manuseio do algoritmo genético, estes termos técnicos serão definidos a seguir para melhor entendimento do trabalho realizado.

4.2.1 Sistema Natural x Algoritmo Genético

Para melhor entendimento dos termos usados na genética natural e aplicados no algoritmo genético, será feita aqui uma analogia entre os termos. Como o problema analisado neste trabalho é o carregamento de *container* os termos conceituados serão direcionados a este contexto.

Um indivíduo ou cromossomo corresponde à ordem das torres (formadas por caixas) que serão carregadas no *container*, onde cada gene corresponde ao índice da torre no cromossomo e a população ou geração é o conjunto de cromossomos. No algoritmo genético, o processo de obtenção de uma nova população de cromossomos também pode ser descrito como formação de uma nova geração de cromossomos. Termos como recombinação, mutação, população, estão diretamente ligados aos indivíduos.

Finalizando, a Tabela 2 mostra a relação existente entre as entidades de sistemas

naturais e os termos que são usados no algoritmo genético.

Tabela 2: Analogia entre Sistemas Naturais e os Algoritmos Genéticos.

Genética Natural	**Algoritmo Genético**
Gene	Torre
Alelo	Índice de cada torre
Indivíduo (cromossomo)	Configuração ou proposta de solução para o problema. Esta configuração é formada por torres.
População	Conjunto de propostas de solução do problema.
Locus	Posição de cada torre dentro do cromossomo
Genótipo	Estrutura, cromossomo
Fenótipo	Estrutura decodificada

O algoritmo, na sua forma mais simples, deve cumprir as seguintes etapas:

1. Representar adequadamente uma configuração do problema. A mais popular é a representação em codificação binária, em que os operadores genéticos de recombinação e mutação são facilmente simulados;

2. Encontrar uma forma correta de avaliar a função objetivo ou o seu equivalente (*fitness*) e identificar as configurações de melhor qualidade;

3. Desenvolver uma estratégia de seleção das configurações, que atribua às configurações de melhor qualidade uma maior participação na formação das configurações da nova população (nova geração);

4. Criar um mecanismo que permita implementar o operador genético de recombinação;

5. Implementar o operador genético de mutação e terminar de gerar a nova população;

6. Parar quando o critério de parada for satisfeito. Caso contrário, voltar ao passo 2.

A seguir, a Figura 12 traz o fluxograma do funcionamento do Algoritmo Genético tradicional.

4.2.2 Funcionamento do Algoritmo Genético

No decorrer desta seção, encontram-se explicados com mais detalhes os principais tópicos do algoritmo genético, a partir da codificação e representação das configurações.

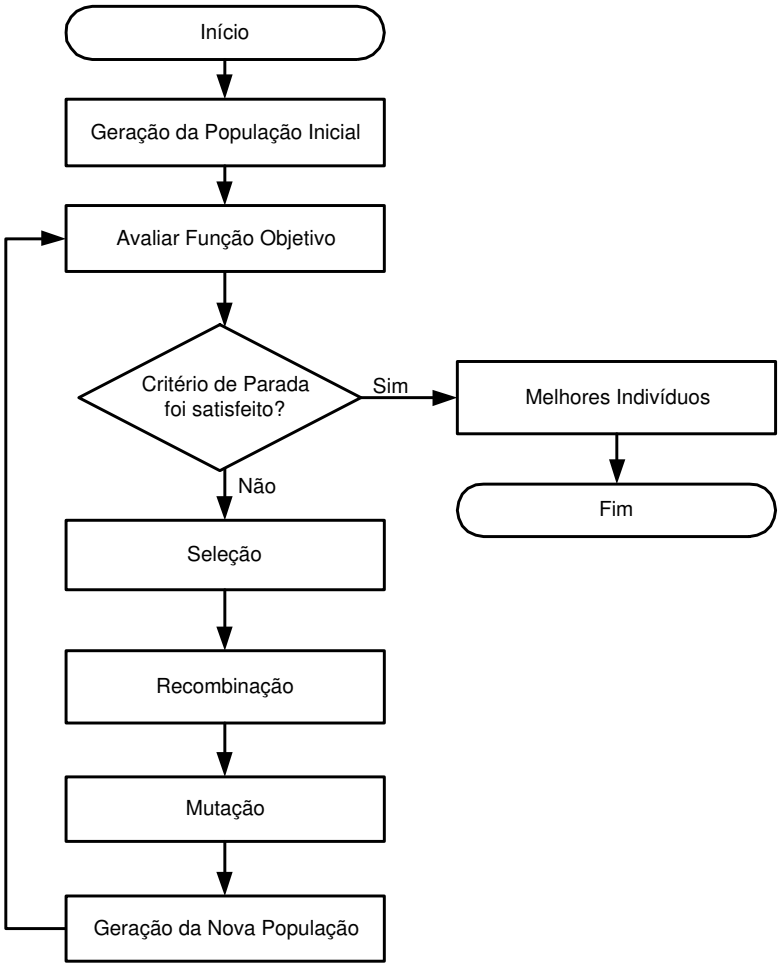

Figura 12: Fluxograma do funcionamento dos Algoritmos Genéticos tradicionais.

4.2.2.1 Codificação

Para se trabalhar com o algoritmo genético é preciso representar as soluções candidatas através de uma codificação.

A forma de implementar a codificação, depende do problema a ser estudado, da natureza das variáveis de decisão do problema ou da forma de representar uma configuração usada no problema. Existem problemas que trabalham com variáveis inteiras, outras reais e ainda outras binárias.

O que deve ser realmente levado em consideração quando se codifica um problema para ser resolvido por algoritmo genético, são as características do problema. São estas características que irão dizer qual a melhor maneira de codificar as soluções candidatas, se é melhor codificar binariamente ou usando variáveis inteiras ou ainda variáveis reais.

4.2.2.2 Função Objetivo

Antes de colocar qualquer Algoritmo Genético em prática é preciso analisar e estudar qual será o problema a ser otimizado.

Faz-se necessário um estudo minucioso sobre os fatores do problema, as variáveis envolvidas, todas as informações que influenciam direta e indiretamente no problema.

Esta função retorna o desempenho de cada configuração avaliada ou a aptidão de cada uma, ou seja, ela é utilizada para verificar a qualidade de cada configuração para mais tarde poderem ser comparadas e verificar quais são as melhores, aplicando o operador de seleção.

Muitas vezes, quando precisa-se padronizar os resultados da função objetivo, para posteriormente serem analisados e submetidos ao processo de seleção, encontram-se algumas dificuldades em retratar a função objetivo como ela é. Quando o objetivo da função não é maximizar e sim minimizar, tem-se um exemplo claro dessa dificuldade, haja visto que para a maioria dos operadores de seleção precisa-se trabalhar com valores padronizados, com exceção do processo de seleção por torneio que não necessita padronizar valores.

Portanto, os algoritmos genéticos trabalham em termos de maximização e muitas vezes o objetivo é minimizar uma função. Em casos como esse, é necessário transformar a função objetivo em uma outra função, chamada de função adaptativa ou *fitness function*, onde esse problema seja corrigido e possa de alguma maneira comparar resultados e aplicar o operador de seleção. Entretanto como já foi dito, para determinados tipos de seleção, como

a seleção por torneio, não existe esse problema e, portanto, trabalha-se sem dificuldades com problemas de maximização ou minimização sem necessidade de transformação ou de padronização.

A função objetivo também está diretamente ligada com a infactibilidade de uma configuração. Ao avaliar uma configuração o resultado pode ser factível ou infactível e essa informação deve ser incorporada na função objetivo. Para corrigir este problema pode-se usar uma penalidade para infactibilidades encontradas ou aplicar os operadores genéticos prevendo este tipo de problema.

4.2.2.3 Métodos de Seleção

O operador de seleção é responsável pela escolha das configurações que serão submetidas aos operadores genéticos como recombinação e mutação, e as configurações resultantes dessas operações farão parte da nova geração. Não é uma boa característica favorecer sempre a seleção do melhor, muito menos uma escolha aleatória. Por um lado há a possibilidade de ocorrer convergência prematura e por outro, a pesquisa é aleatória, deixando de explorar as informações contidas no interior da população. A seguir, é feita a descrição de alguns métodos de seleção.

Seleção Proporcional: O método da seleção proporcional é um método simples, mas que gera alguns problemas na seleção de configurações que podem ter uma participação na próxima geração através de seus descendentes. As configurações de uma geração são selecionadas para a próxima geração utilizando uma roleta.

Cada configuração da população é representada na roleta por uma fatia proporcional à razão do valor da função objetivo da configuração pela média da função objetivo da população.

A roleta é girada um determinado número de vezes, número este definido pelo tamanho da população. A cada giro da roleta uma configuração é apontada pela agulha e então selecionada.

Existem outros métodos de seleção que utilizam a roleta e são derivados do método de seleção proporcional, um exemplo é a **Seleção Estocástica de Resíduos**.

A seguir temos a Figura 13, que representa a roleta usada neste método de seleção.

Seleção por Torneio: Este método também é muito simples e tem a vantagem de eliminar os problemas que são encontrados usando a seleção proporcional.

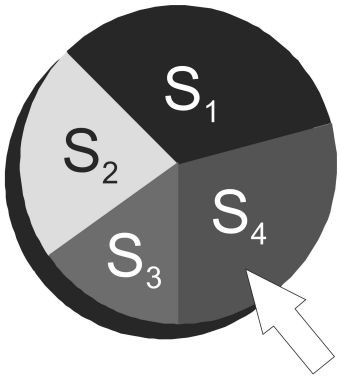

Figura 13: Roleta do Método de Seleção Proporcional.

Ele retorna a melhor configuração, ou seja, a que traz uma função objetivo com valor melhor entre j indivíduos participantes. Este método busca dificultar, mas não elimina, as possibilidades de uma configuração com baixo desempenho ser escolhida. A seleção por torneio está sendo preferida pelos pesquisadores porque apresenta bom desempenho e facilidade de implementação computacional.

Existem ainda outros métodos de seleção como **Seleção Determinística** e **baseada em Ordenamento**.

4.2.2.4 Métodos de Recombinação

A recombinação é o principal operador genético que atua sobre as configurações selecionadas.

Ele é o maior responsável pela troca de material genético entre duas configurações participantes, fazendo com que novas e melhores gerações apareçam.

Por este motivo, fica claro que ele é um poderoso mecanismo de combinar possibilidades e vasculhar todo o espaço de busca para encontrar a melhor solução.

Depois que as configurações são selecionadas, o operador de recombinação entra em ação. As configurações são separadas em pares e a recombinação é realizada conforme a probabilidade pré-determinada.

As configurações escolhidas pelo processo de seleção, devem ser separadas em pares e sofrer a ação do operador de recombinação, trocando parcelas entre si a partir de um ponto escolhido aleatoriamente para formar duas novas configurações.

Existem vários métodos de recombinação, a seguir serão apresentados alguns métodos

e, para ajudar na compreensão, serão usados como o par para recombinação as seguintes configurações:

1 - 1 0 1 0 0 0 0 0 1 0 0 1 1 1 0

2 - 1 1 0 1 1 1 0 1 1 0 0 1 0 0 0

Recombinação de um simples ponto: Aleatoriamente é escolhido o ponto de corte que servirá para trocar material genético entre as duas configurações. A posição escolhida deve ser comum entre o par, mas não necessariamente deve ser igual para todos os pares selecionados.

Recombinação de múltiplos pontos: Similarmente com a recombinação com ponto simples, os locais são escolhidos aleatoriamente. Os locais de corte do indivíduo acima variam de 1 a $N - 1$, sendo N o número de elementos da configuração. Então, supondo que se esteja tratando do tipo com 4 pontos de corte, e que os escolhidos sejam os pontos 4, 6, 9 e 12, tem-se :

1 - 1 0 1 0 — 0 0 — 0 0 1 — 0 0 1 — 1 1 0

2 - 1 1 0 1 — 1 1 — 0 1 1 — 0 0 1 — 0 0 0

Resultado:

Primeiro Corte:

1 - 1 0 1 0 — 1 1 0 1 1 0 0 1 0 0 0

2 - 1 1 0 1 — 0 0 0 0 1 0 0 1 1 1 0

Segundo Corte:

1 - 1 0 1 0 1 1 — 0 0 1 0 0 1 1 1 0

2 - 1 1 0 1 0 0 — 0 1 1 0 0 1 0 0 0

Terceiro Corte:

1 - 1 0 1 0 1 1 0 0 1 — 0 0 1 0 0 0

2 - 1 1 0 1 0 0 0 1 1 — 0 0 1 1 1 0

Quarto Corte:

1 - 1 0 1 0 1 1 0 0 1 0 0 1 — 1 1 0

2 - 1 1 0 1 0 0 0 1 1 0 0 1 — 0 0 0

Portanto resultado final:

1 - 1 0 1 0 1 1 0 0 1 0 0 1 1 1 0

2 - 1 1 0 1 0 0 0 1 1 0 0 1 0 0 0

Recombinação Uniforme: Seguindo cada bit da primeira configuração, é verificado se ocorreu um evento com probabilidade de 50%. Caso afirmativo, ali é um ponto de corte, caso contrário repete-se o procedimento para o bit posterior.

4.2.2.5 Operador de Mutação

A mutação é um operador genético que tem a função de introduzir características novas ao indivíduo ou mesmo restaurar características que se perderam em operações como por exemplo, de recombinação.

Na teoria, a mutação é um evento que possui uma probabilidade p_m de acontecer ou não para cada bit da cadeia de caracteres de todas as configurações da população.

Dependendo da probabilidade p_m, uma quantidade de bits sofrerá a ação da mutação. Serão escolhidos aleatoriamente os bits para serem mutados, estes bits poderão estar localizados em qualquer configuração da população e em qualquer posição dentro da configuração.

Se nos bits escolhidos for encontrado o valor zero, este será substituído pelo valor um e vice-versa.

O operador de mutação tem um papel fundamental na evolução da população, o de introduzir e restaurar características além de ajudar a evitar o problema de ótimo local (convergência prematura). Mas deve-se tomar cuidado com a implementação, pois pode-se ter um custo computacional elevado.

No algoritmo genético simples, a mutação é considerada como um operador secundário, comparado com a recombinação, e tem a finalidade de restaurar a perda de material genético que pode acontecer na população.

Depois da mutação, as configurações podem apresentar infactibilidades, é necessário formular estratégias, permitindo transformá-las em factíveis. Uma alternativa é considerá-las todas factíveis e penalizar as infactibilidades na função objetivo, a fim de que elas sejam eliminadas pelo operador de seleção.

4.2.2.6 Parâmetros dos Algoritmos Genéticos

A cada geração os algoritmos genéticos procuram as configurações com um melhor desempenho que as anteriores.

A aplicação dos parâmetros de controle dos algoritmos genéticos (tamanho da população, taxa de recombinação e a taxa de mutação) bem como qual método utilizar para selecionar, recombinar e mutar, influenciam no comportamento do Algoritmo Genético. Por este motivo serão abordados.

Número de Indivíduos da População (n_p): Nos algoritmos genéticos, para que seja possível efetuar operações como recombinação e mutação, a configuração deve estar devidamente codificada. Uma população pequena possui amostragem insuficiente, podendo conduzir o algoritmo na direção de um ótimo local. Uma população grande contém uma quantidade bem representativa do espaço de busca, e também, a perda da diversidade da população ocorrerá mais lentamente, possibilitando os algoritmos genéticos explorarem mais a informação existente. Entretanto, o número de cálculos da função objetivo por geração pode resultar num tempo computacional inaceitável.

Taxa de Recombinação (p_c): Esse parâmetro controla a freqüência com que o operador de recombinação é aplicado. A cada nova geração, provavelmente uma parcela da população será recombinada.

Taxa de Mutação (p_m): A mutação tem um papel diferente, mas não menos importante que a recombinação. As funções da mutação são: inserir material genético novo e restaurar valores perdidos na recombinação ou mesmo na mutação.

Este parâmetro controla a freqüência com que o operador de mutação é aplicado.

Com base no programa de controle do algoritmo genético, um conjunto de parâmetros é idealizado para definir o tamanho da população, a taxa de recombinação e a taxa de mutação. Esses parâmetros variam de problema para problema e, em grande parte, definem a qualidade do algoritmo.

4.2.2.7 Critérios de Parada

Existem vários critérios de parada, podendo ser implementado no programa apenas um critério e dependendo do problema mais que um simultaneamente.

Alguns critérios de parada dependem da evolução dos resultados do programa e outros são especificados previamente.

Os critérios de parada mais utilizados para terminação de programas são:

1. Executar um número pré-determinado de gerações.

2. Quando a melhor solução encontrada no problema assume um valor equivalente a um valor previamente especificado.

3. Quando a incumbente (melhor solução encontrada) não melhora durante um número especificado de iterações.

4. Quando as configurações da população perdem a diversidade, se tornando muito parecidas umas com as outras, não evoluindo.

5. Usar um critério que depende do tipo de problema analisado. Em problemas reais, são relatados vários critérios de parada.

4.3 Algoritmo Genético Modificado (Chu-Beasley)

O algoritmo genético modificado de Chu-Beasley traz algumas diferenças em relação ao algoritmo genético tradicional. O algoritmo proposto por Chu-Beasley pode ser considerado como um algoritmo genético modificado, mas é significativamente diferente de um algoritmo genético tradicional e de versões modificadas que existem na literatura especializada. Estas diferenças serão apresentadas a seguir.

Chu-Beasley em (CHU; BEASLEY, 1997) apresentaram uma proposta para a resolução do problema generalizado de atribuição (GAP) que trazia o algoritmo genético modificado e que ao ser aplicado ao problema, obtinha resultados de boa qualidade.

O problema GAP consiste em otimizar a atribuição de n tarefas para m agentes onde geralmente $n \gg m$. Cada agente tem recursos limitados. Neste tipo de problema, para o tipo de codificação mais usado, aparecem muitas propostas de solução infactíveis como conseqüência da implementação dos operadores genéticos e na geração da população inicial.

Por este motivo Chu-Beasley propõem um algoritmo genético modificado que pode ser resumido nos seguintes passos.

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

2. Especificar características genéticas do algoritmo, tais como, tipo de codificação, formação da população inicial, manipulação de infactibilidades, tipo de seleção, etc.

3. Encontrar uma população inicial de forma aleatória, que se transforma na população corrente. Encontrar o *fitness* e *unfitness* da população corrente.

4. Implementar a seleção para escolher apenas duas soluções geradoras.

5. Implementar a recombinação e preservar apenas um descendente.

6. Implementar mutação do descendente preservado.

7. Implementar uma fase de melhoria local do descendente preservado.

8. Decidir se o descendente melhorado pode entrar na população substituindo um elemento da população.

9. Se o critério de parada não for satisfeito voltar ao passo 4. Caso contrário terminar o processo.

Chu-Beasley apresentaram um algoritmo genético modificado que apresenta particularidades muito especiais. Esta seção analisa de forma resumida os aspectos mais relevantes do algoritmo genético de Chu-Beasley dando especial destaque para aquelas propostas que são significativamente diferentes de um algoritmo genético tradicional.

O algoritmo genético de Chu-Beasley segundo Alencar e outros colegas de pesquisa em (ALENCAR, 2004), sugere gerar a população de forma aleatória como nos algoritmos genéticos tradicionais. Entretanto, observa-se que essa proposta produz uma população inicial com todos os elementos infactíveis e muito distantes da factibilidade para o caso de problemas complexos de instâncias conhecidas do próprio problema GAP.

O algoritmo genético de Chu-Beasley apresenta uma proposta inovadora na manipulação de infactibilidades. Assim, apresenta-se a proposta de armazenar a função objetivo e as infactibilidades de forma separada e usada com propósitos diferentes. A proposta elimina a necessidade de escolher o parâmetro de penalização quando as duas informações são unidas em um único *fitness*. Portanto, Chu-Beasley sugerem armazenar a função objetivo de cada proposta de solução em um vetor *fitness* e as infactibilidades em um vetor *unfitness*. O *fitness* é usado no processo de seleção e o *unfitness* juntamente com o *fitness* no processo de substituição, isto é, para decidir se o descendente gerado deve ser incorporado na população, substituindo um elemento da população.

A proposta de Chu-Beasley é significativamente diferente aos algoritmos genéticos tradicionais no processo de substituição dos elementos da população. O algoritmo genético tradicional faz uma substituição geracional, substituindo todos (ou quase todos) os elementos da população e geralmente não faz uma verificação da diversidade. O algoritmo genético de Chu-Beasley sugere substituir, em cada passo, apenas um elemento da população corrente.

Essa proposta pretende facilitar duas estratégias cruciais no desempenho do algoritmo:

(*i*) permite produzir descendentes melhorados usando um processo de otimização local do descendente gerado e,

(*ii*) permite um controle absoluto da diversidade dos elementos da população corrente.

Essas duas propostas não podem ser implementadas com eficiência em um algoritmo genético tradicional com substituição geracional.

O algoritmo proposto por Chu-Beasley sugere também a implementação de uma fase de melhoria local do descendente gerado. Essa fase de melhoria local pode ser uma busca local muito simples ou pode ser uma estratégia sofisticada, que leva em conta as características específicas do problema.

Entretanto, a fase de melhoria local tem duas fases:

(*i*) fase de melhoria da infactibilidade e;

(*ii*) fase de melhoria da qualidade.

Assim, se o descendente gerado for infactível então, deve-se tentar tornar factível esse descendente, o que nem sempre é possível. Na seqüência, deve-se tratar de melhorar a qualidade do descendente procurando na vizinhança do descendente que está sendo avaliado. A proposta sugere ainda a substituição de um elemento apenas da população corrente pelo descendente gerado preservando a diversidade completa, isto é, todos os elementos da população corrente devem ser diferentes.

Portanto, se o descendente gerado for igual a um elemento da população, então esse descendente é descartado. Caso contrário o processo segue a seguinte estratégia:

(*i*) se o descendente gerado é infactível, então é verificado se a infactibilidade é menor que a infactibilidade do elemento da população com maior infactibilidade; caso

seja verdadeiro procede-se à substituição e caso contrário descarta-se o descendente gerado;

(*ii*) se o descendente gerado for factível, então deve-se substituir o elemento da população com maior infactibilidade. Logicamente se todos os elementos da população são factíveis, então deve-se verificar se o descendente gerado é de melhor qualidade que o elemento de pior qualidade da população, para que a troca seja possível.

Nesse contexto o *unfitness* é usado para ordenar os elementos da população que são infactíveis, é como "medida de infactibilidade" e o *fitness* representa apenas a função objetivo original do problema e é usado para ordenar propostas de solução factíveis, assim como no processo de seleção.

A proposta de Chu-Beasley para a substituição de uma configuração tem vários aspectos relevantes, tais como:

(*i*) todos os elementos da população são diferentes;

(*ii*) a lógica de substituição incrementa o número de elementos factíveis porque as propostas de solução infactíveis são as primeiras a serem descartadas, isto é, sempre que for gerada uma solução factível elimina-se uma proposta de solução infactível se ainda existir propostas de solução infactíveis na população corrente;

(*iii*) decorrente da observação anterior, o processo encontra uma população corrente apenas com soluções factíveis e esse estágio pode ser atingido em um tempo que depende do tipo de problema e da estratégia de melhoria local;

(*iv*) as melhores soluções sempre são preservadas porque em cada processo de substituição elimina-se apenas a solução de pior qualidade.

A última observação significa que a estratégia funciona melhor que o elitismo dos algoritmos genéticos tradicionais. Entretanto, a grande vantagem do algoritmo genético de Chu-Beasley é o controle absoluto da diversidade. Assim, em problemas altamente complexos e com grande dificuldade de encontrar soluções factíveis, pode ser interessante aumentar o tamanho da população permitindo armazenar soluções factíveis de composição genética diversificada.

4.4 Algoritmo Genético Especializado - Chu-Beasley Modificado

A estrutura tradicional do Algoritmo Genético (seleção, recombinação e mutação) é mantida no algoritmo proposto, mas algumas modificações fizeram-se necessárias para melhor atender o problema.

O algoritmo genético aplicado neste trabalho é baseado no Algoritmo Genético de Chu-Beasley, onde o controle da diversidade é mais acentuado do que o tradicional.

O Algoritmo Genético proposto possibilita o preenchimento do *container* utilizando torres, conceito introduzido por Gehring e Bortfeldt em (GEHRING; BORTFELDT, 1997).

As torres são geradas antes de qualquer etapa do algoritmo. Inicialmente o conjunto de torres está vazio e a cada iteração uma torre é adicionada.

As torres são formadas por uma caixa base (primeira caixa alocada) e por caixas que são colocadas uma após a outra no sentido vertical após a caixa base, sempre priorizando as caixas que têm dimensões iguais a caixa base.

Após todo o processo de geração do conjunto de torres, o algoritmo genético é aplicado.

Primeiramente é gerada a população inicial, em seguida um processo aqui chamado de "Decomposição do Espaço do *Container*" é aplicado na população inicial, este processo será detalhado na seção 4.4.7.

O resultado da decomposição do espaço do *container* são as torres que fazem parte do *container* e sua exata localização dentro dele, ou seja, o resultado são as torres que estão inteiramente alocadas no interior do *container*.

Após o processo de decomposição do espaço do *container* a avaliação da população é realizada, em seguida são selecionados dois cromossomos e realiza-se recombinação, gerando e preservando somente um descendente, logo em seguida a mutação é aplicada no descendente gerado.

O descendente somente fará parte da população, dando origem a nova geração, se existir um cromossomo pior que ele na população corrente, e existindo um pior, este descendente deve ser diferente dos outros cromossomos da população corrente.

O algoritmo genético Chu-Beasley proposto, trabalha com dois critérios de parada, depois de n gerações sem melhoria na incumbente ou pára quando atingir 100% de volume ocupado no *container*. Este resultado seria o ideal, mas muito difícil de ser alcançado.

A seguir, apresenta-se na Figura 14 o fluxograma do funcionamento do algoritmo genético Chu-Beasley proposto.

Portanto, o algoritmo genético proposto segue a estrutura do algoritmo genético tradicional (geração da população inicial, avaliação das configurações, seleção, recombinação e mutação), mas trabalhando apenas com um único descendente para ser aproveitado na nova população, caracterizando-se assim o algoritmo genético Chu-Beasley.

A seguir temos a definição mais detalhada do algoritmo genético proposto e das etapas envolvidas.

4.4.1 Geração do Conjunto de Torres

O conceito de torres como foi citado anteriormente, foi introduzido por Gehring e Bortfeldt em (GEHRING; BORTFELDT, 1997).

As torres são geradas antes de qualquer etapa do algoritmo genético.

Para a construção de uma torre é necessário inicialmente escolher aleatoriamente uma caixa que esteja disponível para uso. Deve-se considerar que no início do processo todas as caixas fazem parte de um conjunto que será chamado de "Conjunto de Caixas Disponíveis".

Esta primeira caixa escolhida, será chamada de "caixa base" e sua largura de "largura da base" ou "largura da torre", seu comprimento será chamado de "comprimento da base" ou "comprimento da torre" e a área da caixa base será chamada de "área da base" ou "área da torre". Isto ocorre porque a caixa base estabelece dimensões de largura e comprimento para a torre, toda e qualquer caixa alocada acima da caixa base deve respeitar tais dimensões.

Portanto, a área da base deverá ser respeitada para que as caixas que serão colocadas acima da caixa base observem seu perímetro, ou seja, toda caixa colocada acima de outra na composição de uma torre, deve ter sua área de contato igual ou menor que a área da caixa que está imediatamente abaixo dela. Isto ocorre para que uma torre não invada o espaço de outra ao serem colocadas no interior do *container*. Deve-se também, seguir um critério de seleção de qual caixa será colocada em cima de outra.

O critério de seleção de caixas para a formação de torres é baseado na heurística proposta no capítulo 3, onde caixas iguais são alocadas juntas procurando minimizar as sobras.

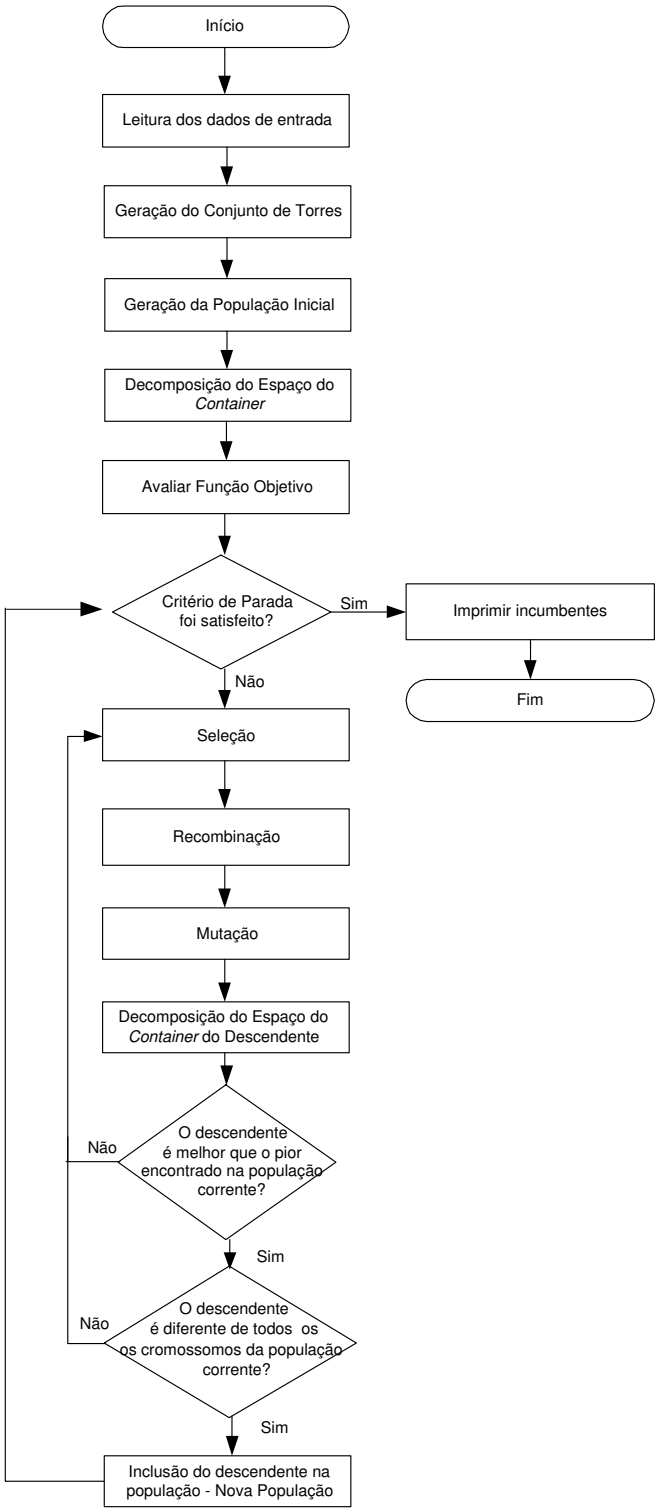

Figura 14: Fluxograma do Algoritmo Genético Chu-Beasley Proposto.

Logo abaixo é descrito passo a passo o processo de criação de torres:

1. Primeiramente é realizada uma escolha aleatória da caixa base, primeira caixa da torre, deve ser considerado que a caixa base será sempre alocada na origem do *container*.

2. A caixa é rotacionada visando a minimização primeiramente do espaço restante na lateral direita do *container*, então a dimensão que retornar a menor sobra é fixada como largura da caixa.

3. Depois, nesta ordem de prioridade, rotacionam-se as duas dimensões livres para que minimize o espaço restante superior. Então a dimensão que retornar a menor sobra é fixada como altura da caixa.

4. E por último, na ordem de prioridade, a única dimensão livre da caixa é atribuída ao comprimento da caixa.

5. A partir desta posição fixa, outras caixas com dimensões iguais serão colocadas uma em cima da outra até que não haja mais caixas disponíveis e iguais a caixa base ou até que o espaço acima seja minimizado, não permitindo a alocação de mais outra caixa, formando assim uma torre.

6. Logo após a formação da torre é verificado se o espaço residual na parte superior da torre é excessivo. Será considerada sobra excessiva se esta ultrapassar cerca de 4% da altura do *container*. Se não for constatado sobra excessiva deve-se ignorar os passos relacionados a ela.

7. Verifica-se a sobra na parte superior da torre, se esta sobra for maior que 4% da altura do *container* é verificado se esta sobra é maior ou igual ao tamanho da menor dimensão de caixa disponível para o carregamento, e se esta caixa que se enquadra na sobra têm dimensões menores que a dimensão da largura e do comprimento da caixa base. Se esta condição for satisfeita, não serão executados os passos a partir do passo 8.

8. Para um resultado negativo da condição acima, este passo deve ser executado, se a sobra for maior que 4% da altura do *container*, mas o tamanho da sobra é menor que o tamanho da menor dimensão de caixa encontrada avaliando os tipos de caixas disponíveis, ou a caixa que se enquadra na sobra têm dimensões maiores que a caixa base, deve-se escolher a outra dimensão que está livre, ou seja, a altura da caixa é trocada pelo comprimento.

9. Verifica-se a sobra na altura da torre com esta nova dimensão na altura da caixa, se esta for maior que 4% da altura do *container* é verificado se a sobra é maior ou igual ao tamanho da menor dimensão de caixa disponível para o carregamento e se esta caixa que se enquadra na sobra têm dimensões menores que a dimensão da largura e do comprimento da caixa base. Se esta condição for satisfeita não será executado o passo 10.

10. Caso as tentativas anteriores não forem satisfeitas, deve-se retirar a última caixa colocada acima das outras e voltar ao passo 7.

11. Se mesmo com a tentativa acima, de retirar a caixa que estava mais acima que as outras, a condição dos 4% não for satisfeita, a primeira configuração de torre criada é incluída no conjunto de torres.

A seguir tem-se a Figura 15 que ilustra o processo de formação de uma torre.

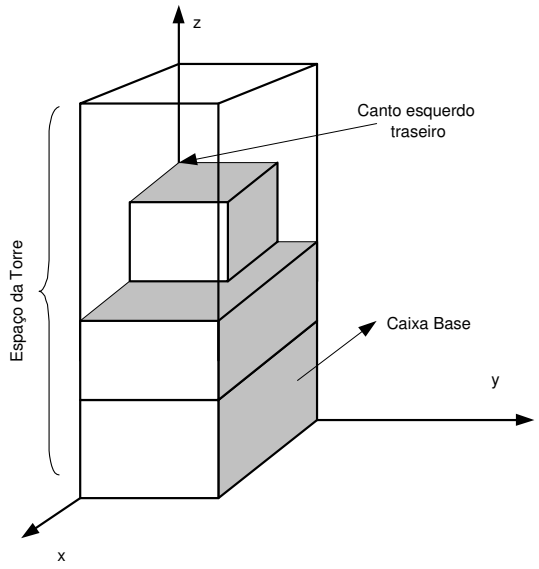

Figura 15: Processo de Geração de Torres.

Várias torres são geradas repetindo tal processo até que não haja mais caixas disponíveis para se criar uma torre. As torres geradas compõem o "Conjunto de Torres". Inicialmente este conjunto deve estar vazio.

O conjunto de caixas disponíveis será atualizado toda vez que uma torre for gerada, ou seja, as caixas que fazem parte da torre que acaba de ser gerada é eliminada do conjunto de caixas disponíveis.

Ao final do processo de geração de torres é eliminada do conjunto de torres a torre que não obteve um bom aproveitamento de seu volume. Torres que foram formadas por apenas uma caixa são eliminadas do conjunto de torres. Considerando que o volume total ocupado por uma torre é a área da base multiplicada pela altura do *container* e o volume realmente utilizado pelo conjunto de caixas que compõem a torre é a soma de todos os volumes das caixas que compõem esta torre.

Após todo o processo de geração do conjunto de torres, o algoritmo genético é aplicado.

A seguir a Figura 16 traz o fluxograma do processo de geração de torres.

4.4.2 Codificação e População Inicial

Cada cromossomo da população, especificamente neste problema, será chamado de padrão de carregamento.

A representação do padrão de carregamento (P) é determinada a partir de um cromossomo da população, formado por índices de torres t_i: $P = t_1, t_2, t_3, ..., t_n$, em que $i = 1, ..., n$ são índices das torres.

A seqüência $t_1, t_2, ..., t_n$ representa a ordem com que as torres devem ser posicionadas no *container*, seguindo as regras de preenchimento descrita na seção 4.4.7.

A população inicial é gerada aleatoriamente, garantindo que o tamanho dos cromossomos seja determinado pelo número de torres geradas, portanto não será permitido a repetição de índices em um mesmo cromossomo. Esta restrição deve ser observada na geração da população inicial, para que uma torre conste somente uma única vez dentro do *container*.

A seguir, a Figura 17 mostra como seria um cromossomo da população com a codificação proposta.

Cada índice de torre que compõe o cromossomo é vinculado com outro vetor auxiliar onde constam quais caixas fazem parte desta torre, como mostra a Figura 18 logo na seqüência.

O vetor auxiliar de caixas é formado por índices de caixas b_j, exemplificando: $B = b_1, b_2, b_3, ..., b_m$, em que $j = 1, ..., m$ são índices das caixas.

A seqüência $b_1, b_2, ..., b_m$ representa a ordem com que as caixas foram alocadas na torre.

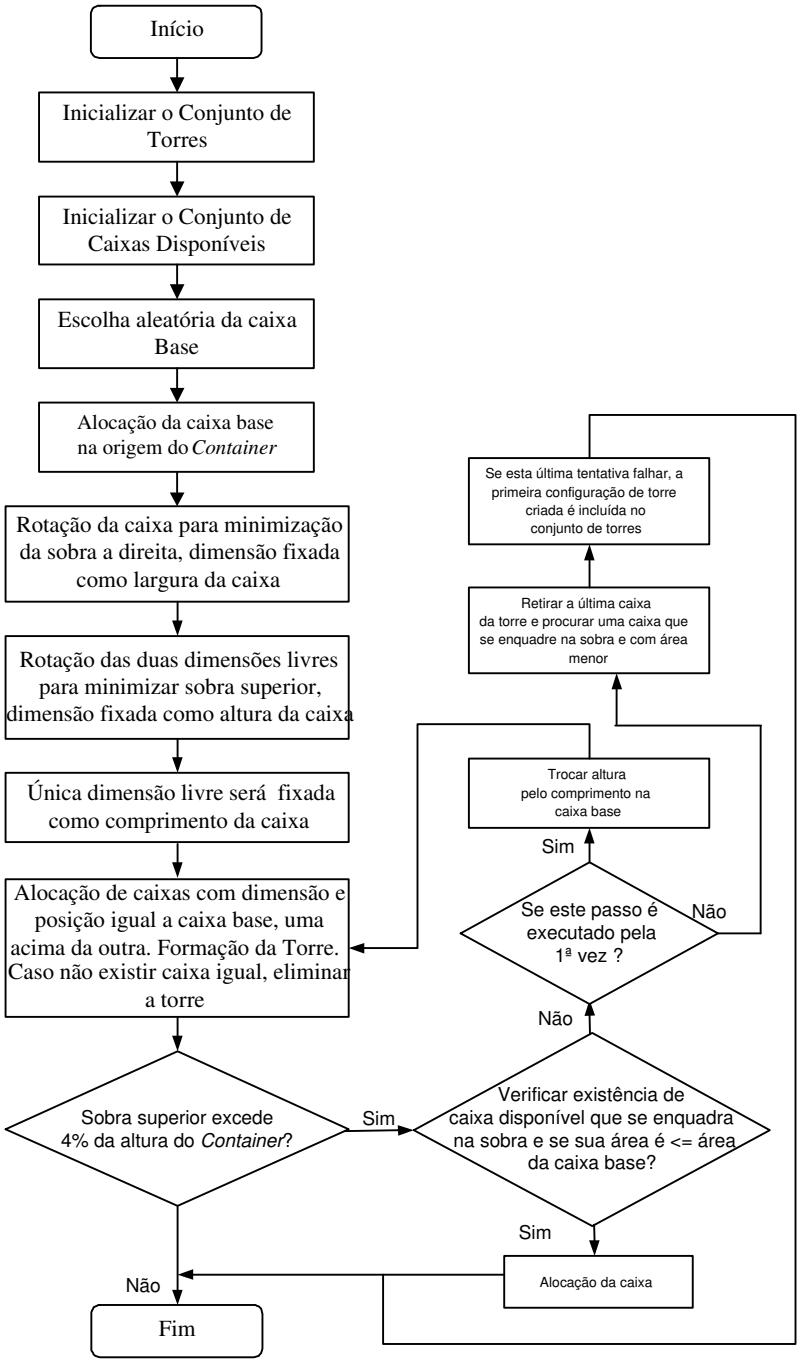

Figura 16: Fluxograma do Processo de Geração de Torres.

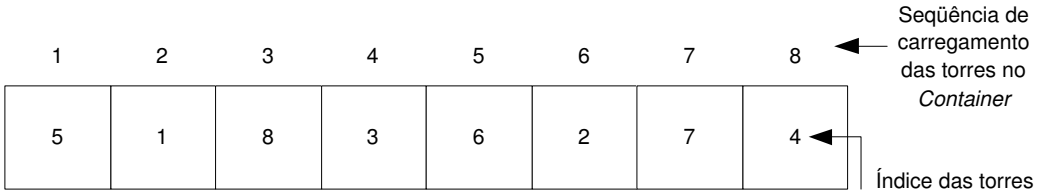

Figura 17: Codificação do Cromossomo.

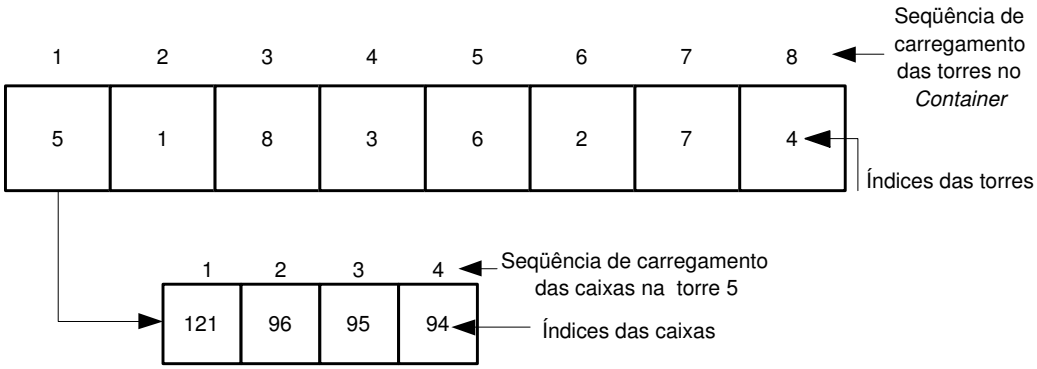

Figura 18: Padrão de Carregamento e vetor auxiliar com o índice das caixas.

Os índices das caixas também representam os tipos de caixas, pois se temos x tipos de caixas, e cada tipo de caixa B tem m_i caixas o representado, então as caixas com índices de 1 a m_1 representam as caixas do tipo 1, as caixas com índices de $m_1 + 1$ a $m_1 + m_2$ representam as caixas do tipo 2 e assim sucessivamente, até completar os índices das caixas disponíveis para o carregamento.

A Tabela 3, traz de maneira simples e didática como identificar o tipo de cada caixa.

Tabela 3: Identificação dos Tipos de Caixas.

Tipo	**Qtde. Caixas**	**Intervalo de Tipos de Caixas**
1	2	1 a 2
2	3	3 a 5
3	5	6 a 10

Além do vetor onde é representado o padrão de carregamento e do vetor auxiliar de caixas de cada torre, a codificação do problema estudado necessita de outro vetor auxiliar onde é sinalizado como as caixas foram rotacionadas e alocadas na torre. A posição das caixas que foram alocadas em uma torre tem sua identificação através de índices. A seguir temos os índices para cada orientação que a caixa pode ter dentro de uma torre.

- Se a largura da caixa está apoiada na base, e a altura e profundidade não variam, então o índice que representa esta posição de caixa é igual a 1.

- Se a altura da caixa está apoiada na base, e a largura da caixa no lugar da altura, com profundidade sem variar, então o índice que representa esta posição de caixa é igual a 2.

- Se a profundidade está apoiada na base, a altura sem variar e a largura da caixa está no lugar da profundidade, então o índice que representa esta posição de caixa é igual a 3.

- Se a largura está apoiada na base, e a altura e profundidade trocam de lugar entre si, então o índice que representa esta posição de caixa é igual a 4.

- Se a altura está apoiada na base, a profundidade está no lugar da altura e a largura no lugar da profundidade, então o índice que representa esta posição de caixa é igual a 5.

- Se a profundidade está apoiada na base, a largura está no lugar da altura e a altura está no lugar da profundidade, então o índice que representa esta posição de caixa é igual a 6.

Finalizando, outros dois vetores auxiliares são necessários para a decodificação do problema, são eles: vetor de vértices x referente ao comprimento da torre e vetor de vértices y referente à largura da torre.

Cada cromossomo é vinculado à dois vetores auxiliares. Estes vetores trazem exatamente as coordenadas (x, y) ocupadas pelos vértices de cada torre dentro do *container*.

A seguir a Figura 19 mostra como os vetores são vinculados.

Portanto depois que o algoritmo genético proposto gerar a população inicial e aplicar o processo de decomposição do espaço do *container* são gerados outros vetores complementares ao padrão de carregamento simplificando seu processo de decodificação.

4.4.3 Cálculo da Função Objetivo

O mesmo tipo de avaliação que ocorre com a heurística proposta no capítulo anterior, acontece neste capítulo, aqui tem-se as mesmas propostas para a avaliação do padrão de carregamento.

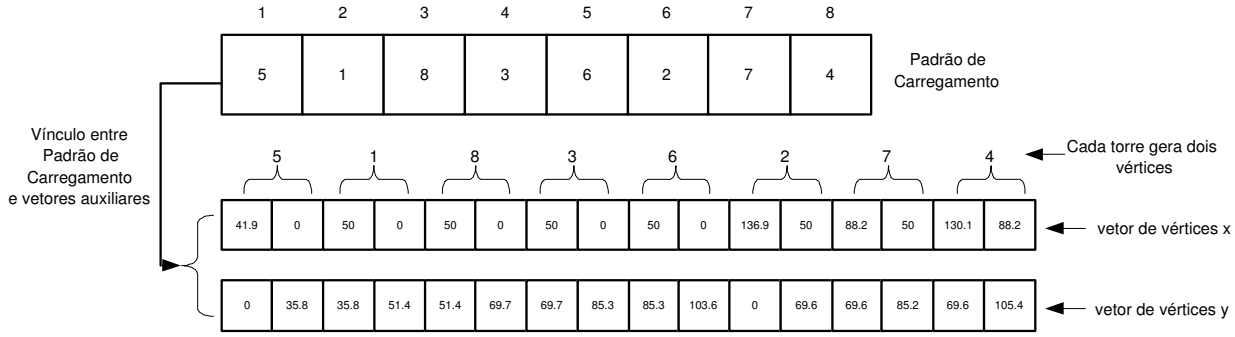

Figura 19: Ligação entre Padrão de Carregamento e Vetores Auxiliares de Coordenadas (x, y).

No algoritmo genético proposto tem-se o objetivo de maximizar o volume da carga alocada no *container*, cálculo que pode ser encontrado na seção 3.2.2.1 do capítulo anterior e também tem-se outro objetivo, mais completo, o de maximizar volume, peso, centro de gravidade e valor da carga carregada, como foi citado no capítulo anterior na seção 3.2.2.2. Este segundo tipo de avaliação foi proposta por Rodrigues em (RODRIGUES, 2005) e foi aplicada no trabalho para fins de comparação.

4.4.4 Seleção

O método de seleção escolhido para ser aplicado no problema foi o método de seleção por torneio com $j = 2$, ou seja, é escolhido para participar de cada jogo dois padrões de carregamento da população corrente. A função objetivo já está armazenada e o padrão de carregamento que detém o maior valor é o escolhido.

4.4.5 Recombinação

A recombinação para este tipo de problema não pode ser aplicada com o método tradicional, como é costume ver em aplicações com algoritmo genético, onde são escolhidos aleatoriamente dois cromossomos da população corrente, para serem recombinados a partir de um ponto também aleatoriamente escolhido e gerarem dois descendentes.

Se aplicássemos este tipo de recombinação correríamos o risco de uma configuração ter uma mesma torre ocupando espaços diferentes no mesmo *container*, ou seja, ter elementos com valores repetidos dentro de um cromossomo, que representa uma proposta de solução na população corrente.

Optou-se portanto, em trabalhar com a recombinação PMX (*Partially Matched Crossover*) como a própria tradução diz "recombinação parcialmente combinada". Este tipo de operador genético recombina dois cromossomos sem perder o que já havíamos conquistado (nenhum valor repetido na codificação dos cromossomos), conforme Díaz, Glover e Ghazini em (DÍAZ et al., 1996) e Laguna em (LAGUNA, 2001).

Esta recombinação trabalha com dois cromossomos previamente escolhidos. É escolhido aleatoriamente o tamanho da faixa que será recombinada e onde a recombinação irá começar.

No algoritmo proposto os dois cromossomos geram somente um descendente e é escolhido aleatoriamente em qual dos dois cromossomos a recombinação será executada para a geração deste descendente. A seguir a Figura 20 mostra o funcionamento da recombinação PMX.

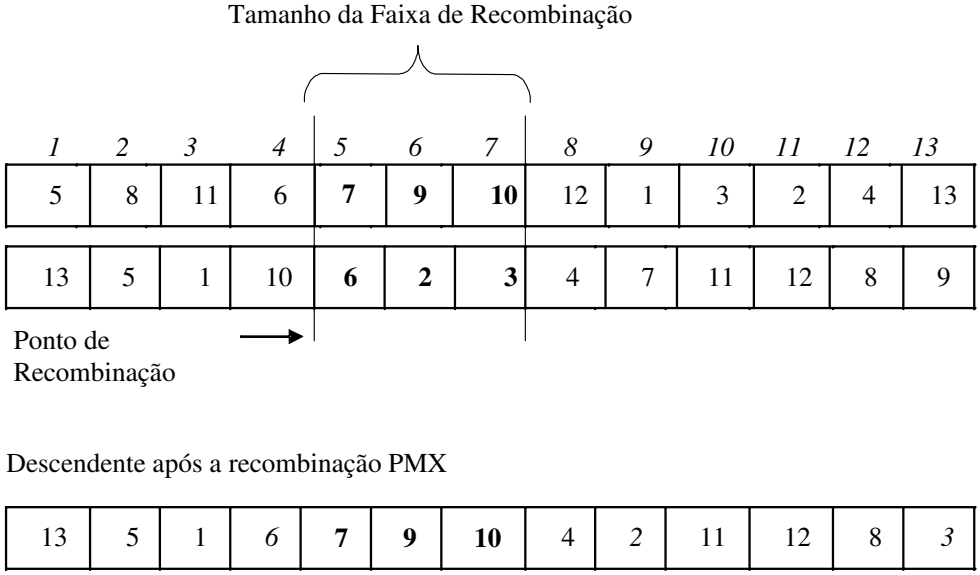

Figura 20: Exemplo de recombinação PMX.

4.4.6 Mutação

A mutação se dá de uma maneira muito simples. São escolhidas aleatoriamente duas torres que fazem parte do cromossomo descendente, gerado na etapa de recombinação. As torres escolhidas são trocadas de posição na seqüência de carregamento de torres do *container*, como mostra a Figura 21 a seguir.

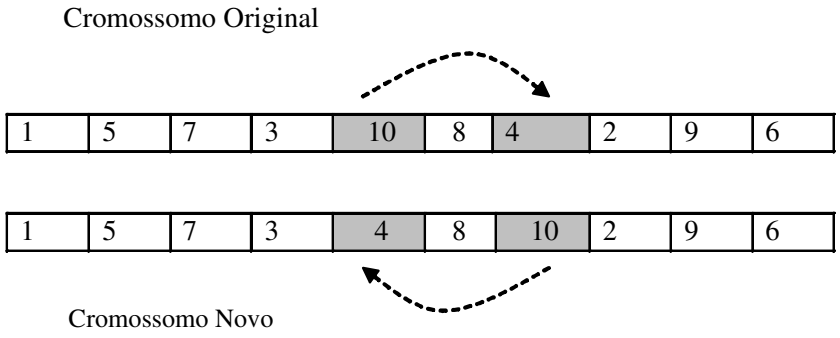

Figura 21: Operador de Mutação.

4.4.7 Decomposição do Espaço do *Container*

O processo de decomposição do espaço do *container* é uma particularidade exigida pelo problema de carregamento do *container*.

Após o processo de geração da população inicial, onde cada cromossomo terá o tamanho da quantidade de torres geradas e cada gene do cromossomo contém um índice de uma torre que indica a ordem que as torres será alocadas no *container*, o algoritmo executa uma etapa que será chamada de decomposição do espaço do *container*.

Nesta etapa, as torres serão alocadas no *container* num processo sistemático seguindo regras de alocação. A ordem com que as torres são alocadas seguem a seqüência dos genes do cromossomo.

Antes de iniciar o processo de decomposição do espaço do *container* é necessário rotacionar todas as torres deixando a maior dimensão para o comprimento da torre e sua menor dimensão para a largura da torre. Feita esta correção, o processo de decomposição do espaço do *container* pode ser iniciado.

Na decomposição do espaço do *container* são utilizados vários conceitos introduzidos por Gehring e Bortfeldt em (GEHRING; BORTFELDT, 1997), um destes conceitos é o de "*container* virtual" e "*container* real". O *container* virtual tem seu comprimento infinito, possibilitando com isso a alocação de todas as torres, já o *container* real como o próprio nome diz, trabalha com as medidas reais do *container*, portanto só as torres que estão com suas áreas inteiramente dentro do *container* real é que farão parte da solução do problema.

Outros dois conceitos usados na decomposição do espaço do *container* são os vértices de alocação e larguras residuais. Quando trabalha-se com vértices de alocação, intuitiva-

mente considera-se o chão do *container* um plano cartesiano onde o eixo x é o comprimento do *container* e o eixo y a largura do *container*. Os vértices de alocação são os dois cantos criados quando uma torre é colocada no interior do *container*.

Inicialmente o único vértice de alocação do *container* (plano cartesiano) é o de coordenadas $(0, 0)$, ou seja, a origem do *container*. Ao alocar uma torre no *container* nestas coordenadas, esta deixa de existir e outras duas coordenadas são criadas, as coordenadas (x_1, y_1) e (x_2, y_2). As coordenadas (x_1, y_1) são na verdade $(comprimentodatorre, 0)$ e as coordenadas (x_2, y_2) são $(0, larguradatorre)$.

A cada torre alocada, o vértice onde esta torre se acomodou é excluído do conjunto de vértices de alocação e outros dois são inseridos neste conjunto. Além disso, toda alteração que ocorre no conjunto de vértices de alocação, faz com que o conjunto seja ordenado novamente. A classificação do conjunto de vértices de alocação é crescente em relação ao eixo x, ou seja, em cada alteração do conjunto, a classificação acontece, ordenando os vértices que possuem menor coordenada x primeiro e os que possuem maior coordenada x ficam para o final.

O conceito de largura residual caminha em conjunto com os vértices de alocação. Toda vez que ocorre a alocação de uma torre, esta produz uma alteração na largura disponível a sua direita. Todo vértice de alocação criado ao acomodar uma torre no *container* terá uma distância entre a posição que ele se encontra em relação ao eixo y e um limitante, este limitante pode ser o lado direto do *container* ou outra torre que foi colocada entre uma torre já alocada e o lado direto do *container*.

As Figura 22 e 23 ilustram os conceitos de *container* virtual e *container* real, além de mostrar a alocação da primeira torre no interior do *container* e a posição dos vértices de alocação com suas respectivas larguras residuais.

A cada alocação de torre no *container*, procura-se acomodar esta torre no primeiro vértice de alocação do conjunto de vértices, ou seja, a posição onde a torre será alocada depende do ordenamento dos vértices de alocação. A torre não será colocada no primeiro vértice de alocação do conjunto, somente se sua largura for maior que a largura residual do vértice onde ela foi acomodada.

As Figuras 24, 25 e 26 logo abaixo ilustram com maiores detalhes a situação descrita no parágrafo anterior.

Como é possível observar na Figura 26, após a alocação da terceira torre e atualização do conjunto de vértices, a largura residual do primeiro vértice é menor que a largura da

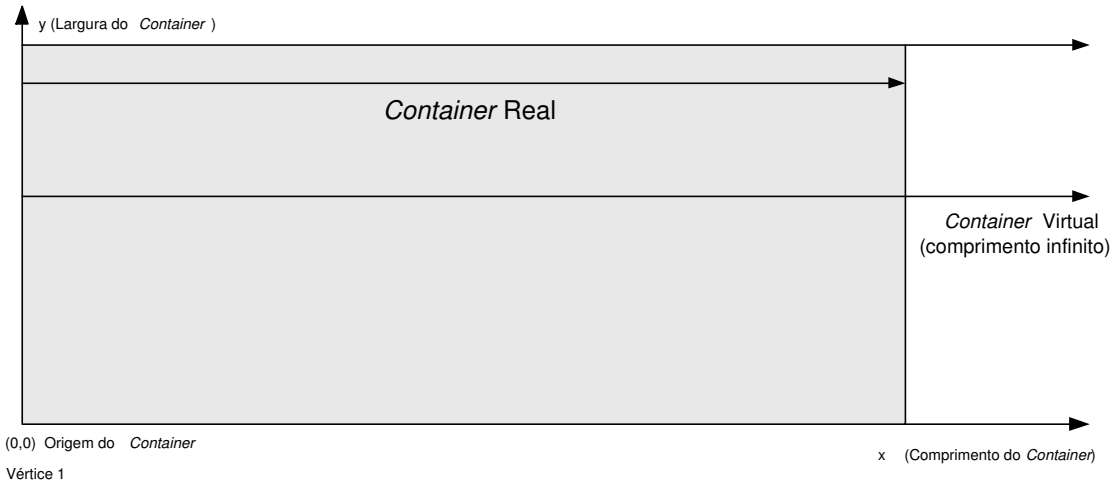

Figura 22: Situação do *container* antes de alocar a primeira torre.

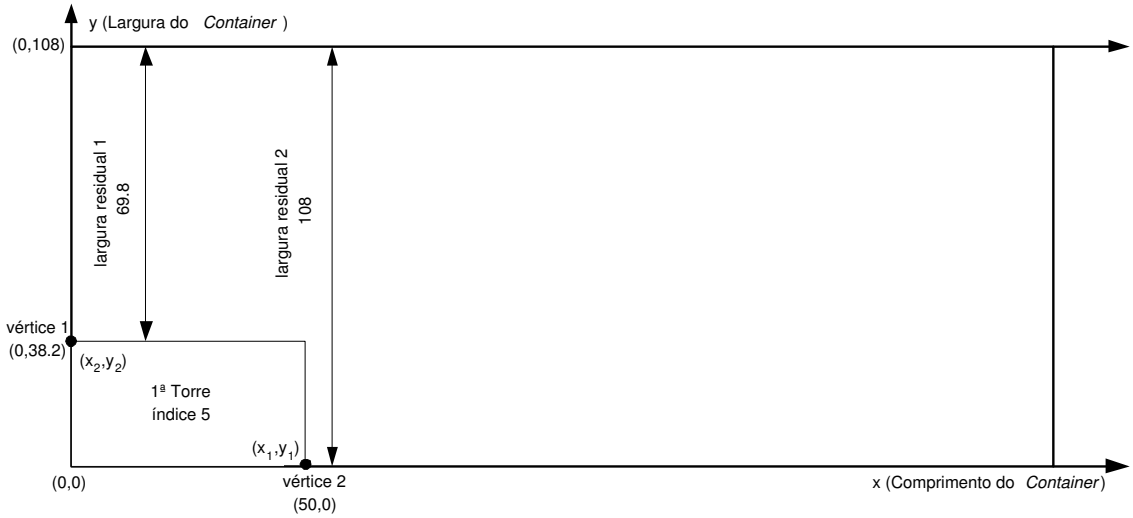

Figura 23: Situação do *container* depois de alocar a primeira torre.

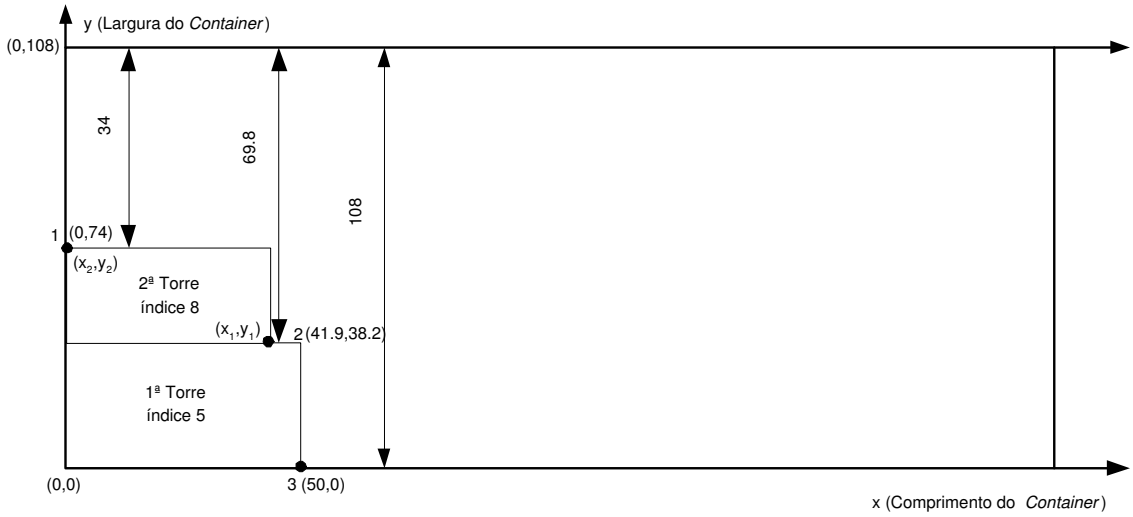

Figura 24: Situação do *container* após a alocação da segunda torre.

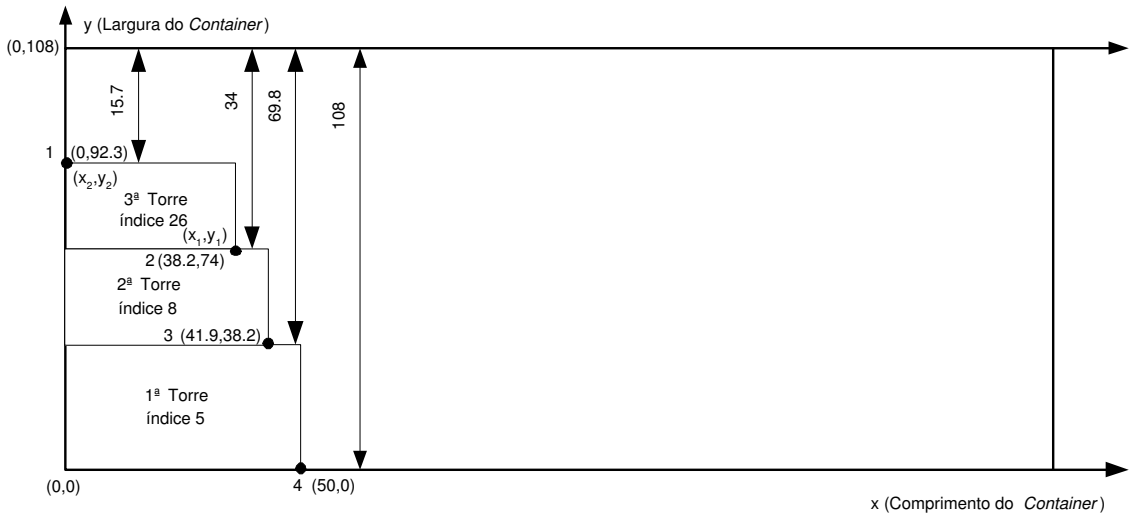

Figura 25: Situação do *container* após a alocação da terceira torre.

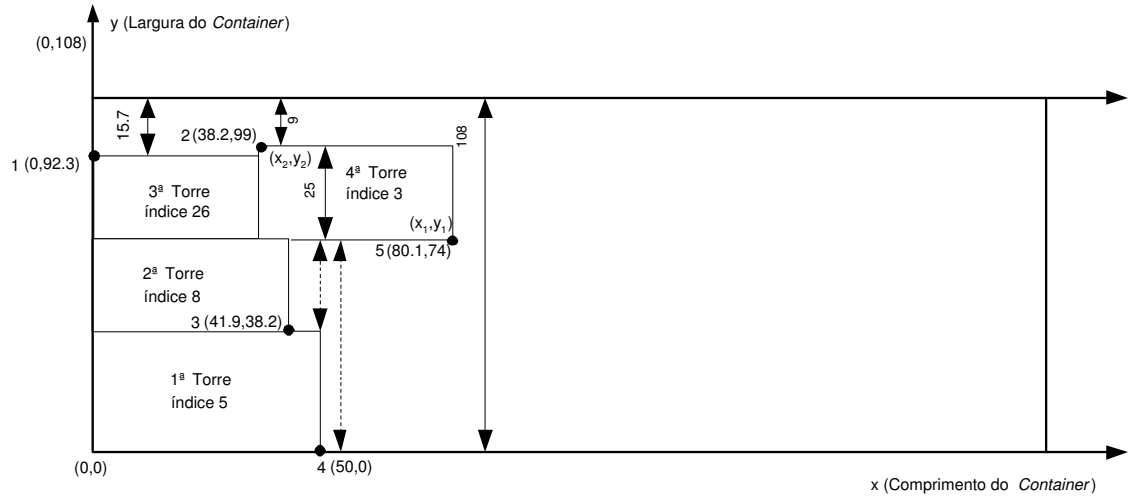

Figura 26: Situação do *container* após alocação de varias torres.

quarta torre. Isto implica que a quarta torre não poderá ser alocada no primeiro vértice do conjunto. Verificando que a quarta torre tem largura menor que a largura residual do segundo vértice, esta torre finalmente poderá ser alocada. O vértice onde a quarta torre foi acomodada é eliminado do conjunto de vértices e os novos vértices gerados são incluídos no conjunto.

Observou-se também que, ao alocar a quarta torre no *container*, as larguras residuais do terceiro e quarto vértices foram obstruídas, com isto, deve ocorrer a atualização das larguras residuais obstruídas.

A Figura 27 abaixo mostra o resultado da atualização.

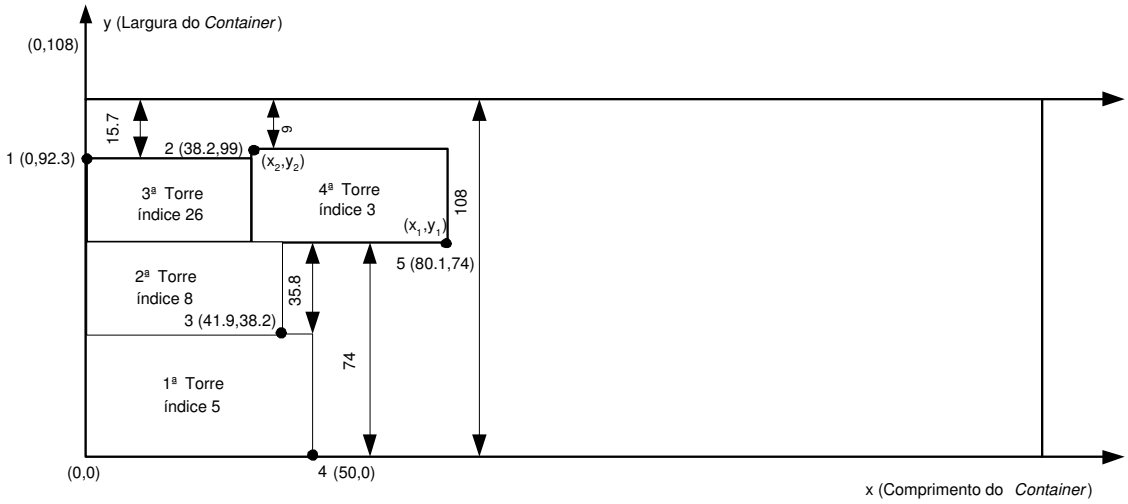

Figura 27: Atualização de larguras residuais obstruídas.

Outra regra que deve ser observada neste processo sistemático de alocação diz respeito ao remanejamento de vértices. Quando um vértice de alocação se encontra na situação do quinto vértice da Figura 27 é necessário fazer um remanejamento do vértice para a menor coordenada y possível. Explicando com maiores detalhes, esta regra funciona da seguinte maneira: o vértice que não tem apoio em sua esquerda com o lado de outra torre ou o próprio *container*, deverá ser deslocado para a esquerda em relação ao eixo y até que se encontre um limitante, sendo este limitante, o lado direto de outra torre ou o próprio *container*. O vértice terá portanto sua coordenada y alterada para a posição deslocada.

A Figura 28 ilustra o deslocamento do quinto vértice para a esquerda do *container* até encontrar seu primeiro limitante. Deve-se observar que o vértice é deslocado para a menor coordenada encontrada em y, ou seja, atingir seu primeiro limitante, no caso da Figura 28 o limitante será o lado esquerdo do *container*, mas em outra situação, poderia ser uma torre com coordenada x_1 maior que o vértice que está sendo deslocado.

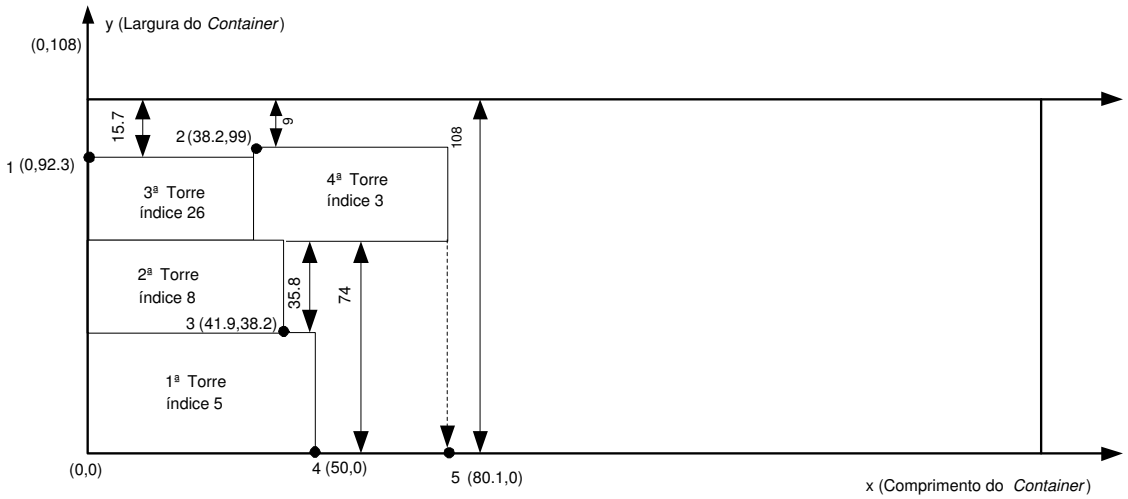

Figura 28: Deslocamento de Vértice para a esquerda.

Ao observar mais atentamente a Figura 28, nota-se que o vértice 1 foi obstruído com a alocação da quarta torre, neste caso este vértice deve ser excluído do conjunto de vértices. Isto ocorre porque o processo de decomposição do espaço do *container* não trabalha com o controle do comprimento residual, somente com a largura residual. Após excluir o primeiro vértice, o conjunto de vértices fica ordenado da seguinte maneira:

- Vértice 1(38.2,99);
- Vértice 2(41.9,38.2);
- Vértice 3(50 ,0);
- Vértice 4(80.1,0);

Como mostra a Figura 29 logo a seguir.

Quando um vértice é remanejado para a esquerda do eixo y está sujeito a obstrução por uma torre. Veja a Figura 30. Na ilustração fica claro a obstrução do vértice remanejado pela quinta torre que acaba de ser alocada no interior do *container*.

Neste caso se aplica outra regra de alocação. Ao acomodar a quinta torre no *container*, o vértice que foi deslocado para a esquerda do eixo y e que sofreu obstrução deve ser novamente remanejado, mas agora para a direita, na mesma posição do eixo y do segundo vértice gerado (y_2) pela torre que acabou de ser alocada. Veja na Figura 31 o novo remanejamento do vértice.

Ao observar mais atentamente a Figura 31, nota-se que o vértice 2 também foi obstruído com a alocação da quinta torre, neste caso este vértice deve ser excluído do conjunto

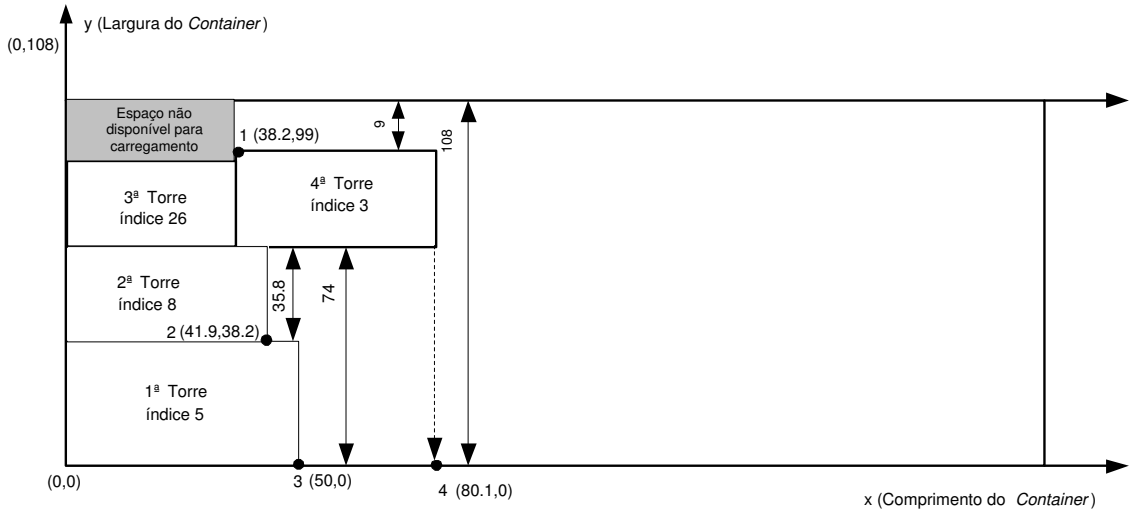

Figura 29: Exclusão do vértice obstruído.

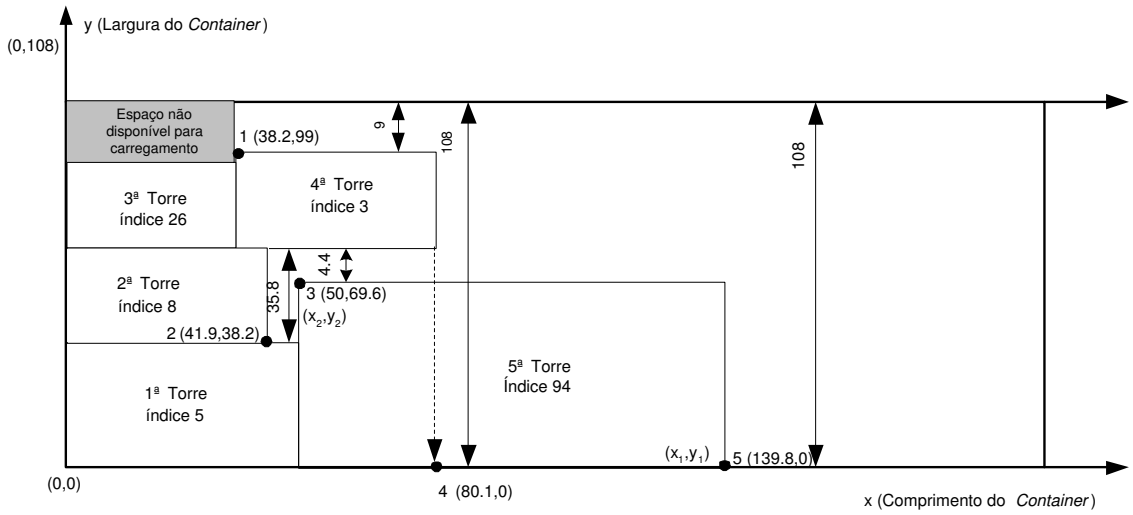

Figura 30: Obstrução do vértice remanejado.

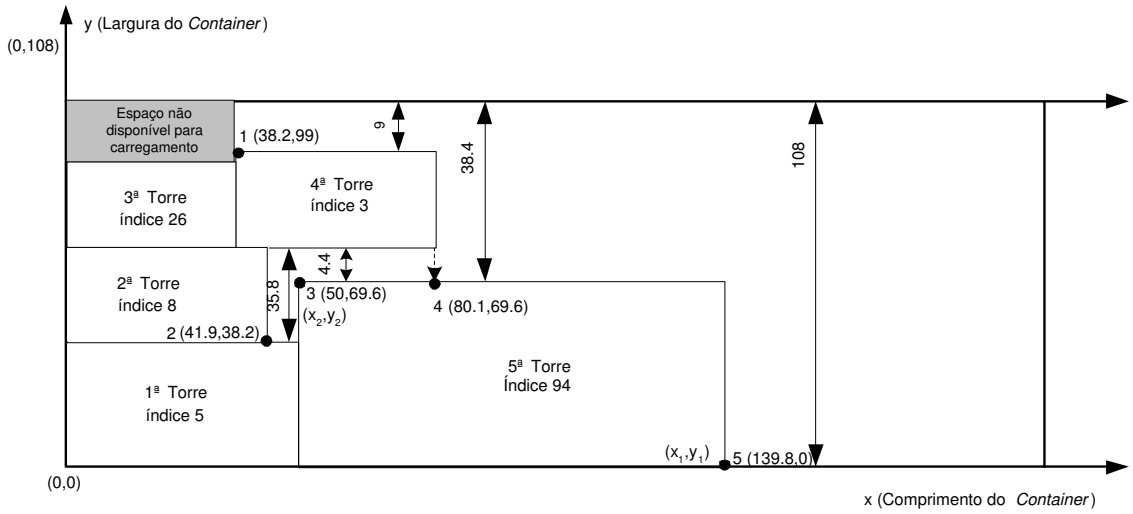

Figura 31: Remanejamento do vértice para a direita.

de vértices. Após excluir o segundo vértice, o conjunto de vértices fica ordenado como mostra a Figura 32 a seguir.

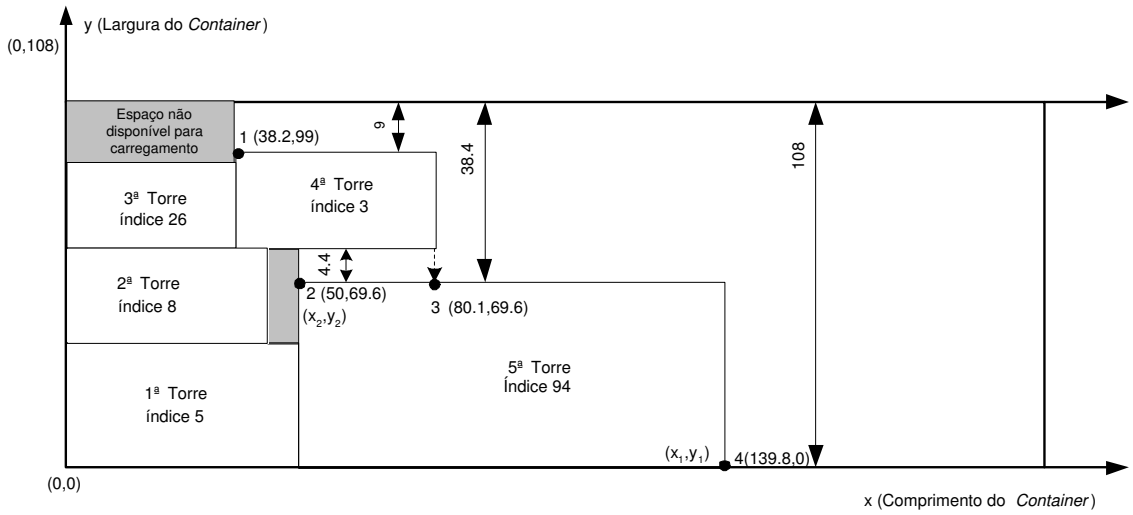

Figura 32: Exclusão de outro vértice obstruído.

Outra regra de alocação que deve ser observada é o deslocamento de uma torre para a direita com o objetivo de encontrar maior área de contato. Quando uma torre alocada no *container* tem seu vértice y_1 remanejado para a esquerda do eixo y, esta torre tem um apoio muito pequeno a sua esquerda e portanto possibilita a verificação se a sua direita encontrará maior área de contato. Se a torre encontrar maior área de contato a direita, esta torre é transladada para a direita até encostar em outra torre. Quando a translação da torre ocorre, o vértice onde ela foi alocada no primeiro momento deve ser novamente inserido no conjunto de vértices e a atualização das coordenadas dos vértices da torre transladada também deve ocorrer. A Figura 33 e 34 exemplifica a situação descrita acima.

O processo de decomposição do espaço do *container* só termina quando forem alocadas no *container* virtual todas as torres que compõem um cromossomo. Sabe-se que o *container* virtual tem seu comprimento infinito por este motivo é possível alocar todas as torres neste *container*, mas a avaliação de cada proposta de solução somente utilizará as torres que fazem parte do *container* real, ou seja, a função objetivo somente utilizará para sua avaliação as torres que tem suas áreas inteiramente dentro do *container* real.

A Figura 35 abaixo mostra quais torres fazem parte do *container* real.

Após a decomposição do espaço do *container* cada proposta de solução é avaliada pela função objetivo, em seguida a seleção, recombinação e mutação são aplicadas.

Este processo será melhor detalhado na seção 4.4.10.

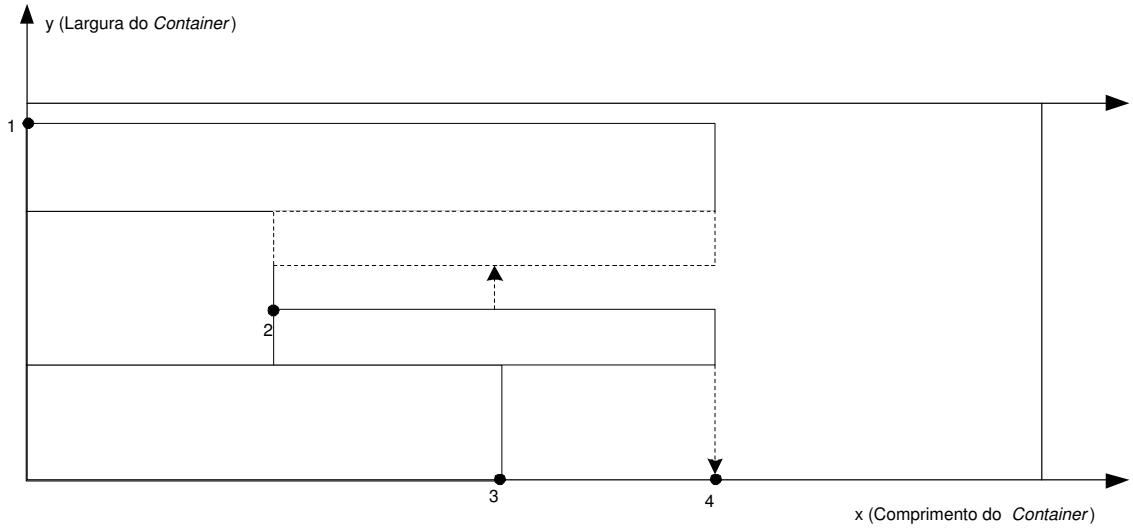

Figura 33: Situação antes de deslocar a torre para obter maior área de contato.

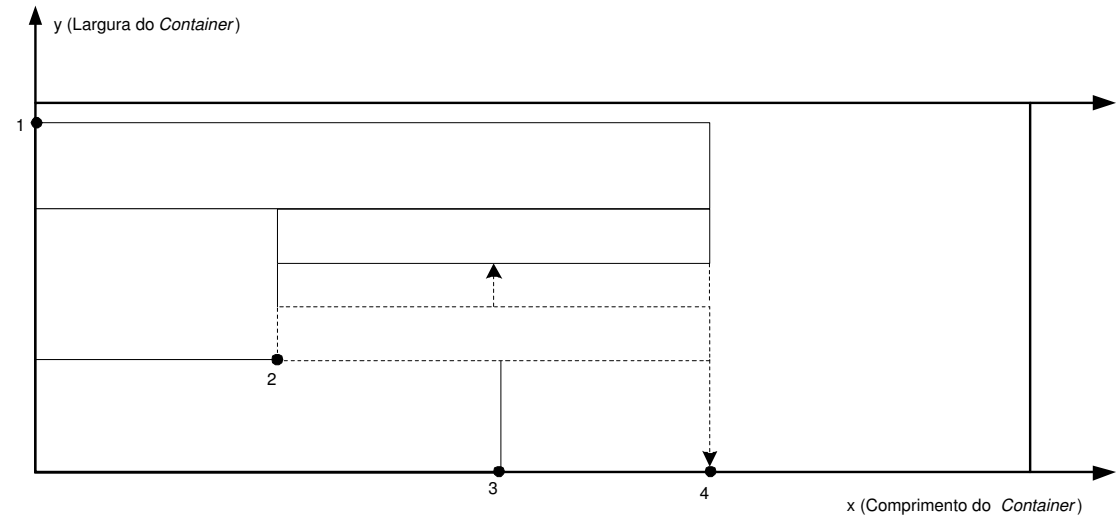

Figura 34: Situação após deslocar a torre para obter maior área de contato.

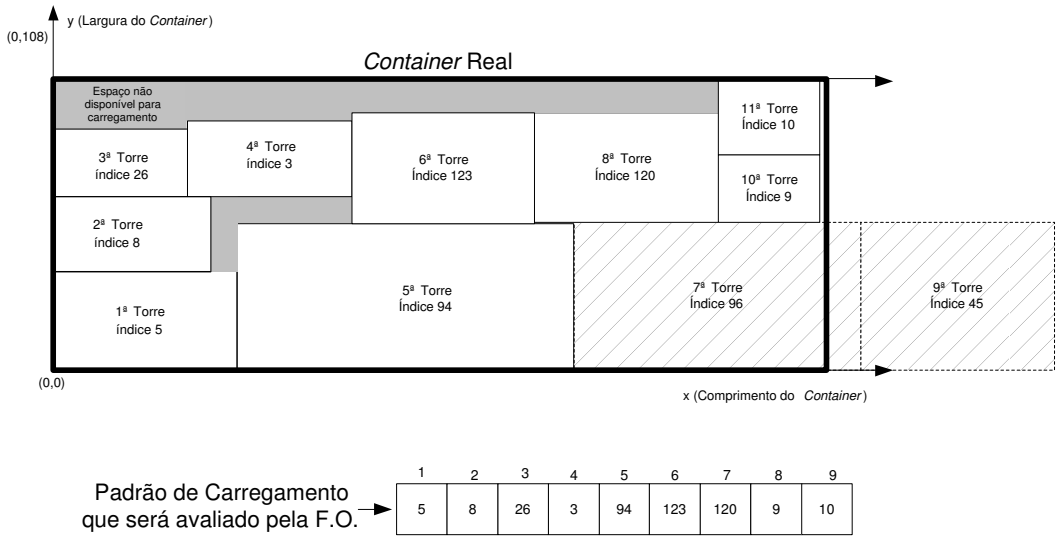

Figura 35: Torres que compõem o *container* real.

4.4.8 Critério de Parada

O algoritmo pára quando a incumbente relacionada ao volume carregado no *container* não melhorar por mais de 20 gerações ou quando a incumbente do volume encontrar o valor de 100% de volume ocupado.

4.4.9 Estrutura Geral do Algoritmo Genético Chu-Beasley Proposto

1. Leitura das caixas disponíveis para o carregamento, bem como as medidas das caixas, os tipos de caixas, a quantidade de caixas de cada tipo e as dimensões do *container*.

2. Geração do conjunto de torres. Cada torre procura alocar uma caixa de cada vez. Ao escolher cada caixa deve-se levar em consideração qual caixa que têm dimensões iguais a da caixa base. Caso não seja possível, priorizar a caixa que tem maior contato com a caixa que vem imediatamente abaixo.

3. Geração da População Inicial - A população é gerada aleatoriamente, são gerados 100 cromossomos e este número não se altera no decorrer do algoritmo.

4. Processo de Decomposição do Espaço do *Container*. A decomposição leva em consideração as regras de alocação de torres no *container*.

5. Avaliação do resultado da decomposição do espaço do *container*, através das subfunções objetivo e da Função Objetivo Geral.

6. Iniciação do *loop*, onde a cada *loop* uma geração é observada.

7. Aplicação dos operadores genéticos: seleção (torneio), recombinação (PMX) e mutação (troca de 2 torres de posição), na mesma subrotina acontece a aplicação dos três operadores citados.

8. Novamente acontece a aplicação do processo de decomposição do espaço e volume do *container* - somente no descendente da etapa anterior.

9. Avaliação do resultado da decomposição do espaço do *container* do descendente, através das sub-funções objetivo e da Função Objetivo Geral.

10. Verificar se o cromossomo descendente é aceito ou não na população.

11. Se o critério de parada for atingido é impresso a incumbente com todos os valores avaliados, revelando o valor da função objetivo e a porcentagem do volume ocupado, valor carregado, peso carregado, gravidade, juntamente com o padrão de carregamento, e os vetores auxiliares (caixas referentes a cada torre carregada, tipo de cada caixa, rotação de cada caixa e coordenadas (x, y) das torres carregadas no *container*).

A Figura 36 mostra o fluxograma mais detalhado do Algoritmo Genético Chu-Beasley proposto.

4.4.10 Detalhamento do Processo de Decomposição do espaço do *Container*.

Como já foi mencionado na seção 4.4.7, o processo de decomposição do espaço do *container* respeita as regras de alocação de torres.

No processo de decomposição do espaço do *container* tem-se uma seqüência de passos a seguir e que deve ser respeitada.

1. Corrigir orientação das torres. Toda torre com largura maior que comprimento deve ser rotacionada para que a torre tenha sua maior dimensão no comprimento e menor dimensão na largura.

2. Atualizar índice de rotação das caixas que pertencem as torres rotacionadas.

3. Inicializar o conjunto de vértices de alocação com o vértice $(0, 0)$ e largura residual igual a largura do *container*.

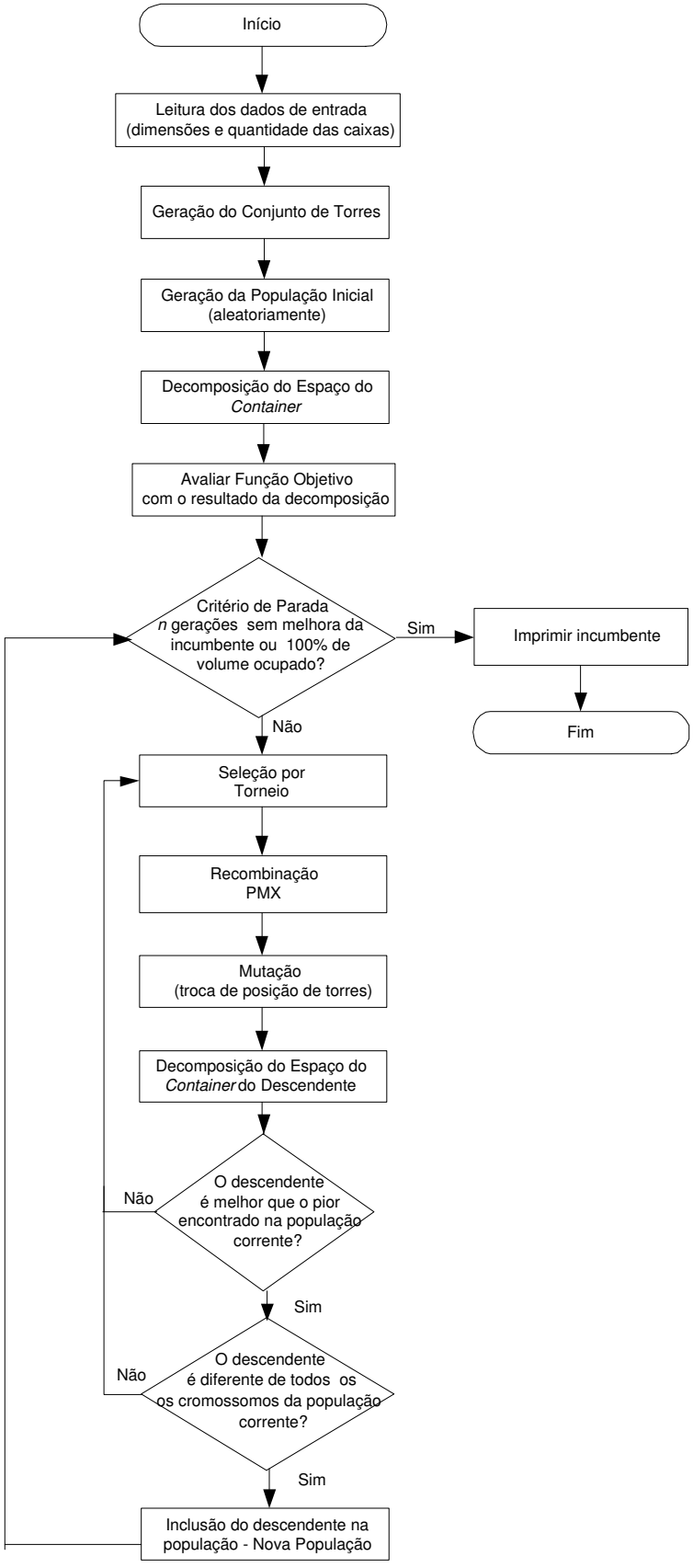

Figura 36: Estrutura detalhada do algoritmo genético Chu-Beasley proposto.

4. Verificar se a torre a ser alocada tem largura igual ou menor que a largura residual do primeiro vértice de alocação.

5. Caso negativo item 4, ir para o próximo vértice do conjunto.

6. Caso positivo, o vértice de alocação utilizado é eliminado e os dois novos vértices (x_1, y_1) e (x_2, y_2) são inseridos no conjunto. Caso nenhum dos vértices disponíveis seja escolhido, passar para a próxima torre usando a seqüência do cromossomo.

7. Calcular a largura residual de cada novo vértice alr_1 e alr_2.

8. Ordenar o conjunto de vértices de alocação por ordem crescente do eixo x (largura do *container*).

9. Verificar se x_1 está dentro do *container* real, ou seja, x_1 deve ser menor que o comprimento do *container*.

10. Caso positivo incluir a torre e suas coordenadas em vetores auxiliares.

11. Reajustar largura residual se houver obstrução.

12. Remanejar o vértice y_1 para a menor coordenada a esquerda em relação ao eixo y quando necessário.

13. Caso o item 12 for realizado, remanejar a torre para obter maior área de contato quando possível.

14. Caso o item 13 for realizado, ordenar o conjunto de vértices de alocação e inserir o vértice excluído no item 6.

15. Verificar necessidade de remanejar para a direita o vértice já remanejado para a esquerda se houver obstrução.

16. Caso positivo o item 15, atualizar largura residual do vértice remanejado.

17. Caso item 15 e 16 positivo, ordenar o conjunto de vértices de alocação por ordem crescente do eixo x.

18. Apagar vértices de alocação, obstruídos pela torre alocada quando houver necessidade.

19. Caso haja obstrução de vértices, ordenar o conjunto de vértices de alocação por ordem crescente do eixo x.

20. Ir para o item 4, repita o processo até alocar a última torre no *container* virtual.

Logo após a decomposição do primeiro cromossomo, um vetor pseudo-cromossomo receberá o conteúdo do vetor solução, onde consta todas as torres que realmente estão dentro do *container* real. Os vetores auxiliares que armazenam as coordenadas de cada torre dentro do *container* também terão seus conteúdos repassados para matrizes que armazenam as coordenadas de cada pseudo-cromossomo. As informações contidas no pseudo-cromossomo serão posteriormente avaliadas pela função objetivo.

O processo de decomposição do espaço do *container* se repete para todos os cromossomos da população inicial e posteriormente este este processo é realizado para o descendente gerado.

A Figura 37 a seguir, traz o fluxograma mais detalhado do Processo de Decomposição do Espaço do *Container*.

4.5 Testes

O algoritmo genético Chu-Beasley proposto neste trabalho foi testado com diferentes dimensões de caixas disponíveis para o carregamento, com diferentes tipos de carga (fortemente homogênea e fortemente heterogênea) e com quantidades diferentes de caixas.

O primeiro teste realizado, utilizou uma carga fortemente homogênea com 100 caixas disponíveis (**Sistema I**). Os dados dimensionais das caixas e do *container* podem ser encontrados no apêndice deste trabalho.

O segundo teste realizado utilizou uma carga fortemente heterogênea com 285 caixas disponíveis para o carregamento (**Sistema II**). Os dados dimensionais das caixas e do *container* também podem ser encontrados no apêndice deste trabalho.

O algoritmo foi desenvolvido em FORTRAN 4.0 e os testes foram realizados em um PC HP BRIO PentiumII Processador IntelMMX/ 256 MB de memória RAM, com sistema operacional Windows 98.

Nos testes realizados, foram utilizados dois tipos de função objetivo. O primeiro cálculo trabalha somente com o volume ocupado pela carga alocada no *container*, veja seção 3.2.2.1, e o segundo cálculo trabalha com uma função objetivo geral proposta por Rodrigues em (RODRIGUES, 2005), veja seção 3.2.2.2.

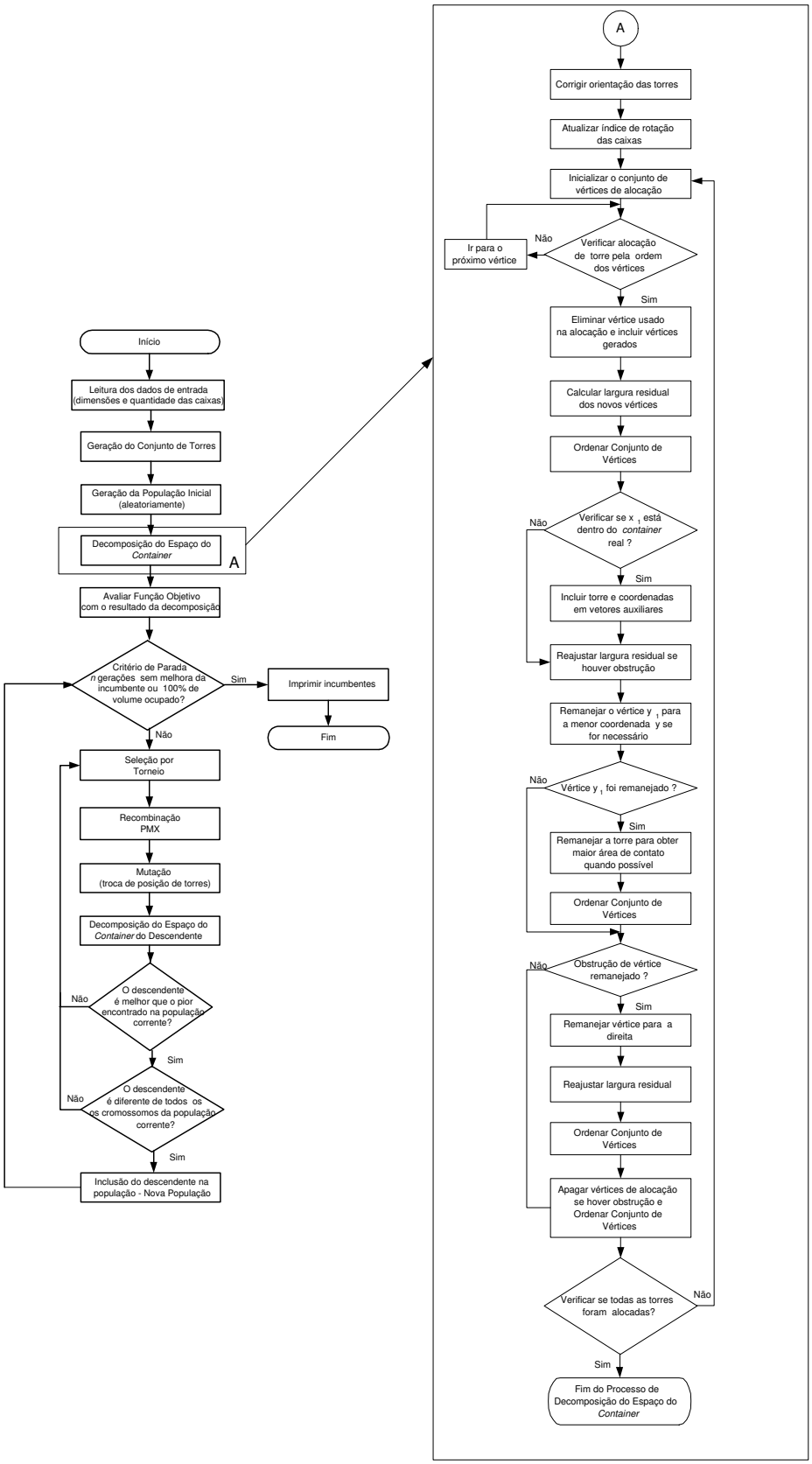

Figura 37: Fluxograma detalhado do Processo de Decomposição do Espaço do *Container*.

4.5.1 Teste com Sistema I

O teste do Sistema I, que trabalha com 100 caixas praticamente iguais (carga homogênea) teve os seguintes resultados:

(a) Considerando somente Volume como Função de Avaliação

O percentual de aproveitamento do volume do *container* pela incumbente foi de aproximadamente 89.18% da carga alocada em seu interior.

(b) Considerando a Função de Avaliação proposta por Rodrigues

Utilizando o cálculo proposto por Rodrigues em (RODRIGUES, 2005), a incumbente alcançou o valor de 74.09% para a Função Geral, 89.18% para o volume ocupado, 21.24% do valor admitido, 4.49% do peso permitido e 134.21% de equilíbrio alcançado.

O programa foi executado em menos de 5 min, e a incumbente (padrão de carregamento), juntamente com os vetores auxiliares estão descritos logo abaixo:

- **Padrão de Carregamento**: 19, 8, 23, 22, 9, 2, 10, 14, 20, 1, 5, 21, 4, 12, 18, 17, 11, 3, 6, 16, 13, 7, 15, 34, 33, 36 e 35.

  O Padrão de carregamento são as torres que preenchem o *container*.

- **Caixas que compõem cada Torre**: torre 19 (caixas:36, 35, 34), torre 8 (caixas:69, 68, 67), torre 23 (caixas:24, 23, 22), torre 22 (caixas:27, 26, 25), torre 9 (caixas:66, 65, 64), torre 2 (caixas:87, 86, 85), torre 10 (caixas:63, 62, 61), torre 14 (caixas:51, 50, 49), torre 20 (caixas:33, 32, 31), torre 1 (caixas:90, 89, 88), torre 5 (caixas:78, 77, 76), torre 21 (caixas:30, 29, 28), torre 4 (caixas:81, 80, 79), torre 12 (caixas:57, 56, 55), torre 18 (caixas:39, 38, 37), torre 17 (caixas:42, 41, 40), torre 11 (caixas:60, 59, 58), torre 3 (caixas:84, 83, 82), torre 6 (caixas:75, 74, 73), torre 16 (caixas:45, 44, 43), torre 13 (caixas:54, 53, 52), torre 7 (caixas:72, 71, 70), torre 15 (caixas:48, 47, 46), torre 34 (caixas:4, 3), torre 33 (caixas:6, 5), torre 36 (caixas:100, 99, 98) e torre 35 (caixas:2, 1).

- **Tipos de Caixas de cada Torre**: torre 19 (tipo:2, 2, 2), torre 8 (tipo:4, 4, 4), torre 23 (tipo:2, 2, 2), torre 22 (tipo:2, 2, 2), torre 9 (tipo:4, 3, 3), torre 2 (tipo:5, 5, 5), torre 10 (tipo:3, 3, 3), torre 14 (tipo:3, 3, 3), torre 20 (tipo:2, 2, 2), torre 1 (tipo:5, 5, 5), torre 5 (tipo:4, 4, 4), torre 21 (tipo:2, 2, 2), torre 4 (tipo:5, 4, 4), torre 12

(tipo:3, 3, 3), torre 18 (tipo:2, 2, 2), torre 17 (tipo:2, 2, 2), torre 11 (tipo:3, 3, 3), torre 3 (tipo:5, 5, 5), torre 6 (tipo:4, 4, 4), torre 16 (tipo:2, 2, 2), torre 13 (tipo:3, 3, 3), torre 7 (tipo:4, 4, 4), torre 15 (tipo:3, 3, 3), torre 34 (tipo:1, 1), torre 33 (tipo:1, 1), torre 36 (tipo:7, 7, 7) e torre 35 (tipo:1, 1).

- **Índice de Rotação de cada Caixa**: torre 19 (tipo:1, 1, 1), torre 8 (tipo:1, 1, 1), torre 23 (tipo:1, 1, 1), torre 22 (tipo:1, 1, 1), torre 9 (tipo:1, 1, 1), torre 2 (tipo:1, 1, 1), torre 10 (tipo:1, 1, 1), torre 14 (tipo:1, 1, 1), torre 20 (tipo:1, 1, 1), torre 1 (tipo:1, 1, 1), torre 5 (tipo:1, 1, 1), torre 21 (tipo:1, 1, 1), torre 4 (tipo:1, 1, 1), torre 12 (tipo:1, 1, 1), torre 18 (tipo:1, 1, 1), torre 17 (tipo:1, 1, 1), torre 11 (tipo:1, 1, 1), torre 3 (tipo:1, 1, 1), torre 6 (tipo:1, 1), torre 16 (tipo:1, 1, 1), torre 13 (tipo:1, 1, 1), torre 7 (tipo:1, 1, 1), torre 15 (tipo:1, 1, 1), torre 34 (tipo:1, 1), torre 33 (tipo:1, 1), torre 36 (tipo:1, 1, 1) e torre 35 (tipo:1, 1).

- **Coordenadas y(largura) e x(comprimento) de cada torre**: torre 19 v1(0, 56.3) v2(35.8, 0), torre 8 v1(35.8, 56.3) v2(71.6, 0), torre 23 v1(71.6, 56.3) v2(107.4, 0), torre 22 v1(0, 112.6) v2(35.8, 56.3), torre 9 v1(35.8, 112.6) v2(71.6, 56.3), torre 2 v1(71.6, 112.6) v2(107.4, 56.3), torre 10 v1(0, 168.9) v2(35.8, 112.6), torre 14 v1(35.8, 168.9) v2(71.6, 112.6), torre 20 v1(71.6, 168.9) v2(107.4, 112.6), torre 1 v1(0, 225.2) v2(35.8, 168.9), torre 5 v1(35.8, 225.2) v2(71.6, 168.9), torre 21 v1(71.6, 225.2) v2(107.4, 168.9), torre 4 v1(0, 281.5) v2(35.8, 225.2), torre 12 v1(35.8, 281.5) v2(71.6, 225.2), torre 18 v1(71.6, 281.5) v2(107.4, 225.2), torre 17 v1(0, 337.8) v2(35.8, 281.5), torre 11 v1(35.8, 337.8) v2(71.6, 281.5), torre 3 v1(71.6, 337.8) v2(107.4, 281.5), torre 6 v1(0, 394.1) v2(35.8, 337.8), torre 16 v1(35.8, 394.1) v2(71.6, 337.8), torre 13 v1(71.6, 394.1) v2(107.4, 337.8), torre 7 v1(0, 450.4) v2(35.8, 394.1), torre 15 v1(35.8, 450.4) v2(71.6, 394.1), torre 34 v1(71.6, 440.1) v2(107.6, 394.1), torre 33 v1(0, 496.4) v2(36, 450.4), torre 36 v1(36, 496.4) v2(72, 450.4) e torre 35 v1(72, 486.1) v2(108, 440.1).

4.5.2 Teste com Sistema II

O teste do Sistema II, que trabalha com 285 caixas, representando uma carga do tipo fortemente heterogênea, teve os seguintes resultados:

(a)Considerando somente Volume como Função de Avaliação

Neste segundo teste, foi utilizado para a avaliação do padrão de carregamento somente

o volume da carga alocada no interior do *container* e o percentual de aproveitamento do volume do *container* foi de 89.44%.

(b)Considerando a Função de Avaliação proposta por Rodrigues

Utilizando a proposta de Rodrigues em (RODRIGUES, 2005) para a avaliação do padrão de carregamento, alcançou-se o valor de 73.39% para a Função Geral, 89.44% para o volume ocupado, 17.2% do valor admitido, 5.52% do peso permitido e 124.84% de equilíbrio alcançado. O melhor resultado entre as simulações.

O programa foi executado em menos de 5 min, e a incumbente (padrão de carregamento), juntamente com os vetores auxiliares estão descritos logo abaixo:

- **Padrão de Carregamento**: 75, 76, 41, 61, 6, 51, 19, 49, 77, 13, 22, 48, 81, 84, 78 e 54.

  O Padrão de carregamento são as torres que preenchem o *container*.

- **Caixas que compõem cada Torre**: torre 75 (caixas:192, 191, 190), torre 76 (caixas:189, 188, 187), torre 41 (caixas:99, 98, 83), torre 61 (caixas:234, 233, 232), torre 6 (caixas:169, 168, 6), torre 51 (caixas:264, 263, 262), torre 19 (caixas:143, 142, 61), torre 49 (caixas:270, 269, 268), torre 77 (caixas:186, 185, 184), torre 13 (caixas:155, 154, 13), torre 22 (caixas:137, 136, 64), torre 48 (caixas:273, 272, 271), torre 81 (caixas:41, 40, 39, 38, 37, 36, 35), torre 84 (caixas:20, 19, 18, 17, 16, 15, 14), torre 78 (caixas:183, 182, 181) e torre 54 (caixas:255, 254, 253).

- **Tipos de Caixas de cada Torre**: torre 75 (tipo:5, 5, 5), torre 76 (tipo:5, 5, 5), torre 41 (tipo:3, 3, 2), torre 61 (tipo:6, 6, 6), torre 6 (tipo:4, 4, 1), torre 51 (tipo:6, 6, 6), torre 19 (tipo:3, 3, 2), torre 49 (tipo:7, 7, 7), torre 77 (tipo:5, 5, 5), torre 13 (tipo:4, 4, 1), torre 22 (tipo:3, 3, 2), torre 48 (tipo:7, 7, 7), torre 81 (tipo:1, 1, 1, 1, 1, 1, 1), torre 84 (tipo:1, 1, 1, 1, 1, 1, 1), torre 78 (tipo:5, 5, 5) e torre 54 (tipo:6, 6, 6).

- **Índice de Rotação de cada Caixa**: torre 75 (tipo:1, 1, 1), torre 76 (tipo:1, 1, 1), torre 41 (tipo:5, 5, 1), torre 61 (tipo:1, 1, 1), torre 6 (tipo:5, 5, 1), torre 51 (tipo:1, 1, 1), torre 19 (tipo:5, 5, 1), torre 49 (tipo:1, 1, 1), torre 77 (tipo:1, 1, 1), torre 13 (tipo:5, 5, 1), torre 22 (tipo:5, 5, 1), torre 48 (tipo:1, 1, 1), torre 81 (tipo:1, 1, 1, 1, 1, 1, 1), torre 84 (tipo:1, 1, 1, 1, 1, 1, 1), torre 78 (tipo:1, 1, 1) e torre 54 (tipo:1, 1, 1).

- **Coordenadas y(largura) e x(comprimento) de cada torre**: torre 75 v1(0, 56.3) v2(35.8, 0), torre 76 v1(35.8, 56.3) v2(71.6, 0) , torre 41 v1(0, 143.2) v2(69.6, 56.3), torre 61 v1(71.6, 56.3) v2(107.4, 0), torre 6 v1(0, 233.0) v2(69.6, 143.2), torre 51 v1(69.6, 112.6) v2(105.4, 56.3), torre 19 v1(0, 319.9) v2(69.6, 233), torre 49 v1(69.6, 168.9) v2(105.4, 112.6), torre 77 v1(69.6, 225.2) v2(105.4, 168.9), torre 13 v1(0, 409.7) v2(69.6, 319.9), torre 22 v1(0, 496.6) v2(69.6, 409.7), torre 48 v1(69.6, 281.5) v2(105.4, 225.2), torre 81 v1(69.6, 331.5) v2(107.8, 281.5), torre 84 v1(69.6, 381.5) v2(107.8, 331.5), torre 78 v1(69.6, 437.8) v2(105.4, 381.5) e torre 54 v1(69.6, 494.1) v2(105.4,437.8).

4.6 Análise de Resultados

Como pode ser observado, o algoritmo genético Chu-Beasley proposto neste trabalho demonstrou ser uma técnica eficiente para o problema de carregamento de *container*.

Os resultados obtidos são altamente satisfatórios, tendo em vista que na literatura encontram-se resultados tão satisfatórios como os aqui apresentados.

Comparando os resultados deste trabalho com os resultados de Rodrigues em (RODRIGUES, 2005), fica claro o melhor desempenho do algoritmo aqui proposto com relação a carga fortemente heterogênea, e mesmo comparando os resultados do teste com carga fortemente homogênea com os resultados obtidos por Rodrigues, a diferença é mínima.

Em Rodrigues (RODRIGUES, 2005), nos testes realizados com as mesmas 285 caixas testadas neste trabalho, ele obteve um percentual médio de volume ocupado no *container* de 67.57%, com função geral no valor de 66.58%. Onde a incumbente segundo Rodrigues em (RODRIGUES, 2005) tem 70.19% de volume ocupado no *container* e função geral 69.0%.

Como foi possível observar, os resultados obtidos neste trabalho são superiores aos resultados de Rodrigues em (RODRIGUES, 2005) quando se trata de carga fortemente heterogênea e com uma diferença mínima quando se trata de carga fortemente homogênea.

O algoritmo genético Chu-Beasley proposto neste trabalho, se mostrou flexível ao se adaptar ao problema específico estudado, trabalhando de maneira aceitável com grupos de caixas fortemente homogêneas, mas o desempenho obtido com a aplicação do algoritmo proposto neste trabalho com caixas fortemente heterogêneas foi satisfatório e de boa qualidade.

5 Conclusões

Como observado a heurística proposta demonstrou ser uma técnica eficiente para o problema de carregamento de *container*, ocupando o espaço disponível quase em seu total.

Os resultados obtidos são altamente satisfatórios, tendo em vista que na literatura encontram-se resultados inferiores aos aqui apresentados.

A heurística proposta se mostrou flexível ao se adaptar ao problema específico estudado, trabalhando de maneira aceitável tanto com grupos de caixas fortemente homogêneas quanto fortemente heterogêneas.

Conforme o objetivo proposto para o referente trabalho, pode-se confirmar quais técnicas heurísticas são eficientes para solucionar problemas de carregamento, especialmente, quando se tem um problema de carregamento complexo e com múltiplas restrições. Técnicas clássicas de otimização e modelos matemáticos, que se tornam bastante complexos para resolver este tipo de problema, não conseguem ser utilizados para obtenção de resultados satisfatórios em tempo razoável.

Comparando os testes realizados com carga fortemente homogênea e fortemente heterogênea observou-se que a heurística proposta obtém resultados melhores e mais satisfatórios quando trabalha com carga fortemente heterogênea, pois na própria teoria da heurística, as caixas de mesmas dimensões estão alocadas juntas no interior do *container*. A carga fortemente homogênea possibilita preencher o Corpo Principal do *container* com mais facilidade, já os Espaços Residuais: Lateral, Superior e Frontal são mais difíceis de serem preenchidos devido o fato da carga ser fortemente homogênea e não ter variações de dimensões.

Mas como nos testes foi utilizada uma carga fortemente homogênea, com algumas caixas somente diferenciadas das demais, foi possível obter um resultado satisfatório. Os resultados encontrados na literatura com carga fortemente homogênea são considerados de má qualidade, mas os resultados encontrados aqui são aceitáveis.

Quanto ao teste realizado com carga fortemente heterogênea, observou-se que os resultados apresentados pela heurística são de boa qualidade. Pois se tem grande quantidade de caixas similares para preencher o Corpo Principal do *Container* e além disso traz uma variedade de outros tipos de caixas com dimensões diferentes que são perfeitamente aplicáveis aos Espaços Residuais: Lateral, Superior e Frontal.

Percebe-se, portanto, que os resultados obtidos com a aplicação da heurística proposta pelo trabalho é influenciado pelas dimensões e pela quantidade de caixas disponíveis para a formação do padrão de carregamento, bem como pelas dimensões do *container*. O que leva a concluir-se que dependendo das características e da quantidade de caixas, a heurística pode ser mais adequada para o carregamento do *container* ou não ser indicada.

Sobre os resultados obtidos com a aplicação do algoritmo genético de Chu-Beasley observou-se que mais uma vez a carga fortemente heterogênea proporciona resultados melhores que os já apresentados por Rodrigues em (RODRIGUES, 2005). Observou-se também que os resultados com carga fortemente homogênea têm uma diferença mínima em relação aos resultados apresentados por Rodrigues.

Portanto para estudos futuros propõe-se que outros processos de decomposição do espaço do *container* sejam implementados para fins de testes e comparações. Um dos processos de decomposição do espaço do *container* indicado é a decomposição por camadas também proposta por Gehring e Bortfeldt em (BORTFELDT; GEHRING, 2001). Outra proposta para estudos futuros está relacionada com a formação de torres, propõe-se que outras formas de geração de torres sejam estudadas e testadas para fins de comparações.

As diferenças na obtenção e preenchimento dos subespaços, e as diferenças de restrições na seleção das caixas são os fatores que diferenciam significativamente os dois métodos de preenchimento, e aliados às características das caixas influenciam no resultado final do padrão de carregamento obtido.

O relato acima demonstra que a utilização das duas propostas, tanto a heurística como o Algoritmo Genético, como objeto de comparação para obtenção da melhor configuração de cargas, podem ser adotados afim de se obter o padrão de carregamento mais adequado.

Como se observa, heurísticas e metaheurísticas, como os algoritmos genéticos aqui aplicados, são técnicas eficientes e que se adaptam adequadamente a problemas complexos de carregamento, com diversas restrições e diferentes configurações de caixas.

Por fim, comprovou-se que o problema de carregamento de *container* por ser de difícil resolução matemática, tem como técnicas de solução aproximada, heurísticas e

metaheurísticas, sendo uma delas a heurística construtiva aqui apresentada e o algoritmo genético especializado também aqui proposto.

Referências

ALENCAR, M. P. L. *Análise Crítica do Algoritmo Genético de Chu-Beasley para o Problema Generalizado de Atribuição*. [S.l.]: XXXVI SBPO - Simpósio Brasileiro de Pesquisa Operacional organizado pelo SOBRAPO, 2004. 1391–1399 p.

ARMBRUSTER, M. A solution procedure for a pattern sequencing problem as part of a one-dimensional cutting stock problem in the steel industry. *European Journal of Operational Research*, v. 141, p. 328–340, 2002.

BISCHOFF, E.; DOWSLAND, W. B. An application of the micro to product design and distribution. *Operational Research Society Journal*, v. 33, p. 271–280, 1982.

BISCHOFF, E. E.; MARRIOTT, M. D. A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, v. 44, p. 267–276, 1990.

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

BORTFELDT, A.; GEHRING, H. A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, v. 131, p. 143–161, 2001.

CARVALHO, J. M. V. Lp models for bin packing and cutting stock problems. *European Journal of Operational Research*, v. 141, p. 253–273, 2002.

CHEN, C. S.; LEE, S. M.; SHEN, Q. S. An analytical model for the container loading problem. *European Journal of Operational Research*, v. 80, p. 68–76, 1995.

CHU, P.; BEASLEY, J. E. A genetic algorithm for the generalized assignment problem. *Computers and Operation Research*, v. 24, n. 1, p. 17–23, 1997.

DAVIES, A. P.; BISCHOFF, E. E. Weight distribution considerations in container loading. *European Journal of Operational Research*, v. 114, p. 509–527, 1999.

DOWSLAND, K. A.; DOWSLAND, W. B. Packing problems. *European Journal of Operational Research*, v. 56, p. 2–14, 1992.

DYCKHOFF, H. A typology of cutting and packing problems. *European Journal of Operational Research*, v. 44, p. 145–159, 1990.

DÍAZ, A. et al. *Optimización Heurística y Redes Neuronales*. [S.l.]: Paraninfo, 1996.

ELEY, M. Solving container loading problems by block arrangement. *European Journal of Operational Research*, v. 141, p. 393–409, 2002.

FARLEY, A. A. The cutting stock problem in the canvas industry. *European Journal of Operational Research*, v. 44, p. 247–255, 1990.

FRASER, H. J.; GEORGE, J. A. Integrated container loading software for pulp and paper industry. *European Journal of Operational Research*, v. 77, p. 466–474, 1994.

GEHRING, H.; BORTFELDT, A. A genetic algorithm for solving the container loading problem. *International Transactions of Operational Research*, v. 4, n. 5/6, p. 401–418, 1997.

GEHRING, H.; MENSCHNER, K.; MEYER, M. A computer-based heuristic for packing pooled shipment containers. *European Journal of Operational Research*, v. 44, p. 277–288, 1990.

GEORGE, J. A.; ROBINSON, D. F. A heuristic for packing boxes into a container. *Computers and Operational Research*, v. 7, p. 147–156, 1980.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problem. *Operations Research*, v. 9, p. 849–859, 1961.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problem - part i. *Operations Research*, v. 11, p. 863–888, 1963.

GILMORE, P. C.; GOMORY, R. E. Multistage cutting problems of two and more dimensions. *Operations Research*, v. 13, p. 94–119, 1965.

GOMES, A. M.; OLIVEIRA, J. F. A 2-exchange heuristic for nesting problems. *European Journal of Operational Research*, v. 141, p. 359–370, 2002.

GOULIMIS, C. Optimal solutions for the cutting stock problem. *European Journal of Operational Research*, v. 44, p. 197–208, 1990.

HADJICONSTANTINOU, E.; CHRISTOFIDES, N. An exact algorithm for orthogonal 2-d cutting problems using guillotine cuts. *European Journal of Operational Research*, v. 83, p. 21–38, 1995.

HAESSLER, R. W.; SWEENEY, P. E. Cutting stock problems and solution procedures. *European Journal of Operational Research*, v. 54, p. 141–150, 1991.

HOLTHAUS, O. Decomposition approaches for solving the integer one-dimensional cutting stock problem with different types of standard lengths. *European Journal of Operational Research*, v. 141, p. 295–312, 2002.

LAGUNA, M. A guide to implementing tabu search. *Investigación Operativa*, v. 4, n. 1, p. 143–161, April 2001.

LINS, L.; LINS, S.; MORABITO, R. An n-tet graph approach for non-guillotine packings of n-dimensional boxes into an n-container. *European Journal of Operational Research*, v. 141, p. 421–439, 2002.

LODI, A.; MARTELLO, S.; MONACI, M. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, v. 141, p. 241–252, 2002.

LODI, A.; MARTELLO, S.; VIGO, D. Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research*, v. 141, p. 410–420, 2002.

MOHANTY, B. B.; MATHUR, K.; IVANCIC, N. J. Value considerations in three-dimensional packing - a heuristic procedure using the fractional knapsack problem. *European Journal of Operational Research*, v. 74, p. 143–151, 1994.

MORABITO, R.; ARENALES, M. An and-or-graph approach for two-dimensional cutting problems. *European Journal of Operational Research*, v. 58, p. 263–271, 1992.

PISINGER, D. Heuristics for the container loading problem. *European Journal of Operational Research*, v. 141, p. 382–392, 2002.

RODRIGUES, L. L. *Um Algoritmo Genético para o Problema de Carregamento de Container*. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2005.

ROMERO, R.; MANTOVANI, J. R. S. Introdução a metaheurísticas. *Anais do 3º Congresso Temático de Dinâmica e Controle da SBMAC, organizado pela UNESP*, 2004.

SOARES, G. L. *Algoritmos Genéticos: Estudos, Novas Técnicas e Aplicações*. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal de Minas Gerais, Belo Horizonte, 1997.

WU, Y. et al. An effective quasi-human based heuristic for solving the rectangle packing problem. *European Journal of Operational Research*, v. 141, p. 341–358, 2002.

APÊNDICE A – Dados dos Sistemas Testados

Sistema I

O Sistema I é formado por 100 caixas praticamente do mesmo tipo, ou seja, a carga utilizada no Sistema I é fortemente homogênea.

Os dados utilizados em nossos testes foram apresentados em Rodrigues (RODRIGUES, 2005), o *container* tem proporções de 1.35m de altura, 5m de comprimento e 1.08m de largura. Tem como peso máximo de carregamento 10.000 Kg e o valor de R$ 150000.00.

Os valores dos pesos k_1, k_2, k_3 e k_4 que foram utilizados no cálculo de uma das propostas de avaliação do padrão de carregamento são respectivamente 7, 0.5, 0.5 e 2.

Logo abaixo temos a Tabela 4 que revela os dados dimensionais, peso, valor e quantidade de caixas disponíveis para o teste com o Sistema I.

Tabela 4: Dados dimensionais, pesos e valores das caixas do teste com 100 caixas.

Tipo	Altura (cm)	Comprimento (cm)	Largura (cm)	Volume (cm^3)	Peso (kg)	Valor (R$)	Quantidade
A	47.7	46.0	36.0	78991.2	14.1	1253.00	20
B	41.9	56.3	35.8	84451.1	9.6	626.00	25
C	41.9	56.3	35.8	84451.1	10	704.00	20
D	41.9	56.3	35.8	84451.1	10.6	791.00	15
E	41.9	56.3	35.8	84451.1	10.6	1018.00	10
F	45.6	47.3	38.2	82392.8	13.7	1488.00	5
G	44.5	44.1	33.5	65742.1	10.7	1096.00	5

Sistema II

Os dados citados abaixo foram utilizados pelo Sistema II. Este Sistema é formado por 285 caixas de tipos diferentes. Esta carga é considerada fortemente heterogênea, pois têm-se poucos tipos de caixas, com muitas caixas de cada tipo.

Os dados utilizados neste teste também foram apresentados em Rodrigues (RODRIGUES, 2005), o *container* tem proporções iguais as do Sistema I. São elas: 1.35m de altura, 5m de comprimento e 1.08m de largura.

O peso máximo de carregamento do *container* e o valor máximo da carga carregada mudaram, neste teste foi considerado o peso máximo de carregamento de 18070 kg e o valor máximo da carga no *container* de 300000 reais.

Os valores dos pesos k_1, k_2, k_3 e k_4 que foram utilizados no cálculo de uma das propostas de avaliação do padrão de carregamento são respectivamente 7, 0.5, 0.5 e 2.

Logo abaixo temos a Tabela 5 que revela os dados dimensionais, peso, valor e quantidade de caixas disponíveis para o teste com o Sistema II.

Tabela 5: Dados dimensionais, pesos e valores das caixas do teste com 285 caixas.

Tipo	Altura (cm)	Comprimento (cm)	Largura (cm)	Volume (cm^3)	Peso (kg)	Valor (R$)	Quantidade
A	18.3	50.0	38.2	34953.00	4.2	548	55
B	15.6	50.0	38.2	29796.00	3.9	508	30
C	69.6	59.3	86.9	358660.63	53.0	1410	68
D	69.6	57.6	89.8	360004.61	82.0	3135	26
E	41.9	56.3	35.8	84451.126	9.6	626	45
F	41.9	56.3	35.8	84451.126	10.6	791	43
G	41.9	56.3	35.8	84451.126	10.6	791	18

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo