UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE SISTEMAS

PROBLEMAS DE CORTE E EMPACOTAMENTO TRIDIMENSIONAL E INTEGRAÇÃO COM ROTEAMENTO DE VEÍCULOS

Autor: Olinto César Bassi de Araújo

Orientador: Vinícius Amaral Armentano

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas – UNICAMP, como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica.

Banca Examinadora:

Vinícius Amaral Armentano	(orientador)
Reinaldo Morabito	DEP-UFSCar
Marcos Nereu Arenales	ICMC-USP
Flávio Keidi Miyazawa	DTC-IC-UNICAMP
Takaaki Ohishi	DENSIS-FEEC-UNICAMP
Secundino Soares Filho	DENSIS-FEEC-UNICAMP

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Ar15p	Araújo, Olinto César Bassi de Problemas de corte e empacotamento tridimensional e integração com roteamento de veículos / Olinto César Bassi de Araújo. --Campinas, SP: [s.n.], 2006. Orientador: Vinícius Amaral Armentano Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação. 1. Problema do corte de estoque. 2. Programação heurística. 3. Otimização combinatória. 4. Pesquisa operacional. I. Armentano, Vinícius Amaral. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Three-dimensional cutting and packing problems and integration with vehicle routing
Palavras-chave em Inglês: Three-dimensional cutting and packing problem, Vehicle routing, Metaheuristics, Adaptive memory
Área de concentração: Automação
Titulação: Doutor em Engenharia Elétrica
Banca examinadora: Flávio Keide Miyazawa, Marcos Nereu Arenales, Reinaldo Morabito e Takaaki Ohishi
Data da defesa: 15/12/2006
Programa de Pós-Graduação: Engenharia Elétrica

RESUMO

A adoção de contêineres em grande escala tornou possível o desenvolvimento do transporte multimodal. Atualmente, carregamento de caixas em contêineres é uma importante atividade em empresas que têm no transporte de carga um fator logístico de alto custo. Este trabalho apresenta o desenvolvimento e aplicação de metaheurísticas com memória adaptativa para a resolução de problemas de corte e empacotamento tridimensional, bem como a integração destes com o problema de roteamento de veículos. Mais especificamente, são tratados os problemas de carregamento de contêiner, *bin packing* tridimensional e roteamento de veículos capacitados com restrições de empacotamento tridimensional. Uma nova abordagem, baseada em cubóides de tamanho variável, é utilizada para calcular os padrões de carregamento tridimensional em todos os métodos propostos. Restrições de orientação, estabilidade, centro de gravidade, projeção da base de apoio e múltiplos destinos são consideradas. Extensivos testes computacionais são realizados para demonstrar o desempenho das abordagens propostas.

Palavras-chave: problemas de corte e empacotamento tridimensional, padrões de corte tridimensionais, roteamento de veículos, metaheurísticas, memória adaptativa

ABSTRACT

The wide-scale adoption of the containers made the development of the multimodal transport possible. Nowadays, shipment of boxes in containers is an important activity for companies that have in the load transport a logistic factor of high cost. This work presents the development and the application of metaheuristics with adaptive memory in order to solve three-dimensional cutting and packing problems, as well as their integration with the vehicle routing problem. In particular, problems of container loading, three-dimensional bin packing and vehicle routing with three-dimensional packing constraints are considered. Furthermore, a new approach based on maximal cuboids that fit in given empty spaces is used to calculate the packing patterns in the proposed methods. Constrains on orientation, stability, center of gravity, overhang and multiple destination are considered. Extensive computational experiments are carried out to demonstrate the performance of the proposed approaches.

Keywords: three-dimensional cutting and packing problem, three-dimensional packing pattern, vehicle routing, metaheuristics, adaptive memory.

À minha esposa Mauren e ao meu filho Guilherme.

AGRADECIMENTOS

Ao meu orientador, Prof. Vinícius Amaral Armentano, por ser um exemplo de dedicação à pesquisa e pela sua eterna preocupação com a formação integral dos orientandos. Sua paciência, tolerância e talento incomum para encontrar a melhor direção a seguir na pesquisa só foram superados pela amizade e confiança que sempre depositou na minha pessoa. Registro ainda o esmero que demonstrou na leitura e correção do texto desta tese.

Aos colegas de trabalho professores Erni José Milani e Canrobert Kumpfer Werlang que incentivaram e intercederam na minha liberação para o doutorado.

Ao Prof. Paulo Morelato França por ter me recebido na UNICAMP e ao Prof. Felipe Martins Muller pela confiança e incentivo.

Ao Prof. Reinaldo Morabito por ter sugerido o estudo do problema de carregamento e roteamento de veículos.

Aos colegas e amigos André, Gabriela, Gerardo e Mariela pela inestimável ajuda e colaboração.

A todos os companheiros do DENSIS que promoveram um ambiente acolhedor e propício para colaboração mútua, em especial ao Vinicius Jacques, Aníbal , Ricardo, Sandra e Sachi.

Ao grande amigo Haroldo pelas gratificantes discussões que invariavelmente versavam sobre trabalho, a vida e a obra de Arthur C. Clarke, nesta ordem.

Aos colegas de república Marcelo e Maurício pela convivência que tornou o primeiro ano em Campinas mais agradável e familiar.

À amiga Ana Armentano pela sua alegria e sinceridade.

À Mauren e ao Guilherme, pelo constante incentivo e compreensão nos longos momentos que eu estive ausente.

ÍNDICE

1 **Introdução** 1

1.1 Objetivo e estrutura do trabalho..... 3

2 **Definição do problema e revisão bibliográfica** 5

2.1 Introdução..... 5

2.2 Problemas abordados..... 10

2.2.1 Conjunto de restrições..... 12

2.3 Métodos de resolução para PCE3D..... 13

2.3.1 Métodos exatos e algoritmos de aproximação..... 13

2.3.2 Heurísticas..... 15

2.3.3 Busca em árvore e metaheurísticas..... 27

3 **Problema de carregamento de contêiner** 53

3.1 Introdução..... 53

3.2 Algoritmo de múltiplos inícios para o problema de carregamento de contêiner..... 61

3.2.1 Representação da solução e identificação dos espaços vazios..... 62

3.2.2 Cálculo e avaliação dos cubóides..... 67

3.2.3 Estruturas de memória..... 71

3.2.4 Grau de reconstrução..... 74

3.2.5 Algoritmo proposto..... 76

3.2.6 Adaptações do algoritmo proposto para diferentes conjuntos de restrições..... 80

3.3 Resultados computacionais..... 84

3.3.1 Restrições de estabilidade..... 84

3.3.2 Restrições de empilhamento..... 93

3.3.3 Restrições de distribuição de peso..... 95

4 **Problema *Bin Packing* 3D** 99

4.1 Introdução…..... 99

4.2 Algoritmo de descida em vizinhança variável para o problema *bin packing* 3D..... 101
4.3 Resultados Computacionais..... 105
4.3.1 Resultados computacionais para o conjunto de instâncias de Ivancic et al. (1989)..... 106
4.3.2 Resultados computacionais para os conjuntos de instâncias de Lim e Zang (2005)..... 110
4.3.3 Resultados computacionais para os conjuntos de instâncias de Gendreau et al. (2006a)..... 112

5 **Problema de carregamento e roteamento de veículos** 115
5.1 Introdução..... 115
5.2 Algoritmo de busca tabu para o problema de roteamento de veículos capacitados 3D. 116
5.3 Resultados computacionais..... 124

6 **Conclusões e trabalhos futuros** 133
6.1 Conclusões..... 133
6.2 Trabalhos futuros..... 135

Referências Bibliográficas 137

Apêndice 147

LISTA DE TABELAS

2.1 Exigências práticas para PCE3D conforme Bischoff e Ratcliff (1995a)..... 7

3.1 Conjunto de caixas do exemplo 1..... 59

3.2 Conjunto de caixas do exemplo 2..... 63

3.3 Orientações factíveis do conjunto de caixas do exemplo 2..... 63

3.4 Conjunto de soluções elite..... 73

3.5 Probabilidades de escolha das reduções do cubóide c_{131}..... 74

3.6 Descrição dos parâmetros do algoritmo de múltiplos inícios para PCE3D 80

3.7 Conjunto de critérios de avaliação..... 85

3.8 Resultados para instâncias de Bischoff e Ratcliff (1995a)..... 87

3.9 Resultados estabilidade do carregamento, instâncias de Bischoff e Ratcliff(1995a)..... 89

3.10 Resultados para as instâncias de Chien e Deng (2004)..... 89

3.11 Valores dos parâmetros para os algoritmos AA1 e AA2..... 90

3.12 Resultados adicionais para instâncias de Bischoff e Ratcliff (1995a)..... 92

3.13 Resultados para instâncias de Bischoff e Ratcliff (1995a) com restrições de projeção da base de apoio..... 93

3.14 Conjunto de critérios de avaliação dos cubóides - restrições de empilhamento 93

3.15 Resultados para instâncias de Bischoff e Ratcliff (1995a) com restrições de empilhamento 94

3.16 Tempo computacional para instâncias de Bischoff e Ratcliff (1995a) com restrições de empilhamento..... 94

3.17 Resultados para instâncias de Bischoff e Ratcliff (1995a) com restrições de

empilhamento reduzidas à metade..... 95

3.18 Resultados para instâncias de Bischoff e Ratcliff (1995a) com restrições de empilhamento reduzidas à quarta parte..... 95

3.19 Resultados para instâncias de Bischoff e Ratcliff (1995a) com restrições de distribuição de peso..... 97

4.1 Descrição dos parâmetros do algoritmo de descida em vizinhança variável.... 104

4.2 Valores dos parâmetros do algoritmo para carregamento de contêiner..... 105

4.3 Valores dos parâmetros do algoritmo para bin packing 3D..... 106

4.4 Resultados para o conjunto de instâncias de Ivancic et al. (1989)..... 109

4.5 Resultados para os conjuntos de instâncias de Lim e Zhang (2005)..... 111

4.6 Tempo computacional do algoritmo AA2pb para os conjuntos de instâncias de Lim e Zhang (2005)..... 112

4.7 Resultados para os conjuntos de instâncias de Gendreau et al. (2006a)..... 114

5.1 Descrição dos parâmetros do algoritmo busca tabu para o problema de roteamento de veículo capacitados 3D..... 124

5.2 Valores dos parâmetros do algoritmo AAcr..... 125

5.3 Valores dos parâmetros do algoritmo para carregamento tridimensional..... 126

5.4 Conjunto de critérios de avaliação..... 126

5.5 Resultados do algoritmo AAcr para as instâncias de Gendreau et al. (2006a) 127

5.6 Resultados comparativos para as instâncias de Gendreau et al. (2006a) 128

5.7 Resultados do algoritmo AAcr com diferentes configurações de restrições..... 130

5.8 Resultados comparativos com diferentes configurações de restrições..... 131

LISTA DE FIGURAS

2.1 Classificação dos problemas de corte e empacotamento segundo suas propriedades geométricas..... 6

2.2 Representação das seis possíveis orientações de uma caixa..... 8

2.3 Restrições de estabilidade..... 8

2.4 Sistema de coordenadas para localizar caixas dentro de um contêiner..... 11

2.5 Espaços criados devido ao carregamento de uma caixa..... 12

2.6 Representação da primeira camada vertical de um contêiner..... 16

2.7 Exemplo de padrão de carregamento do tipo pilha..... 17

2.8 Padrão de corte guilhotinado (a) e não guilhotinado (b)..... 18

2.9 Cubóide (2, 2, 3) formado por caixas do tipo t com orientação k..... 18

2.10 Seqüência do carregamento com primas em formato L..... 20

2.11 Representação espacial de uma caixa dentro do contêiner..... 24

2.12 Representação espacial de duas caixas dentro do contêiner..... 24

2.13 Representação do processo de codificação e decodificação dos cromossomos..... 31

2.14 Representação de uma solução, Bortfeldt e Gehring (1998)..... 32

2.15 Adaptação da representação espacial de Ngoi et al. (1994)..... 34

2.16 Contêiner carregado com carga fracionada em três demandas..... 42

2.17 Carregamento factível para o problema de roteamento de veículos capacitados com restrições de empacotamento tridimensional..... 47

3.1 Representações de árvores construtivas..... 54

3.2 Pseudocódigo para métodos de múltiplos inícios..... 56

3.3 Pseudocódigo da fase construtiva..... 56

3.4 Gráfico das probabilidades, calculadas com diferentes funções de tendência,

para uma LRC com cinco elementos..... 58
3.5 Soluções do exemplo 1..... 60
3.6 Carregamento utilizando cubóides..... 64
3.7 Formação dos espaços vazios após o carregamento do cubóide c_{131}..... 65
3.8 Características adicionais dos espaços vazios: comprimento com suporte..... 65
3.9 Espaço vazio muito longe..... 66
3.10 Variação do grau de reconstrução..... 75
3.11 Algoritmo Múltiplos inícios para PCE3D..... 76
3.12 Algoritmo da heurística construtiva aleatorizada..... 78
3.13 Algoritmo para variação do grau de reconstrução da solução..... 79
3.14 Identificação das dimensões dos espaços vazios..... 82
3.15 Soluções ótimas para as instâncias 9 e 10 de Chien e Deng (2004)..... 89
3.16 Contribuição dos diferentes componentes no desempenho do algoritmo AA1.. 91
3.17 Contribuição dos diferentes componentes no desempenho do algoritmo AA2.. 91
4.1 Pseudocódigo para o método de descida em vizinhança variável 101
4.2 Algoritmo de busca em vizinhança variável para o problema bin packing 3D.... 103
4.3 Solução ótima para instância número 8 de Ivancic et al. (1989)..... 107
5.1 Pseudocódigo para a heurística construtiva Clarke e Wright (1964)..... 118
5.2 Pseudocódigo do algoritmo busca tabu para o problema de roteamento de veículo capacitados 3D..... 122
5.3 Pseudocódigo do algoritmo de múltiplos reinícios para strip packing 3D..... 123

Capítulo 1

Introdução

Em um mercado globalizado e altamente competitivo, a pressão sobre as empresas para desenvolverem cadeias de suprimento (*supply chain*) que respondam rapidamente às necessidades dos clientes é muito acentuada. Para permanecerem competitivas, essas empresas devem reduzir custos estratégicos, táticos e operacionais enquanto continuamente melhoram a qualidade do serviço ao cliente. A chave do sucesso na administração da cadeia de suprimentos está na integração das atividades, cooperação, coordenação e compartilhamento de informações através de toda a extensão da cadeia, dos fornecedores até os clientes. Com a integração das atividades consegue-se uma redução sistemática dos custos dos estoques de produtos em processo e produtos acabados e a distribuição de lotes cada vez menores com grande velocidade e confiabilidade. Neste contexto, o transporte deixa de ser feito pelas próprias empresas produtoras ou por operadores monomodais contratados, para dar lugar aos conceitos de consolidação e unitização de cargas e a logística multimodal.

A consolidação de cargas compreende lotes formados de mercadorias em pequenas quantidades, de diferentes clientes e origens, que se destinam a um mesmo ponto, de modo a possibilitar carregamentos completos. A unitização, por sua vez, diz respeito ao processo usado para facilitar o manuseio e transporte de cargas, pelo qual os volumes são embalados ou fixos a dispositivos específicos como paletes ou contêineres. O transporte multimodal tornou-se possível com o uso de contêineres, que foram desenvolvidos nos anos 50 e aperfeiçoados e adotados em grande escala na década de 1980 como uma espécie de forma padrão de acondicionamento de carga geral.

Um contêiner nada mais é do que uma grande caixa de metal na qual podem ser acondicionadas mercadorias diversas, inclusive aquelas que necessitam de

refrigeração. O seu tamanho obedece a padrões aceitos internacionalmente, em que os de 40 pés de comprimento são os mais utilizados na atualidade. Uma grande vantagem do contêiner é a padronização da carga geral, o que possibilitou inúmeros desenvolvimentos tecnológicos e logísticos. Equipamentos específicos para o manuseio das cargas foram desenvolvidos, sobretudo guindastes para auxiliar na movimentação em pátios de alta produtividade. O mesmo aconteceu com os modais de transportes, em que carretas, vagões e navios foram especializados e ampliados com vistas ao transporte exclusivo por contêineres. A tudo isso, soma-se a contribuição do desenvolvimento de sistemas informatizados, que possibilitaram a redução dos custos administrativos e permitiu, em tempo real, o controle de grandes fluxos de cargas.

O transporte é uma das atividades mais importantes da logística, que pode absorver até dois terços do seu custo, além de criar possibilidades para agregar valor ao produto (Ballou, 1999). Em operações de comércio internacional há uma grande utilização do contêiner, pelas facilidades de manuseio e formas de acondicionamento das cargas, que propiciam o transporte de mercadorias com segurança, inviolabilidade e rapidez.

A organização de cargas em contêineres, seja ela paletizada ou distribuída em caixas, é uma das tarefas mais complexas em empresas que tem no transporte de carga um fator logístico importante e de alto custo. Em geral, métodos manuais utilizados para estimar quantas caixas de diferentes tamanhos podem ser carregadas em um contêiner são simplistas e não apresentam um bom desempenho para carregamentos complexos. Ainda que algumas pessoas envolvidas nessa atividade desenvolvam uma considerável experiência em empacotar caixas eficientemente, invariavelmente o resultado é a sub-utilização da capacidade dos contêineres. Em situações extremas é necessário recarregar as caixas em diferentes contêineres devido a estimações equivocadas de capacidade. Com uma melhor utilização do espaço nos contêineres é possível obter redução nos custos e tempo de carregamento e descarregamento.

Grande parte da literatura sobre carregamento de contêiner aborda o problema de forma estática, sem considerar fatores relevantes para os procedimentos de transporte. Na realidade, o contêiner, uma vez carregado, pode passar por vários modais de transporte até chegar ao seu destino. Situação em que fatores como estabilidade mínima do carregamento e distribuição de peso devem ser levados em consideração.

Neste trabalho são desenvolvidos instrumentos de otimização para problemas de corte e empacotamento tridimensional, bem como a integração destes com roteamento de veículos. Uma nova abordagem, baseada em cubóides de tamanho variável, é utilizada para calcular os padrões de carregamento tridimensional em todos os métodos propostos. Restrições de orientação, estabilidade, centro de gravidade, projeção da base de apoio e múltiplos destinos são consideradas. Para tratar o problema integrado de roteamento de veículos capacitados com restrições de empacotamento tridimensional é proposta uma heurística de busca tabu. Vários testes computacionais são realizados para demonstrar o desempenho das abordagens propostas.

1.1 Objetivo e estrutura do trabalho

O objetivo da presente tese é propor e analisar métodos heurísticos para resolução dos problemas carregamento de contêiner, *bin packing* 3D e roteamento de veículos capacitados com restrições de empacotamento tridimensional. O estudo evidencia aspectos práticos que usualmente são considerados em situações reais.

O trabalho está estruturado em 6 capítulos. No capítulo 2 os problemas estudados são apresentados e, em seguida, é feita uma extensiva revisão bibliográfica dos métodos de resolução. Alguns métodos são descritos em detalhes para facilitar a compreensão dos algoritmos propostos no decorrer do presente estudo. O capítulo 3 descreve os fundamentos e o desenvolvimento de uma heurística de múltiplos inícios para a resolução do problema de carregamento de contêiner. Restrições de cunho prático são tratadas e resultados computacionais comparativos com outras abordagens são utilizados para validar o desempenho da heurística proposta. A heurística desenvolvida neste capítulo aparece como um componente das abordagens utilizadas nos dois capítulos seguintes. No capítulo 4 é estudado o problema *bin packing* 3D e uma heurística inspirada em descida em vizinhança variável é proposta para a sua resolução. No capitulo 5 é desenvolvida uma heurística de busca tabu para resolver o problema de roteamento de veículos capacitados com restrições de empacotamento tridimensional. Por fim, no capítulo 6 conclusões são apresentadas e trabalhos futuros propostos.

Capítulo 2

Definição do problema e revisão bibliográfica

2.1 Introdução

A denominação problemas de corte e empacotamento (PCE) é aplicada a problemas em que uma ou mais unidades grandes, que podem representar um determinado material ou espaço, devem ser divididas em unidades pequenas. Em geral, o objetivo considerado refere-se à minimização do desperdício, ou seja, a quantidade do material ou espaço não utilizado das unidades grandes. Os problemas de corte e empacotamento tridimensional (PCE3D) são uma generalização natural dos problemas clássicos de corte e empacotamento unidimensional e bidimensional, e desta forma pertencem à classe NP-difícil. Isto significa que muito provavelmente não existem métodos ótimos que resolvam estes problemas em um tempo computacional razoável, e por esta razão a maioria dos estudos sobre PCE3D concentram-se em desenvolver métodos heurísticos de resolução.

Existe um grande número de trabalhos que tratam de PCE, mas estes estudos em sua maioria restringem-se a problemas unidimensionais e bidimensionais. O volume de trabalhos publicados sobre problemas tridimensionais é ainda bastante limitado, provavelmente muito mais em razão da complexidade inerente a estes problemas do que pela falta de aplicações práticas. No entanto, desde as primeiras publicações, especificamente sobre PCE3D, no início dos anos 80 até os dias de hoje houve um significativo aumento no número de trabalhos que abordam PCE3D. Muitos destes

trabalhos estudam o problema de carregamento de contêineres utilizados em transportes multimodais.

A diversidade e os diferentes níveis de complexidade dos PCE3D estão intimamente vinculados com as propriedades geométricas associadas às unidades grandes e pequenas. A Figura 2.1 apresenta uma classificação das diferentes propriedades geométricas que podem ser associadas aos PCE3D.

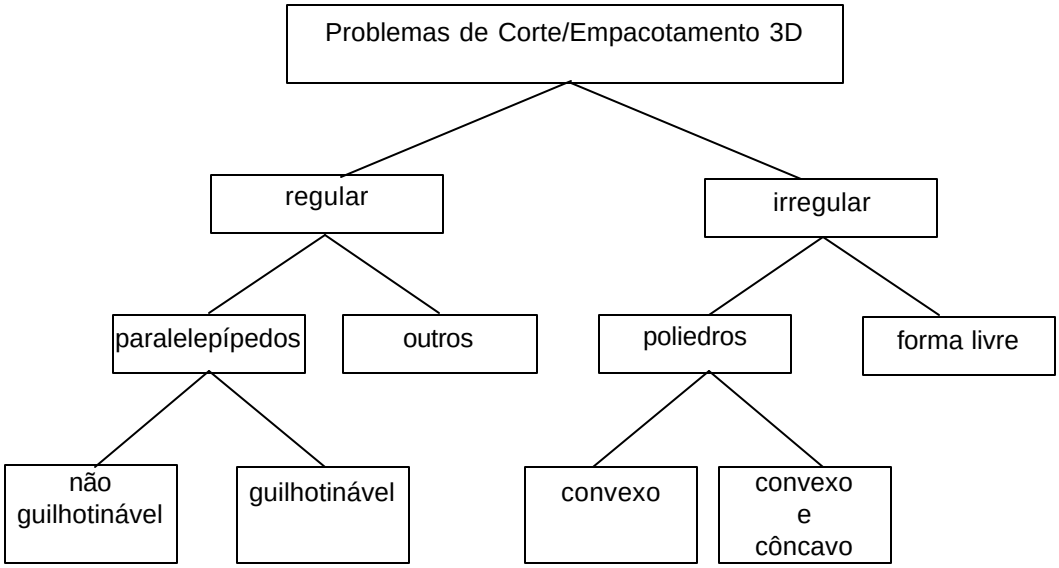


Figura 2.1. Classificação dos problemas de corte e empacotamento segundo suas propriedades geométricas

Não é difícil intuir que problemas que tratam de unidades com formas regulares, em geral, são bem menos complexos de resolver do que aqueles que tratam de unidades com formas irregulares. Analogamente, problemas que utilizam padrões de corte guilhotinados são menos complexos de resolver do que aqueles que utilizam padrões de corte não guilhotinados. Entende-se por corte guilhotinados aquele que, ao ser aplicado sobre um paralelepípedo, produz dois novos paralelepípedos.

Neste trabalho são considerados problemas cujas unidades possuem forma de paralelepípedos e cortes não guilhotinados são permitidos, em que as unidades grandes são denominadas contêineres e as unidades pequenas denominadas caixas ou itens deste ponto em diante, salvo observação ao contrário.

Além das propriedades geométricas, outras restrições de cunho prático devem ser incorporadas aos PCE3D de forma a permitir a aplicação destes em situações reais. Bischoff e Ratcliff (1995a) descrevem doze exigências práticas que podem ser levadas

em consideração quando se deseja modelar PCE3D mais realísticos. A Tabela 2.1 apresenta as doze exigências práticas com uma breve descrição.

Tabela 2.1. Exigências práticas para PCE3D conforme Bischoff e Ratcliff (1995a)

Exigência Prática		Descrição
1.	Orientação	Refere-se à orientação vertical e horizontal das dimensões das caixas. Para uma dada caixa existem seis orientações possíveis, ver Figura 2.2.
2.	Empilhamento	Dada uma determinada caixa, estipula-se o número de caixas que podem ser empilhadas acima, ou mais genericamente, a pressão máxima que pode ser aplicada à parte superior desta caixa sem que ocorra alteração na sua forma.
3.	Manejo	Determina o posicionamento mais adequado para uma caixa dentro do contêiner
4.	Estabilidade	Refere-se ao suporte da base, estabilidade vertical, e das faces laterais, estabilidade horizontal, de cada caixa. A estabilidade vertical pode admitir projeção da base (*overhang*) dentro de um limite estipulado, em geral definido sobre a área da base da caixa. Como exemplo, na pilha de caixas representada na Figura 2.3 a caixa A possui uma boa estabilidade vertical (100% da base possui apoio) e horizontal (três faces laterais possuem apoio) e a caixa B apresenta projeção da base.
5.	Agrupamento de itens	Refere-se à proximidade no posicionamento de caixas semelhantes.
6.	Múltiplos destinos	Considera que a carga do contêiner deve ser dividida segundo os diferentes destinos. Caixas com mesmo destino devem ser posicionadas próximas e a ordem do carregamento deve respeitar, na medida do possível, a ordem em que estas devem ser descarregadas.
7.	Separação de itens	Dentro de um mesmo contêiner alguns itens devem ser posicionados com uma distância mínima entre si. Esta restrição auxilia na modelagem de situações em que a qualidade de alguns produtos pode ser afetada pela proximidade a outros, como produtos alimentícios e químicos.
8.	Carregamento completo de grupos de itens	O carregamento pode ser formado por subconjuntos de itens que possuem uma entidade funcional única, como peças de uma máquina. Neste caso um carregamento só pode ser considerado factível se ao carregar um item de um determinado subconjunto todos os demais itens também são carregados.
9.	Prioridades	O carregamento de alguns itens pode ser mais importante do que outros. A prioridade pode representar o valor agregado aos produtos transportados, que nem sempre é condizente com o volume das caixas.
10.	Complexidade do padrão de carregamento	Padrões complexos de carregamento podem aumentar o tempo utilizado nos procedimentos de carga, bem como impossibilitar o uso de algum maquinário auxiliar.
11.	Limite de peso	Se a carga é densa, o limite de peso dos contêineres pode representar uma restrição mais forte do que o espaço disponível.
12.	Distribuição de peso dentro do contêiner	Para fins de movimentação e transporte é interessante que o centro de gravidade dos contêineres seja próximo do centro geométrico do plano que define a base e tenha a menor cota possível.

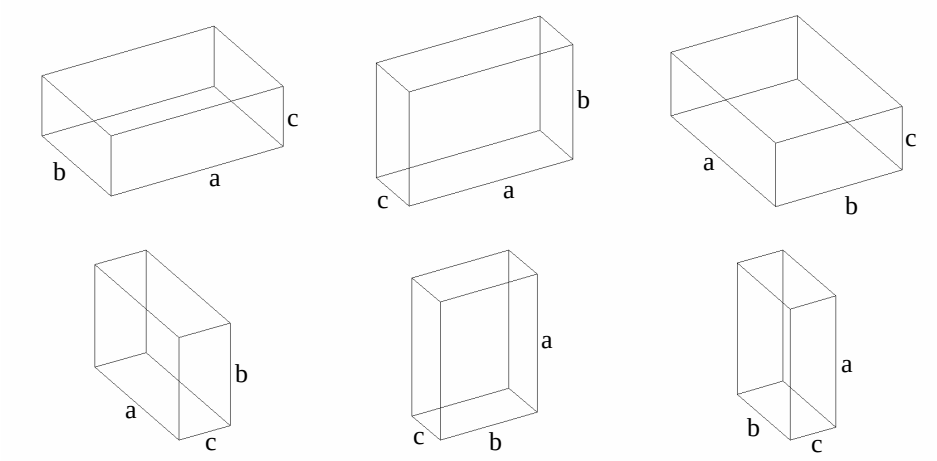


Figura 2.2. Representação das seis possíveis orientações de uma caixa

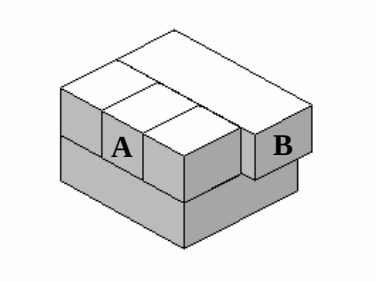


Figura 2.3. Restrições de estabilidade

Algumas das exigências listadas devem ser necessariamente satisfeitas e são denominadas restrições fortes, enquanto outras podem ser satisfeitas na medida do possível e são denominadas restrições fracas. O limite de peso do contêiner, por exemplo, deve ser rigorosamente respeitado e por isso é interpretado como uma restrição forte. Restrições concernentes à orientação e empilhamento também representam restrições fortes, por outro lado, restrições de estabilidade horizontal e complexidade do padrão de carregamento não apresentam um limite claro entre o que é aceitável ou não, e por isso podem ser consideradas como restrições fracas. As restrições fortes delimitam o espaço de soluções factíveis enquanto as restrições fracas podem ser incorporadas à função objetivo que discrimina a utilidade das soluções ou, até mesmo, serem consideradas como outros objetivos (abordagem multiobjetivo).

Muitas das restrições descritas na Tabela 2.1 podem aumentar consideravelmente a complexidade dos PCE3D, no entanto, negligenciá-las por completo na elaboração de um modelo pode acarretar em pouco valor prático. Este é o caso de muitos modelos que objetivam maximizar o espaço ocupado respeitando restrições geométricas (sobreposição de caixas). Um único objetivo e restrições de sobreposição, ainda que relevantes, podem ser insuficientes para representar corretamente um problema prático, uma vez que muitas características potencialmente importantes do problema são desprezadas. Aplicações concretas de PCE3D necessitam levar em consideração várias restrições e medidas de desempenho.

Obviamente nem todas as exigências práticas apresentadas precisam ser levadas em consideração ao mesmo tempo em PCE3D. A seleção das exigências e a forma como são interpretadas dependem fortemente da aplicação que se tem em mente. Por exemplo, restrições de estabilidade de um problema de carregamento de contêiner diferem substancialmente daquelas empregadas em um problema de carregamento de paletes. Em geral, paletes, ao contrário dos contêineres, não dispõem de suporte lateral e a estabilidade deve ser interpretada de forma mais restrita. Carpenter e Dowsland (1985) estabelecem três regras rígidas que devem ser satisfeitas para estabilidade de um padrão de carregamento em paletes. Atualmente existe a possibilidade de utilizar robôs para amarrar as caixas de um palete cingindo um filme ou *stretch*, o que leva a uma menor exigência de estabilidade.

Uma outra consideração que deve ser feita em relação ao aumento da dificuldade na resolução dos PCE3D refere-se à heterogeneidade do conjunto de caixas a ser carregado. Se todas as caixas são iguais o conjunto é considerado homogêneo e, no caso contrário, o conjunto de caixas varia entre fracamente heterogêneo a fortemente heterogêneo. Um conjunto de caixas é dito fracamente heterogêneo se existir um grande número de caixas em cada tipo e fortemente heterogêneo se existir um pequeno número de caixas em cada tipo. Caixas de um mesmo tipo possuem características iguais, tais como dimensões, peso e capacidade de empilhamento. A maior dificuldade em se lidar com conjuntos heterogêneos de caixas reside no fato destes conjuntos dificultarem a obtenção de superfícies planas durante a construção de um padrão de carregamento. Se restrições de estabilidade vertical são impostas, é natural que o volume utilizado do

contêiner decaia de acordo com o grau de heterogeneidade das caixas que compõem o carregamento.

2.2 Problemas abordados

Neste trabalho são abordados três PCE3D, todos eles tendo como base o problema de carregamento de contêiner. Estes problemas são definidos a seguir.

No problema de **carregamento de contêiner** é dado um conjunto de caixas com uma utilidade associada a cada caixa e um único contêiner. O objetivo é escolher um subconjunto de caixas que caiba no contêiner e maximize a utilidade satisfazendo um conjunto de restrições. Se a utilidade de cada caixa é igual ao seu volume, o objetivo é interpretado como maximização do volume ocupado. Mais formalmente, o contêiner é caracterizado pelas dimensões comprimento, largura e altura $L \times W \times H$ e volume V. As caixas são agrupadas em m tipos que podem assumir $k = 1,...,6$ orientações e cada tipo t é identificado por três dimensões espaciais $d_{1tk}, d_{2tk}, d_{3tk}$, um volume v_t e uma quantidade q_t de caixas, $t = 1, ..., m$. Adicionalmente, pode-se associar a cada tipo de caixa um peso w_t e um valor b_{tk} que expressa a pressão, em unidades de peso por área, que a face superior do tipo de caixa t com orientação k pode suportar. Restrições quanto à orientação das caixas são representadas por um conjunto de parâmetros binários, em que $u_{tk} = 1$ se a orientação k da caixa do tipo t é permitida e $u_{tk} = 0$ caso contrário. Sem perda de generalidade, assume-se que as dimensões, tanto do contêiner como das caixas, são inteiros positivos.

No problema ***bin packing* 3D** existe um número suficiente de contêineres idênticos para carregar todo o conjunto de caixas, e o objetivo é minimizar o número de contêineres necessários para esta tarefa. Se os contêineres possuem dimensões diferentes o problema é denominado carregamento de múltiplos contêineres (*Multi-container Loading*), e o objetivo passa a ser a minimização dos custos do carregamento.

O problema de **carregamento e roteamento de veículos** consiste em uma extensão do problema de roteamento de veículos no qual restrições de empacotamento tridimensionais são introduzidas. O objetivo é minimizar o número de veículos utilizados e a distância total percorrida para fazer as entregas das demandas, que neste caso são representadas por conjuntos de caixas.

Em todos os problemas expostos, algumas suposições são assumidas implicitamente:

- Somente caixas com formas de paralelepípedos são consideradas.
- O centro de gravidade das caixas coincide com o centro geométrico.
- As caixas são dispostas ortogonalmente dentro do contêiner, ou seja, cada uma das dimensões das caixas é paralela ou ortogonal a uma das faces do contêiner.

Para localizar as caixas dentro do contêiner utiliza-se um sistema de coordenadas de tal maneira que, quando o contêiner é visto de frente, a origem coincide com o vértice esquerdo inferior posterior e as três coordenadas referem-se a posições ao longo do comprimento, largura e altura, nesta ordem, ver Figura 2.4. Um espaço vazio i destinado à alocação de caixas é delimitado por um paralelepípedo reto com dimensões $X_i \times Y_i \times Z_i$ e volume ve_i, cujas coordenadas (x_i, y_i, z_i) correspondem ao vértice esquerdo inferior posterior. Quando o contêiner está vazio existe apenas um espaço vazio com coordenadas $(0, 0, 0)$ e com as mesmas dimensões do contêiner.

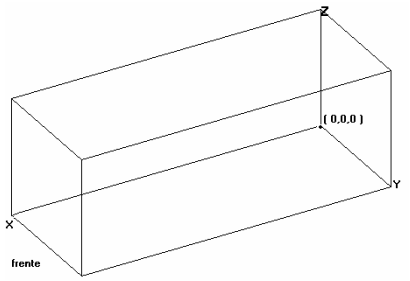


Figura 2.4. Sistema de coordenadas para localizar caixas dentro de um contêiner

Para cada caixa carregada, três novos espaços são criados: o superior, o frontal e o lateral, ver Figura 2.5. O espaço interseção faz parte tanto do espaço frontal como do lateral, mas não é considerado como um espaço vazio para carregamento porque está integralmente contido nestes outros dois. O procedimento que identifica os espaços para carregamento deve atualizar as dimensões dos espaços que possuem intersecção com aquele utilizado, após cada carregamento Os espaços vazios também podem ser descartados se for verificada a impossibilidade de alocação de qualquer uma das caixas ainda não carregadas.

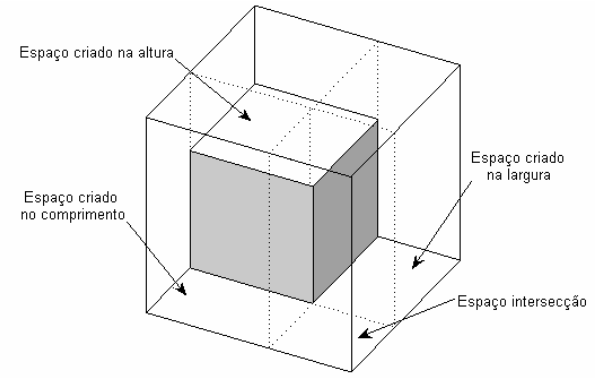


Figura 2.5. Espaços criados devido ao carregamento de uma caixa.

2.2.1 Conjunto de restrições

As restrições para PCE3D consideradas neste trabalho são encontradas em diversos problemas práticos e estão em conformidade com Bischoff e Ratcliff (1995a). O modo como as restrições são interpretadas é descrito a seguir:

R1. **Orientação**: uma ou duas dimensões de algumas caixas não podem ser orientadas verticalmente. Esta restrição modela situações em que, devido aos produtos empacotados nas caixas, instruções do tipo “esta face para cima” são utilizadas. Se nenhuma restrição sobre a orientação de uma caixa for imposta, então esta pode ser carregada com uma das seis orientações possíveis.

R2. **Estabilidade**: uma porcentagem da área da base de cada caixa deve ser apoiada pelo piso do contêiner ou por outras caixas. Para garantir que o centro de gravidade das caixas seja amparado, a porcentagem da base da caixa que deve ser apoiada varia entre 55% a 100%.

R3. **Empilhamento:** assume-se que um peso colocado sobre uma caixa exerce pressão diretamente sobre a área de contato, e que a face superior da caixa do tipo t com orientação k pode suportar uma pressão b_{tk}.

R4. **Distribuição de peso dentro do contêiner:** uma distância mínima entre o centro de gravidade calculado e esperado é imposta.

R5. **Múltiplos destinos:** o carregamento de demandas formadas pelo agrupamento de diferentes itens deve respeitar a ordem com que estes devem ser entregues nos respectivos destinos.

As restrições descritas acima são agrupadas de modo a originar problemas distintos. As restrições R1 e R2 sempre são consideradas e as demais conforme o problema em questão. Deste modo, por exemplo, quando é feita referência ao problema de carregamento de contêiner com restrições de empilhamento R3 subtende-se que as restrições de orientação R1 e estabilidade R2 também são consideradas.

2.3 Métodos de resolução para PCE3D

Esta seção apresenta uma revisão bibliográfica dos métodos de resolução para PCE3D com ênfase em métodos heurísticos. Para expor o assunto uma divisão em métodos exatos e de aproximação, heurísticas, busca em árvore e metaheurísticas é adotada. Quando um trabalho apresenta mais de uma abordagem, este é citado em cada subseção correspondente. A redundância favorece a apresentação dos diferentes métodos e auxilia em um exame comparativo.

Alguns métodos, que fundamentam os algoritmos propostos neste trabalho, são descritos em detalhes.

2.3.1 Métodos exatos e algoritmos de aproximação

A modelagem matemática de restrições para PCE3D, como as descritas na Tabela 2.1, é uma tarefa muito complexa, e modelos matemáticos propostos na literatura negligenciam grande parte destas restrições.

CHEN et al. (1995) propõem um modelo de programação linear inteira mista para o problema genérico de carregamento de contêiner. O modelo compreende um conjunto de caixas e contêineres com dimensões distintas. Restrições de sobreposição, orientação de caixas e balanceamento de peso são tratadas no modelo, no entanto, não é garantido um carregamento estável. Ainda que os autores afirmem que restrições de estabilidade podem ser incorporadas, nenhuma proposta neste sentido é apresentada. O objetivo do modelo é minimizar o espaço total não utilizado, com a possibilidade de considerar casos especiais como a seleção de um contêiner dentre vários e contêiner

com comprimento variável. O modelo é validado com a resolução de instâncias com 6 caixas e fica evidente a impossibilidade de aplicá-lo em problemas reais. Os autores concluem que procedimentos mais eficientes devem ser utilizados para tratar problemas de maior porte, uma vez que o número de variáveis e restrições crescem quadraticamente com o número de caixas.

Miyazawa e Wakabayashi (1997) apresentam um algoritmo de aproximação[1] para o problema de carregar caixas com orientação fixa em um contêiner com largura e comprimento limitados e altura ilimitada, em que o objetivo é minimizar a altura do empacotamento. Neste trabalho é provado que o algoritmo proposto tem razão de aproximação assintótica entre 2,5 e 2,67, melhorando os resultados anteriormente conhecidos e respondendo à questão levantada por Li e Cheng (1992) sobre a existência de tal algoritmo. Em Miyazawa e Wakabayashi (2000), os autores apresentam um algoritmo com razão de aproximação assintótica próxima de 2,67 para o problema em que as caixas possuem orientação vertical fixa e a base pode girar 90^o. Este resultado é melhorado em Miyazawa e Wakabayashi (2004), com uma razão de aproximação assintótica de 2,64.

Lai et al. (1998) apresentam um modelo baseado em grafos para o problema de carregamento de contêiner com múltiplos destinos com o objetivo de minimizar o comprimento dos setores do contêiner utilizados para armazenar as demandas de cada destino. O modelo não considera restrições de estabilidade, somente restrições de sobreposição. O problema é resolvido com métodos exato e aproximado em que um grafo é construído com base no algoritmo de Christofides e Whitlock (1977) para PCE bidimensionais. Dentro de contêiner, cada posição factível de uma caixa corresponde a um vértice do grafo e dois vértices são ditos adjacentes se não possuem o mesmo tipo de caixa e intersecção de espaço. Deste modo, o problema a ser resolvido é equivalente a encontrar o máximo clique do grafo gerado, isto é, o objetivo passa a ser encontrar o maior conjunto de vértices adjacentes dois a dois. Métodos de redução do espaço de soluções são utilizados. Os testes computacionais utilizam instâncias com quatro tipos de caixas e três demandas, em que a quantidade máxima de caixas em cada demanda não ultrapassa três unidades.

[1] Para uma introdução a algoritmos de aproximação veja Carvalho et al. (2001).

Martello et al. (2000) propõem um algoritmo exato para o carregamento de um único contêiner e deste derivam um algoritmo *branch-and-bound* exato para o problema *bin packing* 3D. Restrições de estabilidade não são consideradas e as caixas possuem orientação fixa, contudo, cortes não guilhotinados são permitidos. Os testes computacionais apresentados mostram que instâncias com um grande número de caixas por contêiner são difíceis de resolver e que casos com até 90 caixas podem ser resolvidos em tempo computacional razoável.

Cecílio (2003) estende vários modelos matemáticos para PCE bidimensionais encontrados na literatura para tratar PCE3D, no entanto, nenhum destes modelos considera explicitamente restrições de estabilidade. O grande número de variáveis e restrições em relação ao número de caixas são características comuns entre os modelos, restringindo a utilidade destes quase que exclusivamente para fins teóricos.

Hifi (2004) propõe dois algoritmos exatos para resolver o problema de corte tridimensional não restrito, isto é, uma quantidade ilimitada de cada tipo de peça para corte é disponível. Todos os cortes são guilhotinados e restrições de estabilidade não são consideradas. O autor considera casos com palete único e com diferentes tipos paletes. Os algoritmos são generalizações de abordagens para o problema de corte bidimensional.

2.3.2 Heurísticas

Abordagens heurísticas são alternativas mais viáveis para tratar PCE3D reais, uma vez que uma solução de boa qualidade conseguida dentro de um tempo razoável é suficiente para atender a maioria das aplicações práticas. Segundo Pisinger (2002), as abordagens heurísticas mais comumente utilizadas podem ser classificadas de acordo com o método utilizado para construir o padrão de carregamento, sendo possível identificar algoritmos que utilizam:

- Paredes virtuais (camadas verticais e/ou horizontais);
- Pilhas de caixas;
- Cortes guilhotinados;
- Cubóides.

Paredes virtuais são utilizadas para construir camadas verticais ou horizontais que proporcionam uma redução no espaço de soluções e possibilitam o uso de estruturas

de dados mais simples na implementação do algoritmo. O contêiner é preenchido camada por camada.

Uma camada vertical j corresponde a uma seção do contêiner que contém todos os espaços vazios com coordenadas que pertencem ao conjunto $CV_j = \{(x, y, z) \mid x_j \le x \le x_{j+1}, 0 \le y \le W, 0 \le z \le H\}$, tal que $(x_j, 0, 0)$ e $(x_{j+1}, 0, 0)$ representam as posições do espaço vazio das camadas verticais j e $j+1$, respectivamente. Em geral, mas não necessariamente, o comprimento da camada é determinado pela caixa alocada na posição $(x_j, 0, 0)$, mais especificamente, se d_{1tk} representa o comprimento da caixa do tipo t com orientação k então $x_{j+1} - x_j = d_{1tk}$ e esta caixa é denominada "caixa que determina a camada" (*Layer determining box – LDB*). A Figura 2.6 exemplifica o espaço correspondente à primeira camada vertical de um contêiner. Procedimentos baseados em camadas verticais são comuns em heurísticas que tratam problemas de carregamento de contêiner, principalmente quando a aplicação considera conjuntos de caixas fortemente heterogêneos.

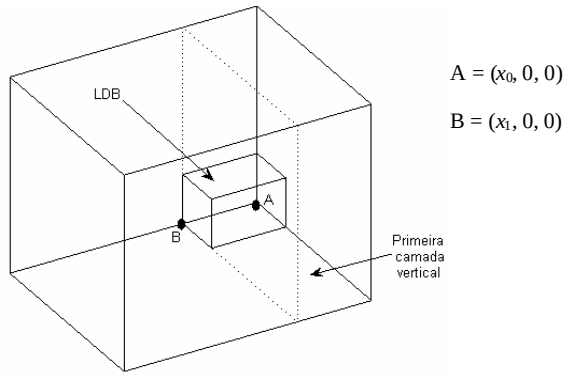


Figura 2.6. Representação da primeira camada vertical de um contêiner

Camadas horizontais são definidas de forma análoga às camadas verticais. Uma camada horizontal j corresponde a uma seção do contêiner que contém todos os espaços vazios com coordenadas que pertencem ao conjunto $CH_j = \{(x, y, z) \mid 0 \le x \le L, 0 \le y \le W, z_j \le z \le z_{j+1}\}$, tal que $(0, 0, z_j)$ e $(0, 0, z_{j+1})$ representam as coordenadas do espaço vazio inicial das camadas horizontais j e $j+1$, respectivamente. Procedimentos que utilizam camadas horizontais são úteis para construir padrões de

carregamento em que a estabilidade vertical é um fator relevante, como ocorre em problemas de carregamento de paletes.

Pilhas de caixas possibilitam decompor PCE3D em dois subproblemas, o problema tridimensional de construção de pilhas e o problema bidimensional de alocação de pilhas no contêiner. Uma pilha é construída a partir da escolha da caixa posicionada no piso do contêiner, denominada caixa base. A seguir, cada caixa acrescida deve ser colocada em cima de outra pertencente à pilha, de maneira que a sua base seja integralmente suportada por uma ou mais caixas abaixo, ver Figura 2.7. Uma vez que as caixas tenham sido distribuídas em pilhas o próximo passo é resolver o problema de empacotamento bidimensional para dispor as pilhas no piso do contêiner. Este procedimento favorece o tratamento de algumas restrições, como restrições de peso, mas em geral produzem padrões de carregamento com pouca estabilidade horizontal e, quando a carga é fracamente heterogênea, baixa ocupação do volume do contêiner.

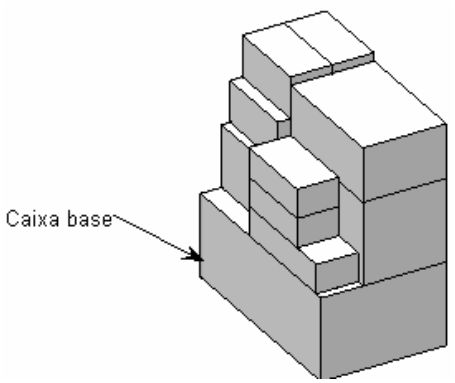


Figura 2.7. Exemplo de padrão de carregamento do tipo pilha

Cortes guilhotinados são aqueles que quando realizados sobre um paralelepípedo produzem dois paralelepípedos. Neste tipo de padrão de carregamento todas as caixas podem ser distinguidas com uma seqüência de cortes guilhotinados. Na Figura 2.8, (a) e (b) apresentam exemplos de padrões de corte guilhotinado e não guilhotinado, respectivamente. Para problemas de PCE3D a restrição de cortes guilhotinados ocorre muito raramente na prática, ao contrário dos problemas bidimensionais. Uma exceção a esta regra pode ser encontrada em Hasamontr (2003), que apresenta um sistema para automatizar cortes tridimensionais guilhotinados em uma indústria moveleira. Às vezes impõem-se esta restrição artificialmente para simplificar o

método de resolução, admitindo-se que a solução guilhotinada pode ser competitiva com a solução aproximada não guilhotinada.

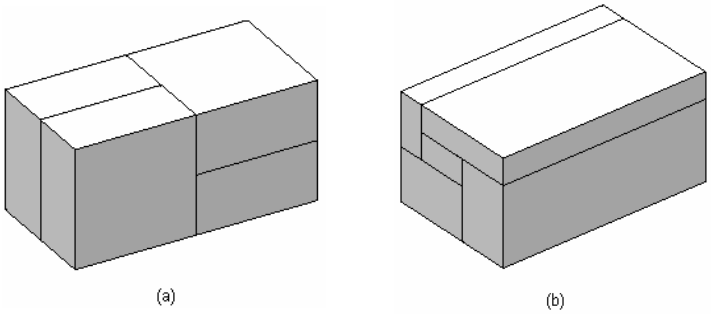


Figura 2.8. Padrão de corte guilhotinado (a) e não guilhotinado (b)

Cubóides são blocos homogêneos formados por caixas do mesmo tipo com mesma orientação. As caixas que formam um cubóide devem ser dispostas ao longo das três dimensões do contêiner de modo a formarem um paralelepípedo, e podem ser representadas por uma tripla $c_{tki} = (c^x_{tki}, c^y_{tki}, c^z_{tki})$, na qual os elementos representam o número de caixas do tipo t com orientação k, na correspondente dimensão, que podem ser carregados no espaço vazio i. O número de caixas que compõe um cubóide é definido por $|c_{tki}| = c^x_{tki} \cdot c^y_{tki} \cdot c^z_{tki}$. A Figura 2.9 apresenta um cubóide $c_{tki}=(2,2,3)$ com $|c_{tki}| = 12$ caixas do tipo t e orientação k, em que o comprimento e a largura são definidos por duas caixas e a altura por três, $c^x_{tki} = c^y_{tki} = 2$ e $c^z_{tki} = 3$. Observe que uma caixa pode ser considerada um cubóide com representação $(1, 1, 1)$. O volume de um cubóide é dado pelo produto do volume do tipo de caixa e o número de caixas que compõe o cubóide, $v_t \cdot |c_{tki}|$.

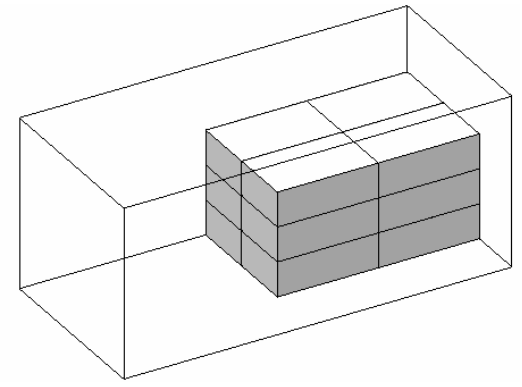


Figura 2.9. Cubóide (2, 2, 3) formado por caixas do tipo *t* com orientação *k*

George e Robinson (1980) propõem o primeiro método heurístico para o problema de carregamento de contêiner com um conjunto heterogêneo de caixas e restrições de sobreposição e estabilidade vertical. Nenhuma restrição é feita sobre a orientação das caixas. A heurística utiliza um procedimento baseado em camadas verticais, proporcionando que espaços não ocupados em uma camada possam ser utilizados nas camadas subseqüentes. As caixas são ordenadas de modo a construir um padrão de carregamento que privilegia o posicionamento próximo de caixas iguais, classificando os tipos de caixas em "*open"* se já foram utilizadas e "*closed*" se ainda não foram utilizadas. Duas regras de ordenação para escolher caixas com status *open* são apresentadas. A primeira regra dá prioridade para o tipo de caixa com maior quantidade disponível, enquanto a segunda regra seleciona o tipo de caixa com a maior das menores dimensões. Caso não existam tipos com status *open,* seleciona-se o tipo de caixa com maior prioridade entre todas as caixas disponíveis. Os resultados computacionais sugerem que a primeira regra é ligeiramente superior, mas os autores não manifestam qual critério é genericamente mais eficiente. Alternativamente, uma variedade de soluções pode ser obtida escolhendo-se diferentes tipos de caixas e orientações para determinar o comprimento de cada camada. O algoritmo é descrito através de fluxogramas que explicam em um mesmo sistema os procedimentos de identificação de espaços vazios dentro do contêiner, escolha e carregamento das caixas, dificultando a compreensão deste como um todo. Uma descrição mais sucinta e clara pode ser obtida se os procedimentos de identificação de espaços vazios são considerados à parte dos procedimentos de escolha e carregamento das caixas (Chien e Wu, 1999). George e Robinson testam a heurística com quinze instâncias diferentes. Um exemplo detalhado é apresentado com 784 caixas distribuídas em oito tipos diferentes e um contêiner com dimensões $5793 \times 2236 \times 2286$ (em milímetros). A melhor solução obtida para este exemplo carrega 783 caixas, com 89,74% de volume ocupado, deixando apenas uma caixa fora do contêiner.

Bischoff e Marriott (1990) examinam quatorze heurísticas para o problema de carregamento de contêiner baseadas em variantes do método de George e Robinson (1980) e duas adaptações do método para carregamento de paletes de Bischoff e Dowsland (1982). O estudo não considera restrições adicionais além de sobreposição e

estabilidade vertical, e o objetivo é definido como a minimização do comprimento do contêiner necessário para acomodar todo o conjunto de caixas. Para avaliar o desempenho das heurísticas os autores utilizam dois grupos de instâncias, cada um com 1, 2, 3, 4, 5, 10, 15 e 20 diferentes tipos de caixas. No primeiro grupo de instâncias o número total de caixas é constante, 500 caixas, enquanto no segundo grupo o volume total das caixas é constante. Nenhuma das heurísticas pode ser dita claramente superior às demais, e uma abordagem que aplica seqüencialmente, em qualquer ordem, as quatorze heurísticas é proposta como a melhor alternativa. Dowsland e Dowsland (1992) observam que não é surpresa a falta de dominância entre as heurísticas deste tipo, uma vez que todas elas são compostas de uma série de regras *ad hoc* derivadas do senso comum que eventualmente não são apropriadas para uma determinada situação.

Han et al. (1989) mostram que a idéia de paredes virtuais não precisa ficar restrita à definição de camadas verticais. Um algoritmo que iterativamente constrói padrões de carregamento a partir de prismas em formato L é proposto para o problema de carregamento de um único contêiner com caixas idênticas, ver Figura 2.10. Inicialmente, toda a base e uma das faces verticais do contêiner são escolhidas para formar o prisma e, a seguir, o arranjo de caixas é calculado por programação dinâmica (semelhante à abordagem para problemas bidimensionais de Steudel, 1979). O preenchimento de uma seqüência de prismas em formato L gera um padrão de carregamento formado por cortes guilhotinados. Para validar o desempenho do algoritmo é utilizada uma instância com um contêiner de dimensões $48"\times42"\times40"$ e caixas com $11"\times6"\times6"$. Os autores obtêm uma solução com 195 caixas carregadas e 95,16% de utilização do volume do contêiner, superior ao resultado publicado por *US General Services Administration* para o mesmo problema com 82,50% de aproveitamento do volume. Devido ao reduzido número de testes não é possível verificar a validade da abordagem.

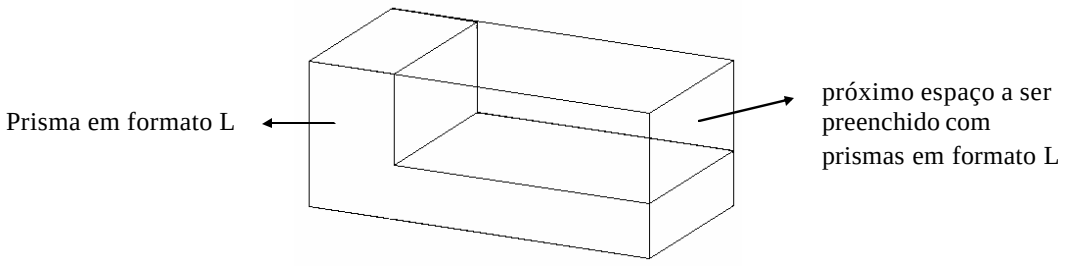


Figura 2.10. Seqüência do carregamento com primas em formato L

Gehring et al. (1990) também apresentam um algoritmo baseado em camadas verticais para carregamento de um conjunto de caixas fortemente heterogêneo em um único contêiner. Uma ordenação baseada em volume é utilizada para determinar a seqüência com que as caixas são carregadas e, uma vez escolhida a LDB da camada, os espaços vazios são preenchidos a partir de um elaborado conjunto de regras com critérios baseados na maximização do volume do espaço vazio preenchido. O algoritmo prevê interação com o usuário de modo a permitir a escolha da orientação da primeira caixa da camada e alteração na ordenação das caixas antes que o procedimento de resolução seja iniciado. As camadas verticais que compõem o padrão de carregamento são isoladas, ou seja, os espaços vagos em uma camada são descartados e, desta forma, a ordem em que as camadas são dispostas pode ser alterada para balancear o peso no contêiner. Duas instâncias são utilizadas nos resultados computacionais e nenhuma comparação com outras abordagens é apresentada. Em Gehring e Bortfeldt (1997) resultados computacionais mais detalhados são apresentados, mas os algoritmos são avaliados considerando exclusivamente o critério de maximização do espaço ocupado. Davies e Bischoff (1999) examinam a estabilidade do padrão de carregamento gerado pelo algoritmo de Gehring et al. e observam que não existem limites para projeção da base das caixas e, teoricamente, algumas caixas podem ser carregadas sem qualquer suporte da base. O estudo demonstra sérias desvantagens no que se refere à utilização do espaço e estabilidade do carregamento, no entanto, revela uma boa distribuição de peso no contêiner devido à utilização de camadas verticais isoladas.

Haessler e Talbot (1990) apresentam uma heurística que inicialmente constrói pilhas de caixas e em seguida posiciona estas pilhas no piso do contêiner. O trabalho é orientado de maneira a atender as necessidades de uma empresa de grande porte que recebe pedidos de clientes e os despacha em contêineres de vários tamanhos, dependendo do tipo de produto solicitado e das exigências impostas. O projeto da heurística considera caixas de baixa densidade, expressa em unidades de massa por unidades de volume, e procura favorecer padrões de carregamento que evitem riscos de danos e coloquem os produtos de um mesmo pedido tão próximo quanto possível. Os autores observam que as pilhas favorecem a diminuição da complexidade e tempo do carregamento. Um exemplo numérico que utiliza dados reais é apresentado para ilustrar o os passos da heurística. Um aumento significativo na utilização dos veículos e uma

melhora na qualidade de atendimento dos clientes são as principais vantagens observadas pela empresa patrocinadora do trabalho.

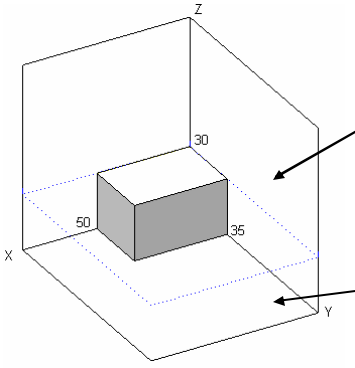
Mais recentemente, Cecílio (2003) apresenta cinco métodos de resolução para o problema de carregamento de contêiner. Estes métodos são compostos por um refinamento do algoritmo de George e Robinson (1980) e o uso de diferentes critérios de classificação das caixas disponíveis para carregamento. O refinamento consiste em modificar o procedimento que escolhe o tipo de caixa que é utilizado para preencher os espaços vazios das camadas verticais. Cinco critérios de classificação são organizados em 60 arranjos, cada um com três critérios, e duas versões para criar o padrão de carregamento são propostas utilizando estes arranjos. A primeira versão, denominada versão Arranjo, consiste em executar a heurística utilizando um arranjo de classificação de caixas diferente em cada execução, fornecendo como solução final o melhor resultado obtido dentre todas as iterações em termos de volume empacotado. A segunda versão, denominada versão Camada, utiliza diferentes arranjos de critérios de classificação das caixas para gerar camadas verticais no contêiner, aquela que obtiver melhor avaliação é incluída no padrão de empacotamento. Combinando o algoritmo de George e Robinson e os procedimentos propostos, os cinco métodos são os que seguem:

1. Heurística de George e Robinson com refinamento.
2. Versão Arranjo.
3. Versão Arranjo com refinamento.
4. Versão Camada.
5. Versão Camada com refinamento.

Os testes computacionais compreendem instâncias geradas pela autora, instâncias da literatura de corte e empacotamento e exemplos reais de carregamento de contêiner. Todas as heurísticas propostas foram capazes de empacotar as 784 caixas do exemplo apresentado em (George e Robinson, 1980) e obtiveram desempenho igual ou superior a um software, fornecido por uma empresa de consultoria, para otimização de empacotamento de caixas em paletes e contêineres. Em (Cecílio e Morabito, 2004) mais testes computacionais são apresentados, incluindo as instâncias de Morabito e Arenales (1994).

Uma característica em comum entre heurísticas que utilizam paredes virtuais é que a seqüência com que as caixas são carregadas reproduz a seqüência do carregamento físico, simulando procedimentos usuais de carregar caixas do fundo para frente do contêiner ou em camadas horizontais a partir do piso. Ngoi et al. (1994) apresentam uma heurística construtiva que utiliza uma representação espacial para modelar o processo de carregamento. Esta representação espacial permite que todos os espaços vazios sejam avaliados, independentemente da ordem com que as caixas são carregadas. Uma matriz tridimensional é utilizada para representar as posições e dimensões de todas as caixas, bem como os espaços vazios. Esta matriz é composta por uma cadeia de matrizes bidimensionais que representam camadas horizontais do contêiner. Os elementos das matrizes bidimensionais representam as caixas empacotadas a partir de um número de identificação, elementos com valor zero representam espaços vazios. As Figuras 2.11 e 2.12 mostram as matrizes associadas ao carregamento de uma e duas caixas, respectivamente, em um contêiner de dimensões $90 \times 120 \times 100$ em uma dada medida de comprimento. Conforme mais caixas são empacotadas, novas matrizes podem ser criadas e novas colunas e linhas podem ser adicionadas às matrizes já existentes. A primeira linha e coluna das matrizes são reservadas para armazenar as coordenadas dos espaços vazios, em especial, o primeiro elemento determina a cota da superfície representada pela matriz. Deste modo, a matriz $M_{0,30}$ da Figura 2.11 representa a projeção vertical dos espaços compreendidos entre as alturas 0 e 30 do contêiner. Da mesma forma, a matriz $M_{30,100}$ representa a projeção vertical dos espaços compreendidos entre as alturas 30 e 100 do contêiner. O espaço ocupado pelo tipo de caixa é assinalado com o identificador X_1. Os espaços para carregamento são delimitados por um procedimento que reúne as informações contidas nas matrizes bidimensionais através da projeção vertical de superfícies horizontais. Para um melhor entendimento deste procedimento, considere agora que um tipo de caixa com identificação X_2 é adicionada ao carregamento, ver Figura 2.12. A partir da matriz $M_{0,23}$ verifica-se que X_2 está localizado no espaço com coordenadas $(0,35,0)$ e dimensões $38 \times 40 \times 23$. Isto pode ser compreendido observando a primeira linha e coluna da matriz. Na primeira linha, o elemento com valor 35 corresponde à coordenada y e a diferença 75 – 35 = 40 à largura do espaço ocupado. Como a caixa aparece na segunda linha da matriz, subentende-se que o valor da coordenada x é zero e que o comprimento do

espaço vazio é 38 – 0 = 38. A altura do espaço ocupado pela caixa X_2 pode ser identificada na matriz $M_{0,23}$, uma vez que esta matriz representa a superfície com maior cota que possui a referida caixa. Os autores propõem um algoritmo para o problema de carregamento de contêiner que avalia todos os espaços vazios disponíveis para proceder o carregamento de uma caixa. A solução final, armazenada na representação espacial, é convertida em um arquivo *script* de Autocad para apresentação gráfica. Os resultados computacionais utilizam as instâncias de Loh e Nee (1992) e quinze instâncias propostas pelos autores, que consistem de conjuntos de 100 a 200 caixas divididas entre 6 a 10 diferentes tipos. Os resultados demonstram que o volume utilizado, em geral, é superior a 80% do volume do contêiner.



$M_{30,100}$ =

100	35	120
50	0	0
90	0	0

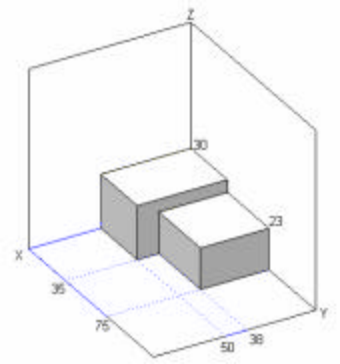
Matriz que representa a camada horizontal compreendida entre Z= 30 e Z = 100.

$M_{0,30}$ =

30	35	120
50	X_1	0
90	0	0

Matriz que representa a camada horizontal compreendida entre Z= 0 e Z = 30, sendo X_1 a identificação do tipo de caixa.

Figura 2.11. Representação espacial de uma caixa dentro do contêiner



$M_{0,23}$ =

23	35	75	120
38	X_1	X_2	0
50	X_1	0	0
90	0	0	0

Z = 0 a 23

$M_{30,100}$ =

100	35	75	120
38	0	0	0
50	0	0	0
90	0	0	0

Z = 30 a 100

$M_{23.30}$ =

30	35	75	120
38	X_1	0	0
50	X_1	0	0
90	0	0	0

Z = 23 a 30

Figura 2.12. Representação espacial de duas caixas dentro do contêiner

Bischoff e Ratcliff (1995a) observam que as heurísticas propostas na literatura utilizam fortes simplificações da realidade, sendo que algumas restrições práticas podem ser facilmente incluídas e outras exigem o desenvolvimento de algoritmos completamente novos. Os autores apresentam duas heurísticas para carregamento de um único contêiner para exemplificar o tratamento de algumas das doze exigências práticas sugeridas no trabalho. A primeira heurística proposta evidencia a estabilidade do carregamento para transporte e utiliza a metodologia descrita em Bischoff et al. (1995). Este método utiliza o princípio de construir camadas horizontais com dois tipos diferentes de caixas a partir do piso do contêiner. As caixas que formam cada camada são escolhidas segundo um critério de maximização da área utilizada. A utilização de camadas horizontais para construir o padrão de carregamento consegue suprimir as desvantagens, sob o ponto de vista da estabilidade, dos procedimentos que utilizam camadas verticais. A segunda heurística é projetada para tratar carga fracionada, ou seja, o contêiner deve ser carregado com grupos de itens com diferentes destinos. A metodologia consiste em produzir padrões de carregamento densos, em que itens com o mesmo destino são carregados próximos e a ordem de carregamento respeita a seqüência de entrega. As caixas são carregadas uma a uma, obedecendo a um esquema de ordenação baseado em quatro critérios. Dado um tipo de caixa t com orientação k e um espaço vazio i com coordenadas (x_i, y_i, z_i) e dimensões $X_i \times Y_i \times Z_i$, o esquema de ordenação pode ser formulado da maneira seguinte.

1. Critério principal: maior potencial de utilização do espaço

   $$u = g \cdot [v_t / ve_i]$$

   em que v_t é o volume da caixa do tipo t, ve_i é o volume do paralelepípedo que determina o espaço vazio i e

   $$g = \min\left(\left\lfloor \frac{Z_i}{d_{3tk}} \right\rfloor, \hat{q}_t\right),$$

   em que $\hat{q}_t$ é a quantidade de caixas do tipo t disponíveis para carregamento.
2. Critério de desempate 1: menor proeminência ao longo do comprimento

$$p = d_{1tk} + x_i.$$

3. Critério de desempate 2: caixa com maior volume v_t .
4. Critério de desempate 3: espaço vazio com menor coordenada y_i.

O critério principal favorece a construção de pilhas de caixas do mesmo tipo, enquanto o primeiro critério de desempate favorece a obtenção de uma frente de carregamento homogênea. O terceiro critério tem por fim carregar o mais cedo possível as caixas de maior volume e o quarto critério, de menor importância, é uma regra de seleção quando todas as demais falham. A representação espacial proposta por Ngoi et al. (1994) é utilizada para facilitar a identificação dos espaços vazios do contêiner. Em razão do limitado número de instâncias disponíveis para realizar testes comparativos, os autores propõem um algoritmo para gerar instâncias com um número arbitrário de caixas que podem ser facilmente reproduzidas. Para realizar os testes computacionais são geradas 700 instâncias que compreendem 3, 5, 8, 10, 12, 15 e 20 tipos de caixas e um contêiner com dimensões $587 \times 233 \times 220$ (as dimensões em cm de um 20 ft ISO contêiner). Os resultados confirmam a superioridade de cada heurística nas características que procuram evidenciar, ou seja, estabilidade e padrões de carregamento densos.

Chien e Deng (2004) propõem um sistema de apoio a decisão com interface gráfica que incorpora uma heurística para carregamento de um único contêiner. Tanto a interface como a heurística são implementadas em Matlab®, o aplicativo resultante é utilizado por empresas de grande porte em Taiwan. A representação espacial de Ngoi et al. (1994) é utilizada e o padrão de carregamento pode ser gerado por meio de camadas verticais ou horizontais, e, em qualquer caso, as caixas são carregadas segundo uma ordenação baseada em cinco critérios. Os testes computacionais utilizam onze instâncias reais de empresas de transporte de Taiwan, demonstrando o melhor desempenho do algoritmo em relação aos procedimentos manuais.

Lim et al. (2005) apresentam uma heurística para o problema de carregamento de um único contêiner com objetivo de maximizar o volume ocupado. O problema é dividido em seleção de caixas, seleção de espaços, orientação da caixa e geração de novos espaços vazios. Especializações para tratar instâncias homogêneas e heterogêneas são apresentadas. Os espaços vazios produzidos com o carregamento de caixas não possuem intersecção, ver Figura 2.5. Para proceder o carregamento as caixas

são ordenadas segundo o volume, comprimento do diâmetro e perímetro. O desempenho da heurística proposta, se comparada com Bischoff e Ratcliff (1995a), é inferior (2,5% em média) para o conjunto de instâncias de Loh e Nee (1992) e superior (4,7% em média) para o conjunto de instâncias de Bischoff e Ratcliff (1995a).

2.3.3 Busca em árvore e metaheurísticas

Heurísticas construtivas conseguem simular procedimentos físicos usualmente utilizados para carregamento de caixa em contêineres. No entanto, estas heurísticas geralmente são projetadas para considerar especificamente algumas características associadas ao carregamento, não sendo possível observar dominância entre estes procedimentos quando as características do carregamento abrangem um grande número de situações (Dowsland e Dowsland, 1992). Poucas heurísticas de melhoria são encontradas na literatura, muito provavelmente devido à dificuldade de definir uma vizinhança com movimentos promissores que opere diretamente sobre uma solução construída. Algoritmos de busca em árvore e metaheurísticas são mais genéricos e flexíveis, sendo o desafio especializá-los para um determinado problema ou classe de problemas.

Busca em árvore é uma forma bastante difundida de resolver problemas combinatórios, em que heurísticas construtivas gulosas são um caso particular deste método. Como explorar todos os nós folhas de uma árvore é impraticável devido ao custo computacional envolvido, faz-se necessário utilizar diferentes algoritmos para explorar parcialmente a árvore de forma eficiente.

Metaheurísticas são estratégias genéricas inteligentes que supervisionam procedimentos heurísticos e promovem um alto rendimento. O termo metaheurística foi utilizado pela primeira vez por Glover (1986) no artigo seminal sobre busca tabu. Uma visão geral do estado da arte em metaheurísticas pode ser encontrada em Blum e Andrea (2003).

Carregamento de contêiner

Morabito e Arenales (1994) tratam o problema de maximizar o volume ocupado por um conjunto de caixas em um único contêiner com restrições implícitas de estabilidade vertical. Somente cortes guilhotinados são permitidos. O trabalho apresenta

métodos heurísticos para PCE3D não-restritos e restritos, isto é, o número de caixas de cada tipo é ilimitado e limitado, respectivamente. Um dos métodos é baseado na representação do espaço de soluções em um grafo-e/ou, e consegue encontrar a solução ótima para o exemplo proposto por George e Robinson (1980), na qual é possível carregar 784 caixas.

Pisinger (2002) apresenta uma extensão do algoritmo de George e Robinson (1980) em que somente cortes guilhotinados são permitidos, restrições de estabilidade vertical não são consideradas e as caixas podem assumir qualquer orientação. O algoritmo proposto constrói padrões de carregamento por intermédio de camadas verticais que são preenchidas por subcamadas com orientação horizontal ou vertical. Uma busca por melhores soluções é feita através da variação do comprimento da camada vertical, ou seja, escolhendo-se diferentes LDB's. Na impossibilidade de enumerar todos os comprimentos possíveis de camadas, uma busca em profundidade com largura limitada é implementada. Cada nó da árvore representa uma solução parcial que possui ramificações que correspondem a diferentes comprimentos da camada subseqüente, ordenados segundo um determinado critério. Para evitar tempos computacionais excessivos quando as caixas são muito pequenas em relação ao contêiner, limitantes são impostos para restringir a quantidade de camadas avaliadas. Um procedimento é proposto para gerar instâncias com características encontradas em problemas de carregamento de contêiner reais. Cada instância considera um contêiner com dimensões $590 \times 230 \times 230$ (em centímetros). Instâncias homogêneas envolvem entre 20 a 700 caixas, instâncias fortemente heterogêneas entre 70 a 100 caixas, cada uma com dimensões diferentes, e instâncias fracamente heterogêneas envolvem entre 50 a 150 caixas distribuídas em 20 tipos diferentes. Comparações com outros algoritmos encontrados na literatura são utilizadas para validar o desempenho do algoritmo proposto. O volume utilizado é superior a 95% para instâncias em que o volume total das caixas ultrapassa 200% do volume do contêiner. Este tipo de instância possui muitas caixas de cada tipo e pode ser considerado de fácil resolução por se assemelhar a um problema irrestrito.

Eley (2002) propõe uma abordagem para o problema de carregamento de contêiner que constrói padrões de carregamento com cubóides. Segundo o autor, padrões de carregamento com cubóides possuem características desejáveis do ponto de vista prático, conforme segue:

- o problema não é resolvido como se fosse um “quebra-cabeça” tridimensional e as instruções para carregar as caixas tendem a ser mais simples;
- blocos homogêneos são fáceis de carregar e possibilitam uma redução no tempo de carregamento;
- as restrições de empilhamento são satisfeitas mais facilmente.
- dentro de uma estrutura de blocos é mais difícil ocorrer deslizamento de caixas, propiciando um aumento na estabilidade dinâmica da carga.

Neste algoritmo, uma heurística construtiva gulosa gera os cubóides escolhendo as caixas com maior volume primeiro e examinando todos os espaços vazios dentro do contêiner. As caixas podem assumir todas as orientações permitidas. O bloco é designado para o espaço vazio no qual, após o carregamento, a soma do volume dos espaços que não podem ser preenchidos com as caixas remanescentes é mínima. Uma busca em árvore é implementada para melhorar o desempenho do algoritmo, na qual o nó raiz representa o contêiner vazio, os nós folhas uma solução completa e os demais nós soluções parciais. Considerando um dado espaço vazio, as seis orientações possíveis e m tipos de caixas, cada nó pode ser ramificado em $6 \cdot m$ soluções parciais. Na impossibilidade de avaliar todos as soluções possíveis, cada nó expandido é avaliado com o preenchimento do espaço remanescente pela heurística gulosa e apenas um determinado número de nós com melhor avaliação são selecionados para serem expandidos, caracterizando uma busca em árvore com largura limitada. O método de busca em árvore utilizado é denominado Método Piloto (*Pilot Method*), utilizado com sucesso por Duin e Voss (1999) para resolver o problema de Steiner em grafos. Para evitar simetrias, soluções com mesma profundidade na árvore, mesmo valor de função objetivo e mesmo número de itens de cada tipo são consideradas iguais, sendo uma das soluções escolhida aleatoriamente para ser descartada. Comparações com outras abordagens utilizando conjuntos de instâncias testes da literatura comprovam o bom desempenho do algoritmo.

Lim et al. (2003) apresentam uma nova abordagem para o problema de carregamento de contêiner. São consideradas restrições de sobreposição e de orientação, sendo o objetivo maximizar o espaço ocupado. A heurística proposta utiliza uma técnica, denominada *multi-faced buildup,* na qual qualquer uma das seis faces do contêiner pode ser tratada como base ou piso do contêiner. As caixas são ordenadas de

acordo com um critério que considera o volume e a área base da orientação escolhida. Todos os espaços vazios disponíveis em todas as faces bases são avaliados e as caixas, seguindo a ordem estipulada, são carregadas no espaço vazio com melhor avaliação. Uma busca em árvore com largura limitada é utilizada para melhorar o desempenho da heurística, na qual k nós, no máximo, são avaliados em cada nível. Os autores utilizam as instâncias propostas por Bischoff e Ratcliff (1995a) para os testes computacionais. O algoritmo guloso consegue 84,3% em média de espaço ocupado, enquanto o algoritmo com busca em árvore consegue 87,8%. Eley (2002) obtém melhor desempenho, com 88,75% de volume ocupado do contêiner.

Gehring e Bortfeldt (1997) apresentam um algoritmo genético para o problema de carregamento de contêiner que, inicialmente, gera um conjunto de pilhas de caixa e depois posiciona as pilhas no piso do contêiner. São consideradas restrições de orientação, empilhamento, estabilidade vertical, orientação, limite e distribuição de peso. O procedimento que gera as pilhas de caixas procura minimizar os espaços não ocupados do prisma definido pela caixa base, enquanto o procedimento que posiciona as pilhas no piso do contêiner procura maximizar a área utilizada. Deste modo, o problema é decomposto em dois subproblemas, um problema de corte e empacotamento tridimensional e outro bidimensional. O cromossomo é representado por uma permutação de pilhas de caixas que são utilizadas por um algoritmo construtivo que resolve o PCE bidimensional de cobertura do piso do contêiner. A Figura 2.13 exemplifica o processo de codificação e decodificação dos cromossomos, em que uma seqüência de pilhas de caixas (codificação) é utilizada por uma heurística construtiva (decodificação) para gerar uma solução. Pode-se dizer que o algoritmo genético evolui a seqüência com que a heurística construtiva escolhe as pilhas de caixas. O algoritmo genético é hibridizado com procedimentos que especializam a decodificação dos cromossomos e a geração da população inicial de acordo com características do problema. Os testes computacionais revelam que a hibridização produz uma melhora em torno de 5% no valor da função objetivo, sem aumento significativo nos tempos computacionais. Os autores apresentam um algoritmo para gerar instâncias aleatórias com as restrições consideradas, no entanto, o gerador de números aleatórios e as sementes utilizadas não são descritos, prejudicando a reprodutibilidade dos dados. Adicionalmente, as instâncias propostas por Loh e Nee (1992) e Bischoff e Ratcliff (1995a) também são utilizadas para os testes computacionais.

O algoritmo genético proposto apresenta um bom desempenho, com volume ocupado de até 7,17% superior em relação às abordagens de Bischoff et al. (1995) e Bischoff e Ratcliff (1995) quando tomadas em separado. Os autores observam que o algoritmo é especialmente projetado para problemas fortemente heterogêneos, uma vez que, neste caso, as lacunas entre as pilhas de caixas não interferem decisivamente no aproveitamento do espaço. Um melhoramento deste algoritmo genético que utiliza, camadas verticais, é apresentado em Bortfeldt e Gehring (2001), e uma versão paralela deste último em Gehring e Bortfeldt (2002). Nestes últimos trabalhos, instâncias fortemente heterogêneas com até 100 tipos de caixas são geradas conforme os procedimentos descritos em Bischoff e Ratcliff (1995a). Considerando as 700 instâncias propostas por Bischoff e Ratcliff e somente restrições de estabilidade e sobreposição, os três algoritmos apresentam, na média, 87,51%, 90,06% e 90,43% de volume utilizado, respectivamente.

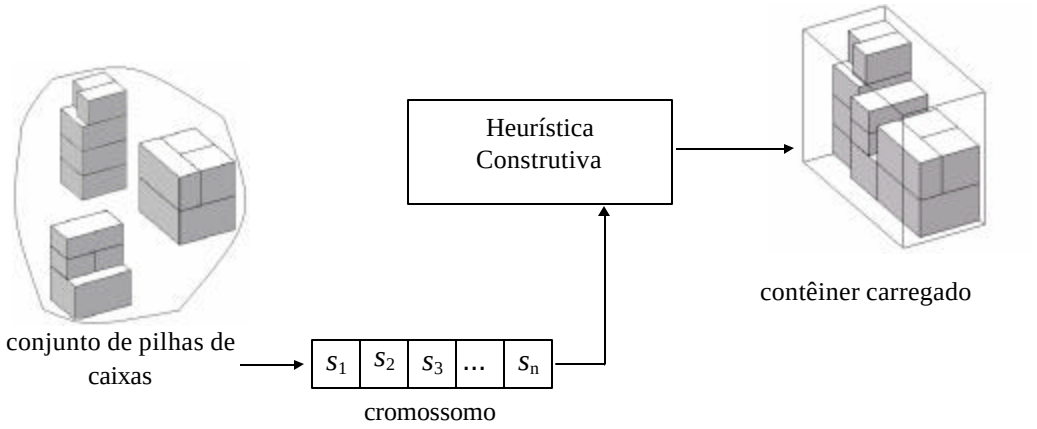


Figura 2.13. Representação do processo de codificação e decodificação dos cromossomos

Bortfeldt e Gehring (1998) propõem um algoritmo de busca tabu para o problema de carregamento de contêiner para o mesmo problema de carregamento contêiner tratado em Gehring e Bortfeldt (1997). O algoritmo de busca tabu supervisiona uma heurística construtiva que gera padrões de carregamento formados por arranjos que combinam até dois cubóides. A busca ocorre no espaço de soluções codificadas, guardando semelhança com o procedimento utilizado nos algoritmos genéticos descritos anteriormente. Para entender a representação da solução utilizada é importante considerar que em cada passo do processo de construção escolhe-se, de uma lista de candidatos, um cubóide para ser inserido na solução. Se uma solução é construída em p

passos, então as listas de candidatos podem ser representadas por $LC_i = \{\sigma_{i1}, \sigma_{i2}, ..., s_{in_i}\}$,

$i = 1, ...,p$. Denotando por s_i o índice do cubóide escolhido no passo i da heurística construtiva, a solução pode ser representada por uma seqüência $S = (s_1, s_2, \ldots, s_p)$, tal que, $s_{is_i} \in LC_i$. A Figura 2.14 demonstra graficamente esta estrutura, na qual a lista S apresenta os elementos escolhidos, da correspondente lista de candidatos, para compor a solução. Observe que o número de termos da seqüência S pode variar de uma solução para outra, como também o número de elementos da lista de candidatos pode variar de um passo para outro. O elemento escolhido para compor a solução, em cada lista de candidatos, aparece hachurado. Deste modo, existem três cubóides candidatos no primeiro passo, e o segundo cubóide é escolhido para ser incorporado à solução, ou seja, $n_1 = 3$ e $s_1 = 2$. No segundo passo, quatro cubóides estão disponíveis e o terceiro é escolhido, ou seja, $n_2 = 4$ e $s_2 = 3$. No passo p o cubóide com índice s_p é escolhido dentre os n_p candidatos possíveis. Se um critério guloso é utilizado para escolher os cubóides, tem-se $s_i = 1$ para $i = 1,..., p$.

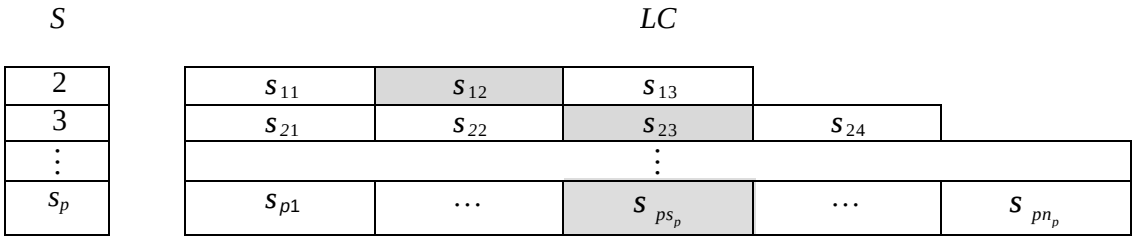


Figura 2.14. Representação de uma solução, Bortfeldt e Gehring (1998)

A vizinhança é definida através de alterações na seqüência S. Para qualquer seqüência de empacotamento S com p elementos, duas estruturas de vizinhanças são definidas conforme segue:

1. Vizinhança ampla: $S' \in N(S)$ se $s'_{i_0} \neq s_{i_0}$, $i_0 \in \{1, ...,p\}$, e $s'_i = s_i$ para todo $i = 1, ..., p, i \neq i_0$.
2. Vizinhança restrita: é definida analogamente à vizinhança ampla. Neste caso i_0 varia ciclicamente no decorrer das iterações, assumindo, para cada

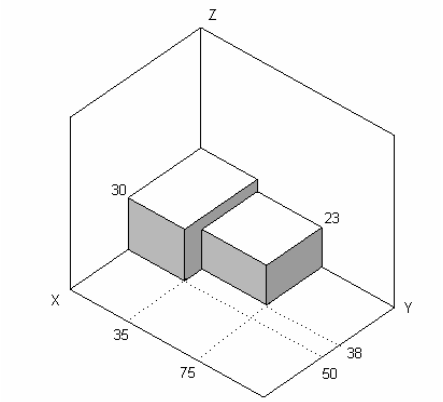
iteração, um valor no intervalo discreto $[1,\ldots, p]$. Inicialmente i_0=1 e quando i_0+1 > p a posição 1 é utilizada novamente.

A lista tabu é estendida a cada iteração para armazenar a seqüência de cubóides da solução corrente. Os autores argumentam que o manuseio da lista tabu não se torna computacionalmente caro porque poucas iterações são necessárias para o critério de parada utilizado. A diversificação é feita através do uso, em diferentes fases do algoritmo, de conjuntos de valores distintos para os parâmetros. Os testes computacionais consideram as 700 instâncias dividas em sete grupos propostas por Bischoff e Ratcliff (1995a). A média do volume utilizado para todas as instâncias é de 90,87% quando somente restrições de estabilidade e de sobreposição são consideradas. Uma versão paralela deste algoritmo é apresentada em Bortfeldt et al. (2003) que produz resultados com 91,6% de volume ocupado com restrições de estabilidade e sobreposição e 92,2% quando é permitida projeção da base de apoio máxima de 45%. Os autores concluem que o algoritmo de busca tabu proposto é mais adequado para instâncias fracamente heterogêneas, uma vez que a qualidade das soluções deteriora rapidamente com o aumento do número de tipos de caixas.

He e Cha (2002) desenvolvem um algoritmo genético para o problema de carregamento de contêiner ponderando três objetivos, a saber, maximização do volume ocupado, peso do contêiner e equilíbrio (centro de gravidade). Rodrigues Neto (2005) estende o trabalho de He e Cha (2002) utilizando uma função de aptidão adicional referente a valores monetários. Instâncias reais de empresas de eletrodomésticos são utilizadas nos testes computacionais. O peso utilizado para mensurar a importância de cada função de aptidão é determinado pelas empresas que fornecem as instâncias reais.

Segundo Bischoff (2006) a literatura referente a PCE3D tem avançado no sentido de propor métodos de resolução para problemas que incorporam fatores adicionais oriundos de situações reais, mas ainda existe um número expressivo de situações de relevante interesse prático que não foram devidamente tratadas. Uma destas situações é o caso em que o carregamento deve satisfazer restrições de empilhamento, para a qual o autor propõe um novo algoritmo. Bischoff interpreta as restrições de empilhamento de forma mais ampla do que em (Gehring e Bortfeldt, 1997), associando um peso w_t e um valor b_{tk} para a capacidade de empilhamento da face superior do tipo de caixa t com orientação k. Mais precisamente, b_{tk}, expresso em unidades de peso por área,

representa a pressão máxima que a face superior da caixa pode suportar. O autor considera que o peso é distribuído entre as faces superiores das caixas que proporcionam suporte, na proporção direta das áreas de contato envolvidas. Os pesos das caixas são proporcionais aos volumes e, portanto, todas as caixas possuem a mesma densidade. A estrutura de dados utilizada para armazenar uma solução é uma adaptação da representação espacial de Ngoi et al. (1994) que possui apenas duas matrizes bidimensionais, uma para descrever a posição das caixas e outra para armazenar a capacidade de empilhamento de cada espaço vazio. A Figura 2.15 apresenta a adaptação proposta para o carregamento da Figura 2.12. Os espaços são calculados de acordo com a proposta de Ngoi et al. (1994), com a diferença que a altura pode ser verificada diretamente nos elementos da matriz que correspondem à alocação da caixa. Deste modo, os elementos da matriz 1 com valor 30 correspondem à altura da caixa X_1 e com valor 23 a altura da caixa X_2. A matriz 2 é interpretada de forma semelhante, no entanto, os elementos da matriz informam a pressão máxima admitida na face superior das caixas. Observe que é atribuído um valor infinito para a pressão que o piso do contêiner pode suportar.



matriz 1

100	35	75	120
38	30	23	0
50	30	0	0
90	0	0	0

matriz 2

100	35	75	120
38	100	93	∞
50	100	∞	∞
90	∞	∞	∞

Figura 2.15. Adaptação da representação espacial de Ngoi et al. (1994)

A superfície de um espaço vazio pode ser composta por mais de uma caixa, de forma que uma caixa do tipo t só pode ser carregada em um espaço vazio i se for satisfeita a desigualdade

$$\frac{w_t}{d_{1tk} d_{2tk}} \leq B_{\min},$$

em que B_{min} determina a menor capacidade de empilhamento das superfícies que proporcionam apoio. O algoritmo carrega uma caixa por vez e avalia todas as combinações possíveis entre caixas e espaços vazios utilizando cinco critérios. Dado uma caixa do tipo t e um espaço vazio i, os critérios de avaliação são calculados conforme segue:

1. Relação entre o volume da caixa e a altura do espaço vazio.

   $C_1 = Z_i \, v_t$.

   Este critério vale-se da percepção que caixas volumosas devem ser carregadas próximas ao piso do contêiner, de forma que caixas menores possam ser colocadas em cima destas.

2. Adequação entre as dimensões da caixa e do espaço vazio.

   $C_2 = ad(X_i \, , d_{1kt}) + ad(Y_i \, , d_{2kt}) + ad(Z_i \, , d_{3kt})$, tal que

   $$ad\left(D, d\right)) = \begin{cases} 1 & \text{se} \quad D - d = 0 \\ 0{,}5 & \text{se} \quad d < 0{,}05 \cdot D \\ 0 & \text{caso contrário} \end{cases}$$

   em que D representa uma dimensão do espaço vazio e d a correspondente dimensão da caixa.

   Observe que a medida da adequação discrimina quando as dimensões da caixa e do espaço vazio são iguais ou muito próximas.

3. Número de espaços não utilizáveis gerados.

   As dimensões dos três espaços gerados se a caixa t com orientação k for carregada no espaço vazio i são: $d_{1tk} \times d_{2tk} \times \left(Z_i - d_{3tk}\right)$, $X_i \times \left(Y_i - d_{2tk}\right) \times Z_i$ e $\left(X_i - d_{1tk}\right) \times Y_i \times Z_i$. Se nenhuma caixa ainda não carregada couber em um dos espaços gerados, então o correspondente espaço é considerado não utilizável e descartado. O espaço referente à intersecção dos espaços frontal e lateral só é computado se os dois espaços, frontal e lateral, forem considerados não utilizáveis. O espaço superior também é considerado não utilizável se nenhuma caixa puder ser carregada por causa das restrições de empilhamento. Este critério

funciona como uma medida do espaço perdido originado da combinação caixa-espaço utilizada.

4. Potencial para construir pilhas de caixas idênticas.

$$C_4 = \min\left\{\hat{q}_t, \frac{Z_i}{d_{3tk}}, 1 + \left\lfloor \frac{B_{top} d_{1tk} d_{2tk}}{w_t} \right\rfloor\right\} \times \frac{v_t}{ve_i}$$

em que $B_{top} = \min\left\{b_{tk}, B_{\min} - \frac{w_t}{d_{1tk} d_{2tk}}\right\}$

Este critério fornece a capacidade potencial de construir uma pilha com caixas idênticas. O primeiro termo entre chaves representa o número de caixas do tipo t não carregadas, o segundo termo representa o número máximo de caixas que podem compor a pilha e o terceiro termo representa a medida equivalente do ponto de vista de capacidade de empilhamento. O multiplicador após o termo mínimo serve apenas para converter a medida de números de caixas para a fração do volume utilizado do espaço vazio. Em outras palavras, este critério é proporcional ao volume ocupado do espaço vazio i por uma pilha de caixas do mesmo tipo em uma dada orientação.

5. Perda relativa na capacidade de empilhamento.

$$C_5 = \begin{cases} \left(B_{avg} - B_{top} d_{1tk} d_{2tk}\right) \dfrac{Z_i - d_{3tk}}{d_{3tk}} & \text{se } \dfrac{B_{top}}{Z_i - d_{3tk}} < D_{\max} \\ 0 \text{ se } \dfrac{B_{top}}{Z_i - d_{3tk}} \geq D_{\max} \end{cases}$$

em que B_{avg} é o peso médio que pode ser empilhado sobre as caixas que compõem a superfície de suporte e $D_{\max}$ a maior densidade entre os tipos de caixas disponíveis para alocação. Este critério proporciona uma medida de degradação da capacidade de empilhamento das superfícies dos espaços vazios. Para tanto, é calculada a diferença entre a capacidade média de empilhamento do espaço vazio e a capacidade de empilhamento resultante da superfície superior da caixa candidata a ser carregada. Esta diferença é ajustada por um fator que

favorece a escolha de caixas com maior altura. O critério recebe valor nulo se a densidade atribuída ao espaço superior do carregamento for maior que $D_{\max}$.

A avaliação final E de uma combinação caixa-espaço vazio é dada por uma soma ponderada dos cinco critérios, com pesos negativos associados aos critérios C_3 e C_5, conforme segue:

$$E = \frac{v_1}{r_1}C_1 + \frac{v_2}{r_2}C_2 + \frac{v_3}{r_3}C_3 + \frac{v_4}{r_4}C_4 + \frac{v_5}{r_5}C_5$$

em que $v_1, ..., v_5$ são parâmetros definidos pelo usuário e $r_1, ..., r_5$ representam o intervalo entre o maior e menor valor dos correspondentes critérios das opções avaliadas. A combinação com maior soma é escolhida para ser adicionada à solução. Ao final, o autor propõe um algoritmo de múltiplos inícios que utiliza, a cada iteração, valores para os parâmetros $v_1, ..., v_5$ escolhidos aleatoriamente dentro do intervalo $(0,1)$ com distribuição uniforme. Adicionalmente, duas abordagens combinadas com o método simplex de Nelder e Mead (1965) também são propostas. Três conjuntos de testes computacionais são conduzidos para o algoritmo de múltiplos inícios utilizando as instâncias propostas por Bischoff e Ratcliff (1995a). O primeiro conjunto de testes demonstra que o algoritmo proposto apresenta desempenho similar ao algoritmo de Bortfeldt e Gehring (2001) quando utiliza as mesmas restrições de empilhamento definidas por estes últimos. O segundo conjunto de testes demonstra que o volume carregado é menor quando a capacidade de empilhamento das caixas diminui, como esperado. O terceiro conjunto de testes considera capacidade de empilhamento infinita, ou seja, somente restrições de sobreposição e estabilidade vertical são consideradas. Neste último caso, o algoritmo demonstra-se competitivo mesmo sendo especificamente projetado para tratar restrições de empilhamento. As abordagens combinadas com o método simplex de Nelder e Mead apresentam melhores resultados tanto para os testes que consideram capacidade de empilhamento como para os testes que consideram apenas restrições de sobreposição e estabilidade vertical. Em todos os testes o algoritmo apresentou melhor desempenho para instâncias fortemente heterogêneas.

Mack et al. (2004) propõem uma hibridização de busca tabu e *simulated annealing* para o problema de carregamento de contêiner, algoritmos seqüenciais e paralelos são apresentados. O padrão de carregamento é formado por arranjos que

consistem na combinação de até dois cubóides em três variantes pré-definidas. Para um dado espaço, todas as possíveis combinações de arranjo cubóides e espaço vazio são avaliadas de acordo com vários critérios. No algoritmo proposto, uma solução resultante do *simulated annealing* é pós-processada pela busca tabu. Segundo os autores, esta seqüência possibilita integrar as vantagens e evitar as deficiências de cada abordagem em separado. Resultados computacionais utilizando as 700 instâncias de Bischoff e Ratcliff (1995a) são analisados. A versão seqüencial produz soluções com um volume ocupado médio de 92,41%. Quando projeção da base é permitida, o resultado passa a ser 92,86% para a versão seqüencial e 93,2% para a versão paralela. Em termos de volume ocupado para este conjunto de instâncias com as referidas restrições, este trabalho apresenta os melhores resultados publicados na literatura até a atualidade.

Mais recentemente, Moura e Oliveira (2005) apresentam uma abordagem GRASP para o problema de carregamento de contêiner. O método usa como base o algoritmo de George e Robinson (1980) com vários aperfeiçoamentos que passa a ser denominado GRMod e a abordagem proposta GRModGRASP. O objetivo é a maximização do volume ocupado levando em consideração restrições de orientação e estabilidade. No GRModGRASP, a busca local aleatoriamente seleciona um ponto na seqüência de empacotamento e remove todas as caixas desta posição até o fim da seqüência. Durante a reconstrução, a partir da solução parcial, a caixa da posição selecionada é proibida de ser escolhida na primeira iteração e todas as decisões utilizam um critério guloso, sem aleatoriedade. O desempenho do algoritmo proposto é avaliado em termos de volume ocupado e estabilidade do carregamento. Duas formas para mensurar a estabilidade propostas por Bischoff e Ratcliff (1995a) são utilizadas. A primeira, denominada Medida 1, calcula a média das caixas que dão suporte a cada caixa posicionada acima do piso do contêiner (quanto maior melhor). A segunda medida, denominada Medida 2, calcula o percentual médio de caixas que não possuem pelo menos três faces laterais em contato com outras caixas (quanto menor melhor). Para os testes computacionais os autores utilizam as instâncias de Bischoff e Ratcliff (1995a) com até 100 tipos de caixas. Os testes comparativos incluem nove algoritmos propostos na literatura, sendo que nenhum destes apresenta sistematicamente taxa de volume ocupado maior que o GRModGRASP. No entanto, na média geral, GRModGRASP obtém o quarto melhor resultado. Com relação à estabilidade, o GRModGRASP apresenta baixo

desempenho para a Medida 2 devido ao uso de camadas verticais e desempenho comparável às demais abordagens para a Medida 1.

Lim e Zhang (2005) resolvem o problema de carregamento de contêiner com a metaheurística *squeaky wheel optimization* (SWO), proposta por Joslin e Clement (1999). Nesta abordagem, cada tipo de caixa carregada recebe um fator de prioridade dinamicamente atualizado através da análise da solução encontrada por uma heurística construtiva. As caixas identificadas como "complicadas" têm o fator de prioridade incrementado, e a heurística construtiva é aplicada novamente utilizando as caixas na ordem de prioridade modificada. O ciclo Construção – Análise – Priorização continua até um determinado critério de parada ser satisfeito. Os autores apresentam resultados para as instâncias de Bischoff e Ratcliff (1995a) com uma taxa de ocupação de volume de $91{,}81\%$. O tempo computacional para obter esse resultado é sensivelmente superior ao dos algoritmos utilizados para comparação.

Bortfeldt e Gehring (1999) propõem um algoritmo genético e uma busca tabu para resolver o problema *strip packing* 3D que são adaptações de algoritmos originalmente propostos para carregamento de contêiner (Gehring e Bortfeldt, 1997; Bortfeldt e Gehring 1998). Duas adaptações são apresentadas para cada abordagem. A primeira adaptação para a busca tabu consiste em determinar três comprimentos diferentes para o contêiner e escolher a melhor solução; a segunda adaptação consiste em iniciar o procedimento com um comprimento superestimado e diminuir o comprimento gradualmente no decorrer da busca. A primeira adaptação para o algoritmo genético modifica o procedimento que faz a varredura de espaços vazios, enquanto na segunda adaptação o comprimento é alterado de forma similar ao procedimento utilizado na busca tabu. Versões paralelas dos algoritmos também são apresentadas. Para os testes computacionais os autores utilizam o conjunto de instâncias de Loh e Nee (1992) e Bischoff e Ratcliff (1995a) com até 50 tipos de caixas. Os testes computacionais demonstram que a segunda adaptação é superior à primeira para ambos os algoritmos. Assim como nos trabalhos anteriores, a busca tabu apresenta melhores resultados para instâncias fracamente heterogêneas e o algoritmo genético para instâncias fortemente heterogêneas.

Bin Packing 3D

O problema *bin packing* 3D é de difícil resolução, motivo pelo qual este problema tem sido tratado com heurísticas mais elaboradas. Uma exceção a esta regra é o trabalho de Bischoff e Ratcliff (1995b) que aborda o problema correlato de carregamento de paletes do distribuidor, no qual caixas de diversos tipos são carregadas em paletes de diferentes dimensões. Os métodos propostos pelos autores correspondem a heurísticas construtivas que utilizam como base a heurística para carregamento de palete descrita em (Bischoff et al., 1995).

Ivancic et al. (1989) desenvolvem uma heurística baseada em programação inteira para resolver o problema de carregamento de múltiplos contêineres com o objetivo de minimizar o custo dos contêineres utilizados. O procedimento proposto utiliza a formulação clássica do problema de corte de estoque de Gilmore e Gomory (1961, 1963), conforme segue.

$$\min \sum_{h=1}^{nc} \sum_{j=1}^{s_h} c_h x_{jh}$$

s.a.

(MC) $$\sum_{h=1}^{nc} \sum_{j=1}^{s_h} A_{jh} x_{jh} \geq N$$

$$x_{jh} \geq 0 \text{ e inteiro },$$

em que :

- nc representa o número de tipos de contêineres,
- s_h número de padrões de carregamento disponíveis para um contêiner do tipo h ($h = 1,...,nc$),
- c_h custo do contêiner do tipo h ($h = 1,...,nc$),
- N vetor de número de cada tipo de caixa disponível,
- A_{jh} vetor de números de cada tipo de caixa empacotada no contêiner do tipo h ($h = 1,...,nc$) com o padrão de carregamento j ($j = 1,...,s_h$),
- x_{ij} número demandado de contêineres do tipo h ($h = 1,...,nc$) com o padrão de carregamento j ($j = 1, ...,s_h$).

Dado que gerar explicitamente todos os padrões de carregamentos possíveis é excessivamente caro computacionalmente, a proposta dos autores compreende gerar heuristicamente padrões de carregamento de "boa qualidade" e então resolver o problema MC. Esta abordagem também é utilizada por Eley (2003) em conjunto com o algoritmo de busca em árvore para carregamento de contêiner proposto anteriormente em (Eley, 2002).

Corcoran e Wainwright (1992) propõem um algoritmo genético para o problema *bin packing* 3D. O cromossomo é representado por uma permutação da seqüência de caixas para carregamento e, deste modo, operadores de cruzamento (*crossover*) e mutação utilizados em problemas de permutação podem ser aplicados. Uma heurística construtiva é utilizada para decodificar e avaliar os cromossomos. O modelo utilizado considera restrições de sobreposição e orientação. Para os testes computacionais são utilizadas instâncias com 50 e 500 itens, cujas dimensões variam de uma unidade até a metade de uma das dimensões do contêiner. As dimensões dos contêineres não são incluídas na análise, no entanto, os autores reportam resultados com 39,1% a 67,2% de aproveitamento do espaço utilizado nos *bins*. Um algoritmo aleatório utilizado para fins de comparação obtém resultados sistematicamente piores.

Variações do *bin packing* 2D e *bin packing* 3D são abordadas por Verweij (1996). Nestes problemas a carga é fracionada segundo a demanda de diversos clientes, e o carregamento deve ser efetuado de forma que seja possível movimentar somente as caixas do destino corrente por ocasião da entrega. Deste modo, as últimas demandas a serem entregues devem ser carregadas primeiro. A Figura 2.16 apresenta um contêiner carregado com quatro demandas identificadas por cores distintas. O autor apresenta em detalhes as estruturas de dados e procedimentos utilizados para a representação da solução. Como método de solução é proposta uma heurística que pode ser interpretado como uma variação da heurística de busca em árvore com largura limitada. Resultados são apresentados considerando as demandas dadas em uma ordem pré-estabelecida. Como esperado, quanto maior o número de demandas menor é o aproveitamento do espaço do contêiner.

Figura 2.16. Contêiner carregado com carga fracionada em três demandas

Eley (2002) implementa duas estratégias para tratar o problema *bin packing* 3D utilizando o algoritmo para carregamento de contêiner apresentado na subseção anterior. A primeira estratégia, denominada estratégia seqüencial, carrega um contêiner após o outro até que todas as caixas tenham sido utilizadas, enquanto a segunda estratégia, denominada estratégia simultânea, carrega um determinado número de contêineres ao mesmo tempo. Na estratégia simultânea o número inicial de contêineres é determinado pelo menor inteiro maior que o quociente da divisão da soma do volume das caixas disponíveis pelo volume de um contêiner, ou seja,

$$\left\lceil \sum_{t=1}^{m} v_t \Big/ V \right\rceil .$$

Se não for possível carregar todas as caixas o número de contêineres é aumentado. Os experimentos computacionais consideram restrições práticas além das restrições de sobreposição de caixas. Comparações com outras abordagens utilizando conjuntos de instâncias testes da literatura comprovam o bom desempenho dos algoritmos. Eley (2003) estende estes procedimentos dentro de uma abordagem baseada em separação de conjuntos para tratar o problema de carregamento de múltiplos contêineres. Restrições adicionais de separação e carregamento completo de grupos de itens são consideradas.

Lodi et al. (2002) resolvem o problema *bin packing* 3D utilizando padrões de carregamento formados por camadas. A primeira camada coincide com o piso de um *bin* e os itens são empacotados com suas bases sobre ele. Os pisos das camadas subseqüentes são definidos pelo item mais alto da camada imediatamente inferior. Obviamente este procedimento não satisfaz restrições de estabilidade vertical. Um algoritmo de busca tabu, denominado TSpack, é proposto para resolver o problema, tendo como base o trabalho de Lodi et al. (1999). Este algoritmo possui como principal

característica uma estrutura de vizinhança parametrizável que varia de tamanho dinamicamente durante a busca e é independente das especificidades do problema considerado. Um movimento, em um dado tamanho de vizinhança, consiste em resolver um problema *bin packing* 3D com uma caixa oriunda de um bin, denominado *target bin*, mais as caixas de outros k *bins*, sendo k um parâmetro que determina o tamanho da vizinhança. Deste modo, os movimentos objetivam esvaziar o *target bin*. O parâmetro k é decrementado de uma unidade sempre que um movimento diminui o número de *bins* da solução e é incrementado se nenhum movimento factível é encontrado. O procedimento de carregamento é efetuado por uma heurística determinística, denominada heurística A, que tem por função produzir soluções factíveis. A lista tabu armazena os valores da função de avaliação utilizada para escolher o *target bin*, e cada vizinhança (diferentes valores de k) tem sua própria lista tabu. O algoritmo *guided local search* de Fareo et al. (1999), que considera restrições de estabilidade vertical, é utilizado para comparação de desempenho. Os resultados computacionais, com instâncias geradas pelos autores, mostram que as duas abordagens possuem características complementares, cada uma delas apresentando melhor desempenho em subgrupos de instâncias diferentes. O algoritmo busca tabu apresenta desempenho melhor em instâncias heterogêneas, muito provavelmente por que não satisfaz restrições de estabilidade vertical.

Faroe et al. (2003) propõem um algoritmo *guided local search* para resolver o problema de *bin packing* 3D tratado por Martello et al. (2000). *Guided local search* utiliza memória para encontrar regiões promissoras do espaço de soluções adicionando à função objetivo um termo que penaliza características “ruins” de soluções previamente visitadas. Em linhas gerais, o algoritmo inicia com uma solução factível obtida com uma heurística construtiva gulosa e iterativamente diminui um *bin* sempre que uma nova solução factível é encontrada. Se o número de *bins* encontrado igualar-se ao limitante inferior previamente calculado, então a solução ótima foi encontrada. Durante a busca, a restrição de sobreposição de caixas é relaxada e a única restrição imposta é que as caixas devem ser posicionadas integralmente dentro de um contêiner. Deste modo, soluções infactíveis, com sobreposição de caixas, são visitadas e o objetivo passa a ser minimizar o espaço de intersecção das caixas carregadas. Um valor de função objetivo zero indica um empacotamento factível. A vizinhança é definida por movimentos que transladam uma caixa para a mesma posição dentro de um contêiner diferente ou para

uma posição dentro do mesmo contêiner ao longo de uma das três dimensões. O espaço de intersecção de cada par de caixas pertencentes a uma determinada solução é penalizado no cálculo do valor da função objetivo. Os autores concluem que a contribuição mais importante do trabalho é apresentar um algoritmo que aplica busca local diretamente sobre a representação da solução do *bin packing* 3D. No entanto, deve-se observar que a restrição de orientação fixa das caixas facilita a definição da vizinhança. As instâncias utilizadas nos testes computacionais são geradas de acordo com Martello et al. (2000) com 50, 100, 150 e 200 caixas.O algoritmo proposto foi comparado com o algoritmo exato de Martello et al. (2000) com limitação de tempo. O algoritmo g*uided local search* encontra soluções similares ou melhores que o exato em todos os casos.

Silva e Soma (2003) desenvolvem uma heurística construtiva gulosa para resolver o problema *bin packing* 3D com a restrição de produzir padrões de carregamento estaticamente estáveis. Neste tipo de problema, um arranjo de caixas é considerado estável se a resultante dos momentos em relação ao centro de gravidade é nula. Os autores encontram dificuldades em estender o estudo para incorporar restrições de estabilidade dinâmica devido à existência de muitos fatores externos de difícil mensuração.

Lim e Zhang (2005) resolvem o problema de bin packing 3D aplicando a metaheurística SWO apresentada na subseção anterior para o problema de carregamento de contêiner com estratégia sequencial, ou seja, os contêineres são preenchidos um após o outro até que todas as caixas sejam empacotadas. Os resultados computacionais reportam 21 soluções ótimas e um total de 694 *bins* para resolver todas as 47 instâncias propostas por Ivancic et al. (1989). Embora os autores reivindiquem serem os primeiros a resolver este conjunto de instâncias com menos de 700 *bins,* Eley (2003) apresenta resultados com 699 *bins* e prova a otimalidade de 25 das 47 instâncias.

Carregamento e roteamento de veículos

Em problemas de roteamento de veículos o objetivo é minimizar o custo das rotas de uma frota de veículos que satisfaçam as demandas dadas pelos clientes e um conjunto de restrições. Usualmente, as demandas são expressas por um número inteiro positivo, que representa o peso ou volume a ser despachado. Isto pode levar a fortes simplificações para muitas aplicações práticas de roteamento de veículos, em que as

demandas são compostas por itens discretos com um determinado formato e cujo carregamento passa a ser um subproblema com restrições adicionais.

O campo de pesquisa para problemas de roteamento com modelos mais realísticos é de particular interesse devido à relevância prática e à difícil resolução. O problema de carregamento e roteamento de veículos, no qual as demandas são compostas por caixas e leva-se em consideração as restrições de empacotamento decorrentes desta suposição, é uma das muitas extensões que podem ser estudadas.

Existem poucos trabalhos na literatura cientifica que tratam o problema de carregamento de roteamento de veículo. Em sua tese de doutorado, Iori (2004) considera o caso especial do roteamento de veículos com restrições de empacotamento bidimensionais. Neste problema, existe uma frota de veículos idênticos com capacidade conhecida que devem servir a um determinado número de clientes. A demanda de cada cliente consiste em um conjunto de peças bidimensionais retangulares com um determinado peso. Os veículos possuem uma superfície bidimensional para carregar as demandas e uma capacidade máxima de peso. O objetivo é encontrar uma partição de clientes em rotas de custo total mínimo de tal forma que, para cada veículo, o peso do carregamento não excede um máximo permitido e a alocação dos itens fica sujeito às seguintes restrições:

Agrupamento de itens : todos os itens de um dado cliente devem ser carregados no mesmo veículo.

Orientação : os itens têm orientação fixa (não podem ser rotados) e devem ser carregados com os lados paralelos aos lados da superfície de carregamento dos veículos.

Múltiplos destinos : operações de descarga na posição de um dado consumidor são executadas a partir de um único lado do veículo. Quando os itens de um determinado destino são entregues, nenhum item pertencente a um outro consumidor deve ser movimentado.

O autor propõe um algoritmo *branch-and-cut* e um algoritmo de busca tabu para a resolução do problema. O algoritmo *branch-and-cut* minimiza o custo das rotas e utiliza duas famílias de restrições para introduzir cortes. A primeira família de cortes leva em conta a capacidade dos veículos e a segunda proíbe padrões de carregamento infactíveis. Métodos de separação heurísticos são utilizados para detectar violação de restrições de

ambas as famílias. Um algoritmo *branch-and-bound* é iterativamente invocado para verificar a factibilidade dos carregamentos. As instâncias utilizadas nos testes computacionais são derivadas de instâncias para o problema clássico de roteamento de veículos capacitados (Toth e Vigo, 2002). Ao todo são utilizadas 60 instâncias com o máximo de 35 clientes e 114 itens. Considerando 24 horas como tempo computacional máximo em um Pentium 4 1.7 GHz, o algoritmo encontra a solução ótima para todas as instâncias, com exceção de cinco casos. Além disso, todas as instâncias com até 25 clientes são resolvidas em menos de uma hora de processamento.

No algoritmo de busca tabu todas as rotas satisfazem a restrição de carregamento seqüencial, no entanto, podem ser infactíveis no que se refere ao peso máximo permitido e o comprimento máximo da superfície de carregamento. O algoritmo nunca produz carregamentos que excedem a largura da superfície de carregamento ou que impliquem em sobreposição de itens. Movimentos infactíveis são penalizados proporcionalmente ao nível de violações. Desta forma, ao executar um movimento, o algoritmo deve considerar a melhora na solução corrente em termos de distância total e factibilidade da rota. Para o roteamento, o autor adota uma abordagem derivada de algumas técnicas consagradas no Taburoute (Gendreau et al., 1994). No que concerne ao carregamento, a restrição de peso é trivialmente verificada, mas as restrições de carregamento bidimensional requerem resolução heurística. Para as 57 instâncias em que o ótimo é conhecido, a busca tabu encontra 33 soluções ótimas. Para instâncias em que o algoritmo *branch-and-cut* foi interrompido antes de provar a otimalidade, a busca tabu encontra soluções factíveis que são em média 0,21% melhores. A busca tabu também foi aplicada em um conjunto de instâncias para o qual a solução ótima não é conhecida. O algoritmo foi capaz de encontrar soluções factíveis em todos os casos. O autor evidencia o fato que introduzir restrições de carregamento bidimensional aumenta consideravelmente o custo das soluções.

Gendreau et al. (2006a) consideram o problema de roteamento de veículos capacitados com restrições de empacotamento tridimensional em que o número de veículos é fixo. Um algoritmo de busca tabu trata o roteamento dos veículos e iterativamente invoca um segundo algoritmo de busca tabu para gerar os padrões de carregamento tridimensional. As demandas são compostas por caixas de diferentes tamanhos e devem ser empacotadas obedecendo as seguintes restrições:

Orientação : A dimensão que determina a altura é fixa, no entanto, a base pode girar 90^{o}.

Empilhamento : As caixas são classificadas em frágeis e não frágeis. Caixas frágeis e não frágeis podem ser empilhadas, mas uma caixa não frágil nunca pode ser colocada sobre uma caixa frágil.

Estabilidade : As caixas devem ser carregadas com não menos que 75% de suporte da área da base.

Múltiplos destinos : as caixas devem ser carregadas de modo que quando os itens de um determinado destino são descarregados, nenhum item pertencente a um outro cliente precisa ser movimentado.

A Figura 2.17 apresenta um carregamento factível, no qual existem três demandas e uma das caixas apresenta projeção da base de apoio. Não é difícil verificar, através das cores distintas de cada demanda, que as restrições de múltiplos destinos são satisfeitas.

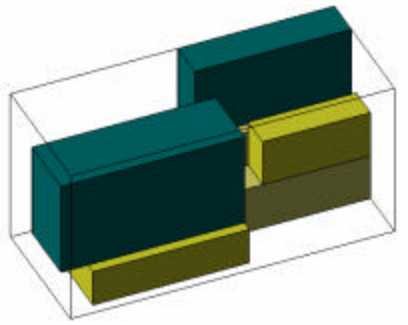


Figura 2.17. Carregamento factível para o problema de roteamento de veículos capacitados com restrições de empacotamento tridimensional

O algoritmo de busca tabu que gera os padrões de carregamento tridimensional explora uma vizinhança que modifica a seqüência com que as caixas das demandas são carregadas nos veículos. Dada tal seqüência, duas heurísticas construtivas são executadas para encontrar uma solução factível que minimize o comprimento utilizado, o algoritmo posição anterior à esquerda (BLA - *bottom left algorithm*) e contato perimetral (TPA - *touching perimeter algorithm*).

O BLA original foi proposto por Baker et al. (1980) para empacotar itens bidimensionais, um de cada vez, tal que o item a ser carregado é designado para a posição mais anterior à esquerda possível da área de carregamento. A adaptação para o caso tridimensional consiste em designar itens para posições em que a base dos mesmos fique apoiada no piso do container ou no topo de itens previamente carregados. Da mesma forma, a face lateral esquerda e a face anterior dos itens carregados devem ficar em contado com uma face do contêiner ou de outros itens. As posições que possuem estas características são denominadas *normais* e são identificadas seqüencialmente de acordo com o menor valor da coordenada y com critério de desempate de menor valor da coordenada x e então o menor valor de z. Para cada posição, rotações de 90^o dos itens são avaliadas. O primeiro item carregado que satisfazer as restrições de empacotamento é selecionado. O algoritmo termina quando todos os itens são empacotados ou o comprimento do carregamento exceder o dobro do comprimento L do contêiner, situação na qual é retornada uma solução com valor de avaliação $2.L$.

O TPA original foi proposto por Lodi, Martello e Vigo (1999) para empacotar itens bidimensionais, um de cada vez, em uma posição normal que proporciona a maior porcentagem do perímetro do item em contado com as faces da área de carregamento ou outros itens empacotados. Na adaptação para o caso tridimensional é calculada a porcentagem das superfícies de contato do item a ser carregado com as faces do contêiner ou de outros itens já empacotados. A posição que proporcionar a maior porcentagem de contato é selecionada. O critério de parada é igual ao do algoritmo anterior.

Depois de executar uma das heurísticas construtivas de empacotamento descritas acima, os itens são classificados em dois tipos: itens completamente empacotados dentro dos veículos são do tipo 1, enquanto itens com uma porção não empacotada no comprimento são do tipo 2. A vizinhança é então explorada considerando todos os possíveis movimentos em que um item do tipo 1 é permutado por um item do tipo 2 na seqüência de empacotamento corrente. Para cada nova seqüência, executa-se as heurísticas construtivas e atribui-se um valor conforme exposto a seguir. Seja $S(i,j)$ o valor associado à solução retornada pela heurística construtiva quando o item i é permutado, na seqüência de carregamento, pelo item j e $j\,(x)$ a razão entre o número de vezes que o item x foi selecionado para ser permutado nos movimentos anteriores e o

número de movimentos executados até então. A avaliação de uma seqüência é dada pela expressão $S'(i,j) = S(i,j) + (\varphi(i) + \varphi(j)).L$. O segundo termo da expressão penaliza as permutações mais freqüentes de cada item considerado.

O algoritmo utiliza duas listas tabu (memória de curto prazo), uma para tipo 1 e outra para tipo 2. Um movimento de permutação entre os itens i e j é considerado tabu se i está em uma lista e j na outra. No entanto, um movimento tabu é executado se vier a melhorar a avaliação da solução incumbente (critério de aspiração). A busca tabu termina quando um carregamento factível é obtido ou após um número pré-determinado de iterações.

O algoritmo de busca tabu desenvolvido pelos autores para tratar o roteamento dos veículos faz uso de alguns procedimentos para roteamento de veículos capacitados utilizados em (Gendreau, Hertz e Laporte, 1994) e para roteamento de veículos capacitados com restrições de empacotamento bidimensionais utilizados em (Gendreau et al., 2006b). Movimentos que levem a soluções infactíveis são permitidos, em que algum veículo tem a capacidade de peso violada ou o comprimento do carregamento excede o comprimento L do contêiner.

Uma solução inicial é obtida a partir de um ou dois algoritmos, um para grafos genéricos e outro para grafos euclidianos. Esses dois algoritmos foram adaptados para o problema em questão, a saber, algoritmo de economias de Clarke e Wright (1964) para roteamento de veículos capacitados, denominado HGEN, e o algoritmo para roteamento periódico de veículos com múltiplos depósitos de Cordeau et al. (1997), denominado HEUCL. O algoritmo HGEN começa com um cliente por rota e iterativamente une duas rotas para minimizar a distância total, desde que a rota resultante seja factível quanto ao empacotamento tridimensional. No entanto, para satisfazer a restrição de número máximo de veículos são permitidas uniões de rotas que levam a soluções infactíveis quanto ao peso ou comprimento do carregamento. Da mesma forma, o algoritmo HEUCL é baseado em designações factíveis sucessivas de clientes para rotas, exceto para a última rota na qual designações com empacotamento tridimensional infactível são aceitas. Quando ambos os algoritmos são executados, a solução com o menor número de veículos infactíveis é selecionada.

Após obter uma solução inicial, uma vizinhança em que um cliente é retirado de uma rota e designado à outra é explorada. Ambas as rotas evolvidas no movimento são re-otimizadas através da inserção generalizada 4-opt GENI (Gendreau et al., 1992). De modo a reduzir o custo computacional, é implementada uma restrição de vizinhança. Dado um cliente i, assuma que os correspondentes custos dos arcos c_{ij} ($j = 1,...,n$) são ordenados de forma não crescente. Define-se como candidatos de i os p clientes correspondentes aos p primeiros custos (p um parâmetro do algoritmo). A cada iteração, um cliente i só pode ser designado para outra rota desde que esta contenha ao menos um dos p candidatos de i. Após executar um movimento, a re-inserção de um cliente na rota original é considerada tabu por q iterações.

As soluções geradas são avaliadas conforme segue:

$$\text{avaliação} = (\text{distância total}) + a\,(\text{excesso de peso}) + b\,(\text{excesso de comprimento}) + g \cdot f(i, j)$$

em que $f(i, j)$ denota a razão entre o número de vezes que um movimento designa o cliente i para o veículo j e o número de movimentos executados. O segundo e o terceiro termo da expressão de avaliação penalizam infactibilidades, enquanto o quarto termo promove uma diversificação na busca. Para calcular o excesso de comprimento a busca tabu que resolve o roteamento invoca iterativamente a busca tabu que gera os padrões de carregamento tridimensional.

Os valores dos parâmetros a e b são ajustados durante o processo de busca, tomando como base a variação da infactibilidade total do peso e comprimento do carregamento, respectivamente. Se a variação da infactibilidade for positiva o respectivo parâmetro é incrementado em 10% e decrementado em 10% se a variação for negativa.

Dois tipos de intensificação da busca são utilizados. O número de iterações do algoritmo de empacotamento tridimensional é dobrado sempre que a solução corrente tem uma distância total menor que a solução incumbente, não existe excesso de peso em nenhuma das rotas e alguma rota tem excesso no comprimento do carregamento. O valor do parâmetro p é dobrado na iteração seguinte quando a solução incumbente é atualizada.

Os testes computacionais consideram vinte e sete instâncias derivadas do problema de roteamento de veículos capacitados e cinco instâncias reais de fábricas moveleiras italianas com distâncias aproximadas por retas no plano (euclidianas). As

instâncias oriundas da literatura possuem no mínimo 15 e no máximo 100 clientes, com um total de caixas variando de 26 a 199. Por sua vez, as instâncias reais apresentam 44 a 64 clientes e um total de caixas variando de 141 a 181. O número de veículos é determinado de modo a garantir a existência de soluções factíveis. Duas versões do algoritmo proposto são consideradas, com um ou múltiplos inícios. Os experimentos não revelam dominância entre as versões apresentadas. As restrições de empacotamento são consideradas todas em conjunto e em separado. Quando as restrições de empilhamento são desconsideradas a distância total diminui, na média, 2,66% e quando as restrições de múltiplos destinos são desconsideradas a distância total diminui 8,74%. Se somente restrições de sobreposição são consideradas, ganha-se 15,87% na distância total em média.

Capítulo 3

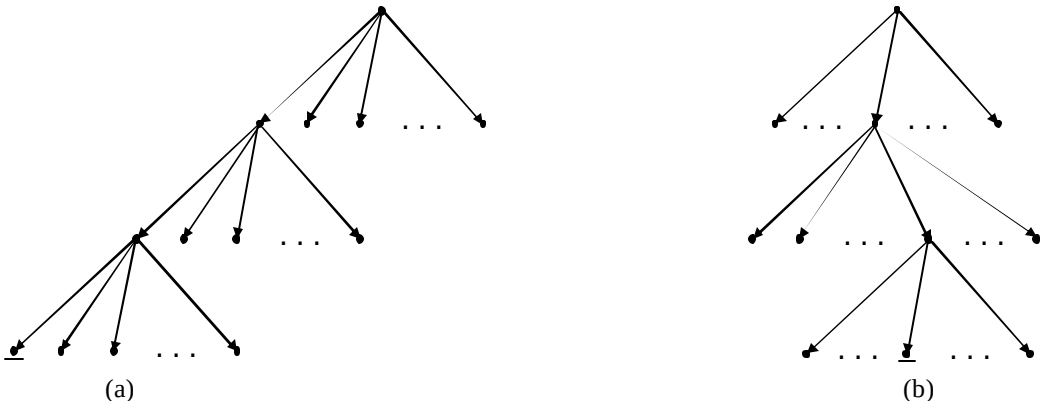
Problema de Carregamento de Contêiner

3.1 Introdução

Heurísticas construtivas surgem naturalmente como uma das primeiras estratégias para resolver problemas combinatórios. Neste tipo de heurística, inicia-se com uma solução vazia e, sucessivamente, elementos são adicionados à solução até que esta seja considerada completa. A natureza dos elementos depende das especificações do problema, por exemplo, para o problema de carregamento de contêiner a escolha de uma caixa e da respectiva posição para carregá-la corresponde a uma decisão de qual elemento adicionar à solução. Este processo pode ser adequadamente representado através de uma árvore de decisões construtivas, na qual o nó raiz corresponde à solução vazia e as folhas correspondem a soluções completas ou parciais infactíveis. Uma solução parcial corresponde a um nó do caminho entre o nó raiz e um nó folha da árvore construtiva.

Se a solução é construída escolhendo-se sempre a melhor opção em cada nó de decisão, e estas decisões não são reconsideradas, a heurística é denominada construtiva gulosa. A Figura 3.1 (a) representa a árvore construtiva associada a uma heurística gulosa que gera uma solução em três passos. Em cada nó várias opções são ordenadas com base em uma avaliação de mérito, em geral a função objetivo. A representação adotada sempre considera que a melhor opção está à esquerda do leitor e, seguindo a ordem estipulada, a pior está à direita. Os nós assinalados com um traço abaixo correspondem a nós folhas. Por outro lado, se as decisões são tomadas, por

exemplo, aleatoriamente, qualquer opção pode ser escolhida a partir de um nó de decisão, conforme representado na Figura 3.1 (b).



(a) (b)

Figura 3.1. Representações de árvores construtivas

Como escolhas gulosas são freqüentemente míopes, por serem decisões locais sem análise de impacto futuro, essas heurísticas podem fornecer soluções razoáveis, mas não necessariamente ótimas ou de boa qualidade. Na maior parte dos casos é possível melhorar a qualidade se uma busca local é aplicada a uma solução completa ou parcial. A busca local produz alterações através de movimentos definidos por uma vizinhança, passando de uma solução completa para outra. Um movimento é uma modificação elementar que altera uma solução. O conjunto de soluções que pode ser obtido desta forma é chamado de vizinhança da solução. Se nenhum movimento melhorar a qualidade, a busca é finalizada e obtém-se um ótimo local.

Dentro deste contexto surge o método de múltiplos inícios, conforme pseudocódigo apresentado na Figura 3.2. Cada iteração consiste de uma fase de construção, seguida por uma fase de busca local, que inicia com a solução construída na fase anterior. Este método possui diversas variações e pode ser especializado para uma grande variedade de problemas.

A Figura 3.3 ilustra a fase construtiva com um pseudo-código. Para cada iteração desta fase, considere o conjunto dos elementos que podem ser incorporados à solução parcial. A seleção do elemento a ser incorporado à solução é determinada pela avaliação dos candidatos, usualmente de acordo com a contribuição que cada um proporciona na função objetivo, ver linha 3. A avaliação dos elementos permite a criação

de uma lista de candidatos ordenada. Um elemento da lista de candidatos é selecionado e incorporado à solução parcial, ver linhas 4 e 5. O procedimento termina quando uma solução completa é obtida.

```
    Procedimento múltiplos inícios
1   Leia e processe os dados da instância e parâmetros do algoritmo
2   iter ← 0
3   Enquanto (critério de parada não satisfeito) faça
4       Fase 1 : (Construção)
5               construa a solução S
6       Fase 2 : (Busca Local)
7               Aplique um método de busca para melhorar S
8               Seja S' a solução obtida
9
10      Se S' é melhor do que a melhor solução obtida então
11          Atualize a melhor solução
12      Fim_Se
13      iter ← iter + 1
14  Fim_Enquanto
```

Figura 3.2. Pseudocódigo para métodos de múltiplos inícios

```
    Procedimento fase construtiva
1   S ← {}
2   Enquanto solução não completa faça
3       Construa a lista de candidatos LC
4       s ← Selecione um elemento da LC
5       S ← S ∪ {s}
6   Fim_Enquanto
```

Figura 3.3. Pseudocódigo da fase construtiva

Martí (2003) classifica os métodos de múltiplos inícios de acordo com três elementos, a saber, memória, aleatoriedade e grau de reconstrução, que podem estar presentes de várias formas na fase de construção. É importante observar que estes elementos não são mutuamente excludentes e não estão restritos aos dois extremos

"presente" e "ausente", ao contrário, podem assumir diferentes graus de intensidade nos intervalos memória/sem memória, sistemático/aleatório e reconstrução total/reconstrução parcial.

O elemento memória pode ser utilizado de forma semelhante às estratégias originalmente propostas por Glover (1986) para busca tabu. O uso de memória permite que decisões associadas a soluções de boa qualidade possam ser enfatizadas durante o processo construtivo, principalmente nos primeiros passos, quando a avaliação das decisões ainda é muito incipiente. De fato, dificilmente é dado saber se uma decisão avaliada como menos atrativa no estágio inicial da construção vai propiciar vantagens ao final do processo (Glover, 2000). Alguns autores utilizam um conjunto de soluções de boa qualidade, denominado conjunto elite, para explicitar características desejáveis e, deste modo, favorecer a construção de soluções que incorporem estas características (Fleurent e Glover, 1999; Binato et al. 2002; Ahmadi e Osman 2003; Armentano e Araújo, 2006). Outras formas de utilizar memória para construir soluções podem ser encontradas em Patterson et al. (1999) e Glover (2000). O uso de memória está vinculado à convicção que uma estratégia deve utilizar informações geradas no decorrer das iterações do algoritmo. Por outro lado, o uso de memória deve ser evitado se o propósito é obter uma amostragem não induzida do espaço de soluções.

Aleatoriedade é a forma mais simples de se obter uma amostragem não induzida do espaço de soluções, no entanto, é importante lembrar que a representação da solução e o método de construção podem por si só incluir um certo grau de tendenciosidade no algoritmo (Montgomery et al., 2004). Também é possível gerar soluções de forma sistemática, semelhante aos métodos utilizados em inteligência artificial para busca em árvore, que utilizam uma ordem pré-determinada para visitar os nós da árvore fazendo suposições sobre o custo dos nós folhas (Pearl, 1984; Nilsson, 1982).

O grau de reconstrução indica o número de elementos que são fixados de uma iteração para outra. Comumente os métodos propostos na literatura constroem soluções a partir do conjunto vazio, mas existe a possibilidade de estrategicamente fixar alguns elementos de forma a permitir a geração de soluções com determinadas características. Este procedimento emula estratégias baseadas em memória e é especialmente útil para identificar decisões tomadas no início do processo construtivo que

podem levar a soluções de boa qualidade. Existe ainda o caso em que a busca local é aplicada repetidas vezes durante a fase construtiva em uma solução parcial (Russel,1995; Chiang e Russel, 1996; Fleurent e Glover, 1999).

A metaheurística *greedy randomized adaptive search procedures* (GRASP) permite ilustrar como é possível combinar a geração sistemática e aleatória de soluções. Na fase construtiva, as possibilidades de escolha são ordenadas de melhor para pior em uma lista restrita de candidatos (LRC) de acordo com uma função de avaliação gulosa. Esta função mede, de forma míope, o benefício de selecionar cada elemento. A LRC é composta por um determinado número dos elementos candidatos com melhor avaliação (aspecto guloso e sistemático) e um elemento desta lista é escolhido aleatoriamente para ser adicionado à solução em construção (aspecto aleatório). A cada iteração da fase construtiva a LRC é atualizada (aspecto adaptativo). Observe que se a LRC contém todos os elementos candidatos então o método recai na geração aleatória de soluções, e se a LRC contém apenas o melhor elemento tem-se uma heurística construtiva gulosa. Uma forma de criar uma restrição implícita no número de candidatos pode ser obtida com o uso de funções de tendência (*bias*), conforme sugerido por Bresina (1996). As funções de tendência são baseadas na posição $pos(s)$ atribuída a cada elemento candidato s na LRC. Bresina sugere as seguintes funções de tendência:

- aleatório: $bias(pos) = 1$
- linear: $bias(pos) = \frac{1}{pos}$
- logarítmico: $bias(pos) = \log^{-1}(pos + 1)$
- exponencial: $bias(pos) = e^{-pos}$
- polinomial de ordem n: $bias(pos) = pos^{-n}$

A partir dos valores de uma dada função de tendência, a probabilidade $p(s)$ do elemento s ser selecionado é dada por:

$$p(s) = \frac{bias(pos(s))}{\sum_{s' \in LRC} bias(pos(s'))}$$

A Figura 3.4 apresenta graficamente as probabilidades calculadas, segundo diferentes funções de tendência, para uma LRC com cinco elementos. É possível observar que a função que promove maior discriminação entre os elementos é a polinomial com grau 4.

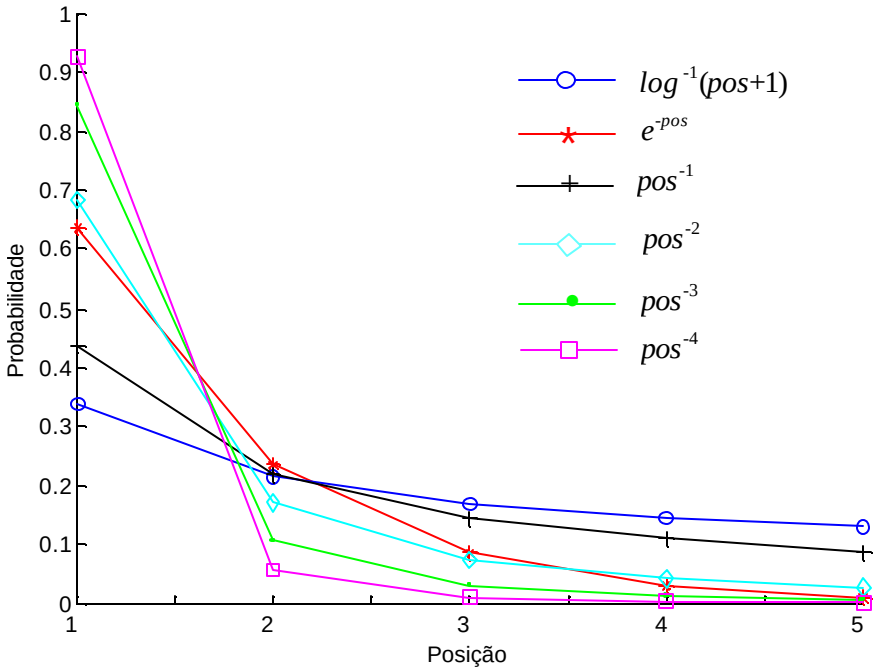


Figura 3.4. Gráfico das probabilidades, calculadas com diferentes funções de tendência, para uma LRC com cinco elementos

Tratando especificamente do problema de carregamento de contêiner, observe que a busca local do algoritmo GRASP proposto por Moura e Oliveira (2005) pode ser entendida como uma reconstrução parcial, segundo a definição apresentada anteriormente.

O exemplo a seguir auxilia a entender a dificuldade de projetar busca local para o problema de carregamento de contêiner com representação direta ou indireta da solução.

Exemplo 1. Considere um contêiner com dimensões $120 \times 100 \times 80$ em uma dada unidade de comprimento e o conjunto de caixas descrito na Tabela 3.1. O objetivo é

selecionar caixas do conjunto dado para serem carregadas com 100% de apoio da base de modo a maximizar o volume ocupado do contêiner.

Tabela 3.1. Conjunto de caixas do exemplo 1

Tipo	Dim. 1	Dim. 2	Dim. 3	Quant.
1	120	100	30	1
2	60	100	25	2
3	100	100	40	1
4	100	100	60	1

A Figura 3.5 (a) mostra uma solução com 68,75% de ocupação do volume do contêiner. O padrão de carregamento é formado por caixas do tipo 1 e 2 e não existe espaço vazio factível para carregar as caixas do tipo 3 e 4. Uma forma de tentar melhorar a qualidade da solução é trocar a caixa do tipo 1 pela caixa tipo 3, uma vez que esta última possui maior volume. Se a caixa tipo 3 for posicionada nas mesmas coordenadas da caixa que é substituída, a solução fica infactível no que se refere à sobreposição de volumes e estabilidade vertical, ver (b). Um novo padrão de carregamento deve ser calculado para que se obtenha uma nova solução factível. Este procedimento pode ser caro computacionalmente e não existe garantia que uma solução factível, como a apresentada na Figura 3.5 (c), existe. De fato, se a caixa tipo 1 da solução inicial for substituída pela caixa tipo 4 não há forma de se obter uma solução factível, ver (d) e (e). De todo modo, verifica-se que pequenas alterações demandam um grande esforço para manter a factibilidade da solução.

Uma alternativa é fazer a busca local em uma representação indireta da solução, por exemplo, com um vetor representando a seqüência com que as caixas são carregadas por uma heurística construtiva. Seja (1,2,2,3,4) a seqüência de tipos de caixas para proceder o carregamento do contêiner em (a). A primeira troca de caixas considerada anteriormente, nesta representação, fica (**3**,2,2,**1**,4). A Figura 3.5 (f) mostra a correspondente solução após decodificação, na qual é possível carregar apenas duas caixas que totalizam 57,29% de ocupação do volume do contêiner. De acordo com as dimensões das caixas empacotadas, a única forma de carregar uma segunda caixa do tipo 2 é permitir projeção da base de apoio. Este procedimento também é relativamente caro computacionalmente e está sujeito a gerar muitas soluções de baixa qualidade, pois não avalia aspectos físicos do padrão de carregamento.

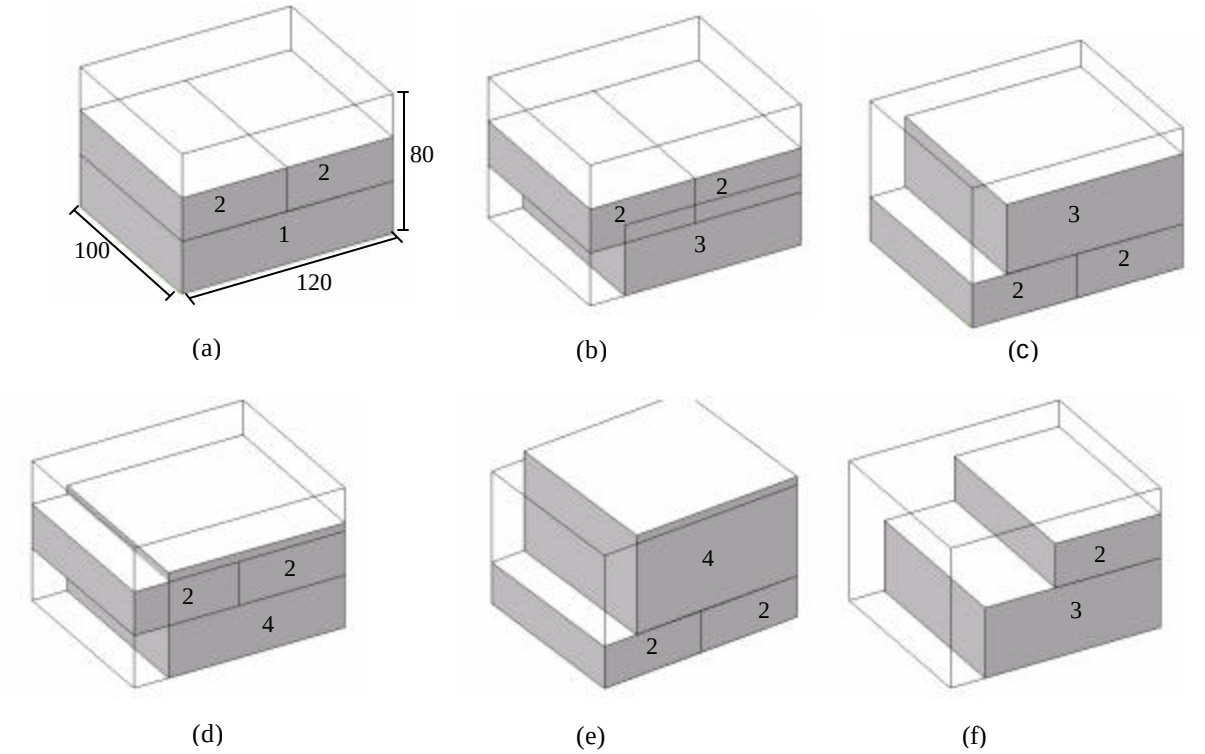


Figura 3.5. Soluções do exemplo 1

Para problemas em que é mais efetivo construir soluções do que aplicar busca local, a fase de busca local do método de múltiplos inícios pode ser eliminada para que seja dada mais ênfase à construção de novas soluções. Nesse caso, deve-se determinar uma vizinhança construtiva que define a forma como as soluções são geradas. A vizinhança construtiva nada mais é do que estipular como os elementos memória, aleatoriedade e grau de reconstrução são inseridos no procedimento que gera as soluções. Na próxima seção é proposto um algoritmo de múltiplos inícios com memória adaptativa para resolver o problema de carregamento de contêiner.

3.2 Algoritmo de múltiplos inícios para o problema de carregamento de contêiner

Na heurística proposta o padrão de carregamento é construído com cubóides de tamanho variável. A denominação tamanho variável está relacionada com o fato do algoritmo não calcular sempre o mesmo tamanho de bloco de caixas para espaços vazios iguais. Esta abordagem pode ser vista como uma generalização dos procedimentos que utilizam carregamento caixa a caixa (George e Robinson, 1980; Bischoff e Ratcliff, 1995a; Bischoff, 2004), cubóides (Eley, 2002) e arranjos de até dois cubóides (Bortfeldt e Gehring, 1998; Bortfeldt et al., 2003; Mack et al., 2004).

Abordagens que constroem padrões de carregamento caixa a caixa procuram posicionar caixas do mesmo tipo próximas umas das outras, favorecendo de um certo modo, a construção de blocos. No entanto, no decorrer da construção da solução a complexidade do padrão de carregamento tende a aumentar, produzindo padrões do tipo "quebra-cabeça". Em contrapartida, abordagens que utilizam cubóides tendem a produzir padrões de carregamento com menor complexidade, mas em alguns casos em detrimento do volume ocupado. Os testes computacionais demonstram que algoritmos que utilizam cubóides apresentam melhor desempenho com instâncias fracamente heterogêneas e algoritmos que carregam caixa a caixa, com ou sem camadas verticais, apresentam melhor desempenho com instâncias fortemente heterogêneas (Bortfeldt e Gehring, 1998; Eley, 2002; Gehring e Bortfeldt, 1997; Bischoff, 2004). O uso de cubóides de tamanho variável associado a estruturas de memória possibilita integrar as vantagens de cada abordagem, proporcionando maior robustez ao algoritmo.

Seguindo a definição de cubóide dada na seção 2.3 do Capítulo 2, denomina-se $\overline{c}_{tki} = (\overline{c}^{x}_{tki}, \overline{c}^{y}_{tki}, \overline{c}^{z}_{tki})$ o maior cubóide formado por caixas do tipo t com orientação k que pode ser designado para o espaço vazio i. O número de caixas que determina cada dimensão do cubóide $\overline{c}_{tki}$ que pode ser designado para o espaço vazio i com dimensões $X_i \times Y_i \times Z_i$, é calculado da seguinte forma:

$$\overline{c}^{z}_{tki} = \min\left\{\left\lfloor \frac{Z_i}{d_{3tk}} \right\rfloor, \hat{q}_t\right\} \qquad (3.1)$$

$$\overline{c}_{tki}^{y} = \min\left\{\left\lfloor \frac{Y_i}{d_{2tk}} \right\rfloor, \quad \left\lfloor \frac{\hat{q}_t}{\overline{c}_{tki}^{z}} \right\rfloor\right\} \tag{3.2}$$

$$\overline{c}_{tki}^{x} = \min\left\{\left\lfloor \frac{X_i}{d_{1tk}} \right\rfloor, \left\lfloor \frac{\hat{q}_t}{\overline{c}_{tki}^{z}\overline{c}_{tki}^{y}} \right\rfloor\right\}, \tag{3.3}$$

em que $\hat{q}_t$ representa o número de caixas do tipo t não carregadas.

Qualquer cubóide c_{tki} que satisfaça $1 \leq c_{tki}^{x} \leq \overline{c}_{tki}^{x}$, $1 \leq c_{tki}^{y} \leq \overline{c}_{tki}^{y}$ e $1 \leq c_{tki}^{z} \leq \overline{c}_{tki}^{z}$ pode compor o padrão de carregamento. Desta forma, o uso de cubóides com tamanho variável gera uma árvore construtiva em que o grau de ramificação é, no máximo, $\sum_{t=1}^{m}\sum_{k=1}^{6}\left|\overline{c}_{tki}\right|$. Esse procedimento aumenta consideravelmente o grau de ramificação se comparado com carregamento caixa a caixa ou cubóides de tamanho fixo que possuem, no máximo, $6 \cdot m$ soluções parciais novas em cada nó da árvore construtiva. No entanto, o uso de memória e funções de tendência para priorizar a construção de determinados cubóides minimiza esta aparente desvantagem.

Para um melhor entendimento, algumas etapas do algoritmo são descritas com mais detalhes antes de apresentá-lo como um todo.

3.2.1 Representação da solução e identificação dos espaços vazios

As estruturas de dados da solução e os procedimentos relacionados à identificação de espaços vazios dentro do contêiner consomem grande parte dos recursos computacionais exigidos por implementações de algoritmos para PCE3D. Desafortunadamente, poucos trabalhos da literatura apresentam estas estruturas e procedimentos em detalhes.

Neste trabalho é utilizada a adaptação da representação espacial de Ngoi et al. (1994) proposta por Bischoff (2006) para armazenar soluções de PCE3D.

O procedimento que identifica os espaços vazios faz uma varredura em uma matriz bidimensional que representa a visão superior do contêiner, denominada matriz superfície. Os valores dos elementos desta matriz correspondem à cota de um segmento da superfície do contêiner. Informações referentes a coordenadas, tipo e orientação das caixas carregadas são armazenadas em uma estrutura separada. A seguir é apresentado

um exemplo da construção de um carregamento formado por cubóides. A forma como os cubóides são calculados é descrita na próxima seção.

Exemplo 2. Considere um contêiner com dimensões $100 \times 50 \times 50$ em uma dada unidade de comprimento e o conjunto com dois tipos de caixas descrito na Tabela 3.2. A Tabela 3.3 apresenta as orientações permitidas para cada caixa, na qual cada orientação corresponde a uma permutação das dimensões apresentadas na Tabela 3.2. Por exemplo, a orientação $k = 3$ utiliza a dimensão 2 para determinar o comprimento, a dimensão 1 para a largura e a dimensão 3 para a altura. Deste modo, a caixa do tipo 1 com orientação 3 deve ser considerada com dimensões $25 \times 30 \times 10$.

Tabela 3.2. Conjunto de caixas do exemplo 2

Tipo	Dim 1	Dim 2	Dim 3	Quant.
1	30	25	10	25
2	25	15	10	25

Tabela 3.3. Orientações factíveis do conjunto de caixas do exemplo 2

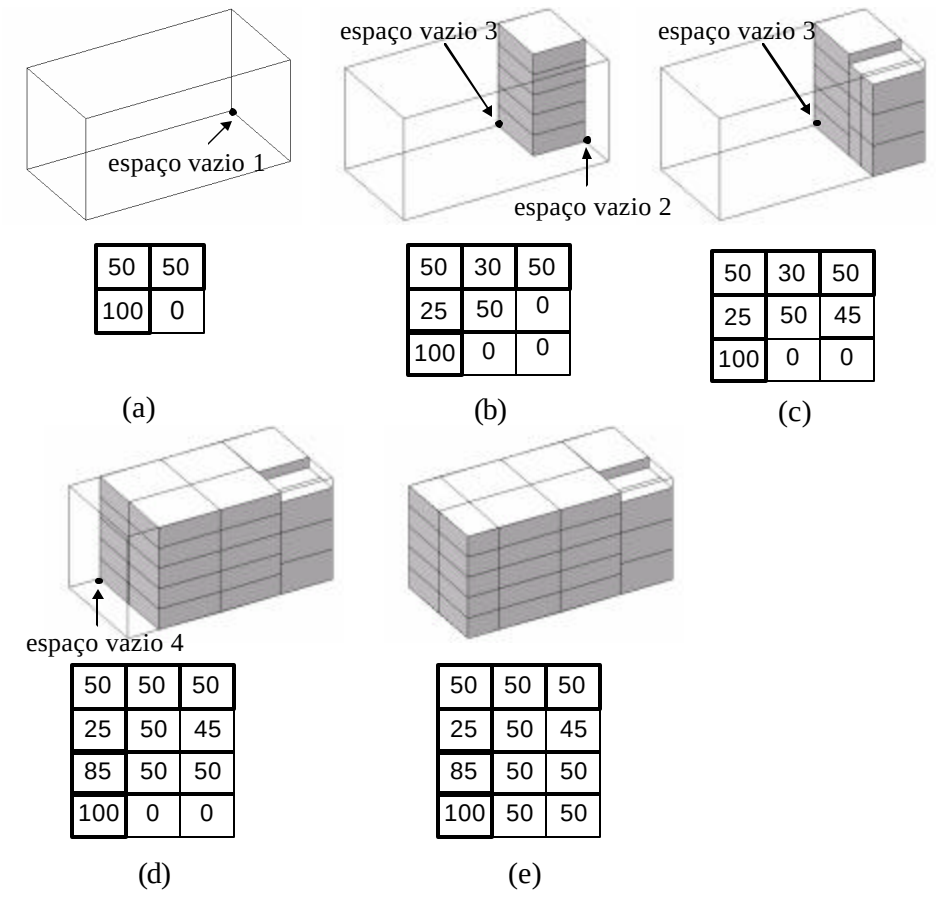
k	1 (1,2,3)	2 (1,3,2)	3 (2,1,3)	4 (2,3,1)	5 (3,1,2)	6 (3,2,1)
u_{1k}	1	1	1	0	1	0
u_{2k}	1	1	1	0	1	0

As restrições quanto à orientação das caixas são representadas através dos parâmetros binários u_{tk}. Se a orientação k da caixa do tipo t é permitida então $u_{tk} = 1$ e, caso contrário, $u_{tk} = 0$. De acordo como exposto na Tabela 3.3, as orientações $k = 4$ e $k = 6$ não são permitidas para os dois tipos de caixa, ou seja, $u_{14} = u_{24} = u_{16} = u_{26} = 0$.

A Figura 3.6 apresenta os espaços vazios gerados e as respectivas modificações da matriz superfície desde o início, quando o contêiner está vazio, até o final do carregamento. Em todas as matrizes a primeira linha e coluna armazenam as dimensões utilizadas para calcular os espaços vazios, com o valor do elemento em comum representando a altura do contêiner. Os demais elementos da matriz representam a cota da correspondente superfície. Em (a) o contêiner possui um único espaço vazio com cota zero e dimensões iguais às do contêiner. Quando o primeiro cubóide $c_{131} = (1, 1, 5)$ é carregado no único espaço vazio disponível, uma linha e uma coluna são acrescidas na matriz superfície, ver (b). Dois novos espaços vazios são gerados, $i = 2$ com coordenadas $(0, 30, 0)$ e dimensões $100 \times 20 \times 50$ e $i = 3$ com coordenadas $(25,0,0)$ e

dimensões $75 \times 50 \times 50$. O espaço vazio superior é descartado por ter cota igual à altura do contêiner.

O carregamento do segundo cubóide $c_{222} = (1, 2, 3)$ no espaço vazio $i = 2$ não altera as dimensões da matriz superfície e nenhum espaço vazio é gerado, uma vez que o espaço vazio com coordenadas $(25, 30, 0)$ está integralmente contido em $(25, 0, 0)$. O espaço vazio superior é descartado porque a altura é menor do que a menor dimensão dos tipos de caixa disponíveis para carregamento. Em (d), o terceiro cubóide $c_{113} = (2, 2, 5)$ é carregado no espaço vazio $i = 3$ e uma linha é acrescida à matriz superfície. O último cubóide $c_{234} = (1, 2, 5)$ é carregado no espaço vazio $i = 4$ de coordenadas $(85, 0, 0)$, ver (e). O volume ocupado corresponde a 99,9% do volume do contêiner.



50	50
100	0

50	30	50
25	50	0
100	0	0

50	30	50
25	50	45
100	0	0

50	50	50
25	50	45
85	50	50
100	0	0

50	50	50
25	50	45
85	50	50
100	50	50

Figura 3.6. Carregamento utilizando cubóides

A Figura 3.7 apresenta em detalhes os espaços formados após o carregamento do primeiro cubóide do Exemplo 2. O espaço superior, conforme exposto anteriormente, foi descartado por possuir altura zero. A primeira linha e coluna da matriz servem para armazenar as dimensões utilizadas para calcular as coordenadas e dimensões dos espaços vazios.

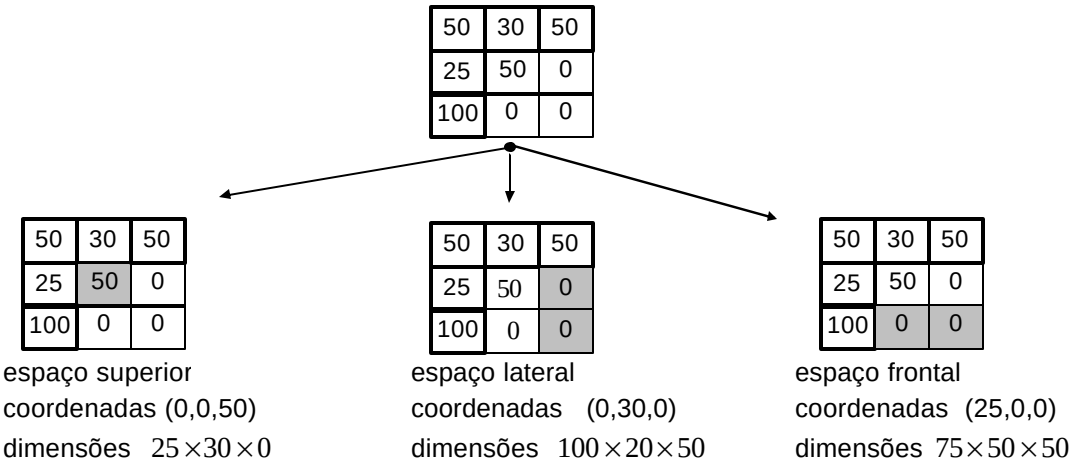


Figura 3.7. Formação dos espaços vazios após o carregamento do cubóide c_{131}

Algumas características adicionais dos espaços vazios são utilizadas para auxiliar no cálculo do padrão de carregamento, como o comprimento com suporte. Esta característica é utilizada no cálculo dos cubóides para evitar uma excessiva fragmentação da frente do padrão de carregamento.

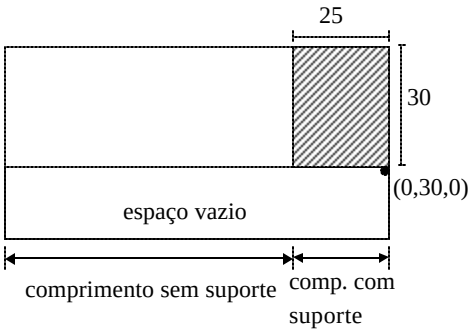


Figura 3.8. Características adicionais dos espaços vazios: comprimento com suporte

Por exemplo, o espaço vazio na Figura 3.6 (b) com coordenadas $(0, 30, 0)$ possui comprimento 100, dos quais 25 possui suporte lateral à direita do contêiner, ver

Figura 3.8. Representando o comprimento com suporte lateral do espaço vazio *i* por s_i^x, a expressão (3.3) pode ser alterada conforme segue:

$$c_{tki}^{x} = \min\left(\min\left(\left\lfloor \frac{X_i}{d_{1tk}} \right\rfloor, \left\lfloor \frac{s_i^x}{d_{1tk}} \right\rfloor \right), \left\lfloor \frac{\hat{q}_t}{c_{tki}^z c_{tki}^y} \right\rfloor \right) \qquad (3.4)$$

Observe que o quociente entre o comprimento com suporte, s_i^x, e o comprimento da caixa que compõem o cubóide, d_{1tk}, fornece o número máximo de caixas que podem ser carregadas com suporte lateral. Deste modo, a expressão (3.4) evita que sejam calculados blocos com excessiva projeção ao longo do comprimento do contêiner e também promove uma melhora na estabilidade horizontal.

Outra forma de evitar a fragmentação da frente do padrão de carregamento é descartar espaços vazios considerados *muito longe* (Verweij, 1996). Um espaço vazio é dito *muito longe* se a sua distância, ao longo do eixo x, em relação ao primeiro espaço vazio mais ao fundo do contêiner exceder a maior dimensão das caixas disponíveis para carregamento. De outra forma, se x_0 representar a coordenada do espaço vazio mais ao fundo do contêiner e $d_{1k_0t_0}$ o maior comprimento das caixas não carregadas, então qualquer espaço vazio que satisfaça $x_i - x_0 > d_{1k_0t_0}$ é considerado *muito longe*, ver Figura 3.9. Este procedimento é particularmente útil em situações em que a carga é fracionada, pois facilita os procedimentos de descarga.

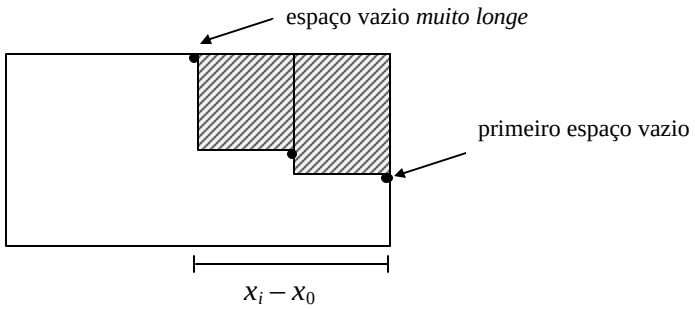


Figura 3.9. Espaço vazio *muito longe*

Verweij (1996) argumenta que a representação espacial de Ngoi et al. (1994), quando implementada em um computador, pode consumir memória em grande quantidade, principalmente para instâncias fortemente heterogêneas. Esta desvantagem é minimizada na estrutura adaptada utilizada neste trabalho devido ao uso de apenas uma matriz bidimensional, ao contrário de Ngoi et al. que utilizam uma matriz tridimensional. Além disso, a construção do padrão de carregamento em blocos tende a diminuir a fragmentação da superfície de carregamento e, conseqüentemente, os procedimentos que incluem linhas e/ou colunas na matriz superfície são menos requisitados.

3.2.2 Cálculo e avaliação dos cubóides

O cálculo de um cubóide candidato para compor o padrão de carregamento é feito em duas etapas. Inicialmente, para cada espaço vazio i são avaliados os correspondentes cubóides $\overline{c}_{tk}$ e, em seguida, são calculadas reduções no número de caixas que compõem o cubóide selecionado. Estas reduções proporcionam o cálculo de cubóides com tamanhos distintos para um dado espaço vazio.

Na primeira etapa, critérios para avaliar a combinação cubóide e espaço vazio são definidos. Estes critérios, em geral, estão relacionados com as características das instâncias, por isso mais de um conjunto de critérios pode ser utilizado. Dado um espaço vazio i com volume ve_i e coordenadas (x_i, y_i, z_i) e um cubóide $\overline{c}_{tki}$, utiliza-se sete critérios conforme descrito a seguir:

1. maior adequação entre as dimensões do cubóide e do espaço vazio.

$$C_1 = ad\left(X_i, \overline{c}^x_{tki} \cdot d_{1kt}\right) + ad\left(Y_i, \overline{c}^y_{tki} \cdot d_{2kt}\right) + ad\left(Z_i, \overline{c}^z_{tki} \cdot d_{3kt}\right)$$

$$ad\left(D, \overline{\overline{cd}}\right) = \begin{cases} 1 & \text{se } D - \overline{\overline{cd}} = 0 \\ 1 \Big/ \left(D - \overline{\overline{cd}}\right) & \text{caso contrário} \end{cases}$$

em que D representa uma dimensão das dimensões X_i, Y_i, Z_i do espaço vazio e $\overline{\overline{cd}}$ a correspondente a uma dimensão $\overline{c}^x_{tki} \cdot d_{1kt}, \overline{c}^y_{tki} \cdot d_{1kt}, \overline{c}^z_{tki} \cdot d_{1kt}$ do cubóide.

Observe que a medida da adequação atribui valor máximo quando as dimensões do cubóide e do espaço vazio são iguais.

2. menor diferença entre a altura do cubóide e do espaço vazio

   $C_2 = Z_i - d_{3tk}\overline{c}^{z}_{tki}$

   Este critério favorece a construção de superfícies planas pouco fragmentadas.

3. maior preenchimento do espaço vazio

   $C_3 = \frac{v_t\left|\overline{c}_{tki}\right|}{ve_i}$

4. maior área da base

   $C_4 = d_{1tk}\ \overline{c}^{x}_{tki}\, d_{2tk}\ \overline{c}^{y}_{tki}$

   Semelhante a C_2, este critério promove a redução da fragmentação da superfície do padrão de carregamento.

5. menor proeminência ao longo do comprimento do contêiner

   $C_5 = d_{1tk}\ \overline{c}^{x}_{tki} + x_i$

   Este critério é utilizado para diminuir a complexidade da frente do padrão de carregamento.

6. espaço vazio com menor coordenada na largura

   $C_6 = y_i$

7. menor perda relativa na capacidade de empilhamento.

$$C_7 = \begin{cases} \left(B_i - \overline{c}^{z}_{itk}\,\frac{w_t}{d_{1tk}\cdot d_{2tk}}\right)\frac{Z_i - \overline{c}^{z}_{itk}\cdot d_{3tk}}{\overline{c}^{z}_{itk}\cdot d_{3tk}}, & \text{se} \quad B_i < \overline{c}^{z}_{itk}\,\frac{w_t}{d_{1tk}\cdot d_{2tk}} \\ 0 \text{ caso contrário} \end{cases}$$

   em que B_i é a capacidade de empilhamento do espaço vazio i.

   Este critério é utilizado quando restrições de empilhamento são consideradas, e proporciona uma medida de degradação da capacidade

de empilhamento das superfícies dos espaços vazios. O primeiro fator corresponde à capacidade de empilhamento resultante se o cubóide $\overline{c}_{tki}$ for carregado no espaço vazio i. O segundo fator serve para ajustar o valor do critério de forma a favorecer a escolha de cubóides que preencham melhor a altura do espaço vazio. O valor final é nulo se a capacidade de empilhamento do espaço vazio após o carregamento for menor que zero.

Os sete critérios de avaliação descritos acima são combinados em quatro conjuntos, em que o primeiro critério é o mais importante e os restantes servem como desempate. O mecanismo proposto por Prais e Ribeiro (2000) é utilizado para selecionar de forma adaptativa um conjunto de critérios de avaliação mais adequado para cada instância.

A seleção utiliza o valor de função objetivo das soluções geradas no decorrer das iterações. Quanto maior o valor médio de função objetivo associado ao conjunto de critérios de avaliação maior a probabilidade de este ser escolhido. Seja CRT_1, CRT_2, CRT_3, CRT_4 quatro conjuntos de critérios de avaliação, $f(S_{incumbente})$ o valor da função objetivo da melhor solução conhecida (solução incumbente) e $\overline{f(S_i)}$ a média do valor de todas as soluções geradas com o conjunto CRT_i. A probabilidade de seleção é recalculada a cada K iterações com $p_i = g_i / \sum_{j=1}^{n} g_j$, em que $g_i = \left(\overline{f(S_i)} / f(S_{incumbente}) \right)^f$ para $i = 1,2,3,4$. O parâmetro f deve assumir valores maiores que 1 para proporcionar uma discriminação entre g_i, $i = 1,2,3,4$. Inicialmente é necessário coletar informações suficientes sobre a qualidade que o uso de cada conjunto de critérios de avaliação pode proporcionar, por isso nas K primeiras iterações uma probabilidade $p_i = 1/4$, $i = 1,2,3,4$, é associada à escolha do conjunto i.

Os melhores t cubóides máximos, avaliados conforme descrito acima, são arranjados em ordem decrescente de mérito em um lista restrita de candidatos (LRC). Um cubóide na lista é então selecionado com uma probabilidade baseada na posição que ocupa na LRC a partir de uma função de tendenciosidade que favorece os elementos com melhor classificação. Seja $pos(s)$ a posição do elemento s na LRC e seja $bias(pos(s))$ a função de tendenciosidade. A probabilidade $p(s)$ de selecionar o elemento s é dada por:

$$p(s) = \frac{bias(pos(s))}{\sum_{s' \in LCR} bias(pos(s'))} \qquad (3.5)$$

De forma semelhante, o cálculo das reduções na quantidade de caixas que compõem o cubóide selecionado $\overline{c}_{tki}$ ao longo das três dimensões é feita probabilisticamente, a partir de uma distribuição de probabilidade que favorece a escolha de valores próximos a zero. Genericamente, seja qb a quantidade de caixas utilizadas ao longo de uma das dimensões de um determinado cubóide $\overline{c}_{tki}$ e r a redução que pode assumir valores no intervalo $[0, qb\text{-}1]$, então a probabilidade de escolher r, pertencente a este intervalo, é dada por:

$$p(r) = \frac{bias(r)}{\sum_{r'=0}^{qb-1} bias(r')} \qquad (3.6)$$

em que $bias(r)$ é uma função de tendência que favorece reduções próximas a zero.

Uma função de tendência polinomial $bias(r) = r^{-n}$ é utilizada na escolha do elemento da LRC e no cálculo das reduções nos cubóides. Seja $f(S_{incumbente})$ o valor da solução incumbente e $f(S_{parcial})$ o valor de uma solução parcial. O expoente n é definido conforme segue:

$$n = \begin{cases} n_1 \text{ para } f(S_{parcial}) < r \cdot f(S_{incumbente}) \\ n_2 \text{ caso contrário} \end{cases}$$

em que n_1, n_2 e r são parâmetros do algoritmo.

No Exemplo 2, o número de caixas ao longo de cada dimensão de $\overline{c}_{131}$ é calculado conforme segue:

$$\bar{c}^{z}_{131} = \min\left\{\left\lfloor \frac{50}{10} \right\rfloor, 25\right\} = 5$$

$$\bar{c}^{y}_{131} = \min\left\{\left\lfloor \frac{50}{30} \right\rfloor, \left\lfloor \frac{25}{5} \right\rfloor\right\} = 1$$

$$\bar{c}^{x}_{131} = \min\left\{\min\left\{\left\lfloor \frac{100}{25} \right\rfloor, \left\lceil \frac{100}{25} \right\rceil\right\}, \left\lfloor \frac{25}{5} \right\rfloor\right\} = 4$$

Do cálculo acima, tem-se o cubóide máximo $\bar{c}_{131} = (4, 1, 5)$ e, como $c_{131} = (1, 1, 5)$ é o cubóide carregado, concluí-se que o cubóide máximo sofreu uma redução de 3 caixas no comprimento, ou seja, 15 caixas no total.

3.2.3 Estruturas de memória

As estruturas de memória favorecem a construção de cubóides com características que possam levar a soluções de alta qualidade. O procedimento utilizado é semelhante ao proposto por Fleurent e Glover (1999).

Um conjunto que armazena características de soluções de alta qualidade (*soluções elite*) é utilizado para influenciar a heurística construtiva. Para todas as soluções elite, cada cubóide máximo $\bar{c}_{tki}$ e o respectivo cubóide reduzido c_{tki} são armazenados, bem como o valor de função objetivo. A finalidade é favorecer o cálculo de reduções que resultem em blocos semelhantes àqueles utilizados nas soluções elite.

Seja E o conjunto de soluções elite e $emax$ o número máximo de soluções elite, $|E| \leq emax$. Inicialmente o conjunto elite está vazio, $E = \{\}$, e, enquanto o número de soluções é menor que $emax$, qualquer solução, desde que diferente das que já foram inseridas, pode entrar em E. A partir do momento que o número de soluções armazenadas em E torna-se igual a $emax$ a regra de aceitação passa a considerar a qualidade da solução candidata, situação em que só é permitido uma solução entrar em E se ela possuir um valor de função objetivo maior que a pior solução elite. Uma solução que entra em E substitui a solução com pior avaliação.

A influência da memória ocorre a partir da definição de uma função de intensidade $h(\bar{c}_{tki},d,r)$, que considera a freqüência com que a redução r aplicada a $\bar{c}_{tki}$ ao longo da dimensão d aparece nas soluções elite, e pode ser definida conforme segue:

$$h(\bar{c}_{tk},d,r) = \sum_{S\in E:(\bar{c}_{tki},d,r)\in S} \frac{f(S)}{f(S_{incumbente})}$$

em que $f(S)$ é o valor que a função objetivo f atribui à solução elite S.

Observe que os valores possíveis da função de intensidade pertencem ao intervalo $[0,1]$. Deste modo, a expressão (3.6) pode ser modificada conforme segue:

$$p'(r) = \frac{bias(r) + h(\bar{c}_{tki},d,r)}{\sum_{r'=0}^{qb-1}\left[bias(r') + h(\bar{c}_{tk},d,r')\right]}$$

Para um melhor controle da influência da memória, a probabilidade de escolher a redução r é calculada como uma combinação convexa da função de intensidade e de tendência, conforme segue:

$$p''(r) = \frac{(1-l)\cdot bias(r) + l\cdot h(\bar{c}_{tki},d,r)}{\sum_{r'=0}^{qb-1}\left[(1-l)\cdot bias(r') + l\cdot h(\bar{c}_{tki},d,r')\right]}, \quad l\in[0,1]$$

em que l é um parâmetro que controla a influência da memória no cálculo. Quando $l=0$ a memória não exerce nenhuma influência, e quando $l=1$ a influência da memória é máxima. No último caso, o cálculo é feito somente em base da história do processo.

A memória só é utilizada após um determinado número de iterações, a partir do qual julga-se que o algoritmo já armazenou informações suficientes sobre a qualidade dos cubóides utilizados nos padrões de carregamento gerados. Esse número de iterações é determinado pelo parâmetro do algoritmo K. Este parâmetro também é utilizado para selecionar diferentes conjuntos de critérios de avaliação.

O cálculo do cubóide $c_{131}=(4,1,5)$ do Exemplo 2 pode ser utilizado para ilustrar esse procedimento. Para tanto, considere a função de tendência linear, $bias(r)=1/(r+1)$, $\lambda = 0{,}6$ e os cubóides de duas soluções elites apresentadas na Tabela 3.4, ordenadas segundo a função objetivo f que determina a porcentagem do volume ocupado do

contêiner. A identificação dos espaços vazios não é explicitada, uma vez que os cubóides são agrupados a partir das dimensões, tipo e orientação das caixas. Cada solução do conjunto elite possui um cubóide igual ao $\overline{c}_{131}$ do Exemplo 2, sendo que na solução S_1 as reduções são nulas ($\overline{c}_{13\_} = c_{13\_}$) e na solução S_2 uma redução de 2 unidades é aplicada no comprimento ($\overline{c}^{x}_{13\_} = 4$ e $c^{x}_{13\_} = 2$).

Tabela 3.4. Conjunto de soluções elite

Solução	Cubóides		f
S_1	$\overline{c}_{13\_}$ = (4,1,5) $\overline{c}_{22\_}$ = (4,2,3)	$c_{13\_}$ = (4,1,5) $c_{22\_}$ = (4,2,3)	96,0%
S_2	$\overline{c}_{13\_}$ = (4,1,5) $\overline{c}_{12\_}$ = (3,2,2) $\overline{c}_{12\_}$ = (2,3,3)	$c_{13\_}$ = (2,1,5) $c_{12\_}$ = (3,2,2) $c_{12\_}$ = (2,3,3)	93,0%

A Tabela 3.5 apresenta as probabilidades de escolha das reduções para $\overline{c}_{131}$ com e sem a influência de memória. Os valores das reduções são listados na primeira coluna e os valores para a função de tendência, função de intensidade e probabilidades de escolha são listados da segunda a quinta coluna para o comprimento e da sexta a nona coluna para a altura. As reduções para a largura não são consideradas porque $\overline{c}^{y}_{131} = 1$ e, obviamente, não pode haver redução nesta dimensão. É possível observar que o uso da memória aumenta a probabilidade das reduções que são utilizadas nas soluções elite (colunas 4, 5, 8 e 9). Ainda na Tabela 3.5, os valores hachurados representam as probabilidades de escolha das reduções aplicadas no Exemplo 2 para o cubóide c_{131}, que correspondem a três unidades no comprimento e zero unidades na largura.

As probabilidades da redução no comprimento $r = 1$ e $r = 3$ diminuem de $0{,}2439$ para $0{,}1408$ e de $0{,}1220$ para $0{,}0742$, respectivamente, quando a influência da memória é considerada. Isto deve-se ao fato destas reduções não aparecerem nos cubóides do conjunto elite, ao contrário das reduções $r = 0$ e $r = 2$, cujas probabilidades aumentam de $0{,}4878$ para $0{,}4972$ e de $0{,}1463$ para $0{,}2912$, respectivamente. No que se refere à altura, nenhuma das soluções elite apresenta redução nessa dimensão. Por este motivo a probabilidade da redução $r = 0$ aumenta de $0{,}4444$ para $0{,}6667$ e as

probabilidades de todas as demais reduções diminuem quando a influência da memória é considerada.

Tabela 3.5. Probabilidades de escolha das reduções do cubóide c_{131}

r	comprimento				altura			
	$bias(r)$	$h(\bar{c}_{13}, x, r)$	$p(r)$	$p''(r)$	$bias(r)$	$h(\bar{c}_{13}, y, r)$	$p(r)$	$p''(r)$
0	1,00	0,51	0,4878	0,4972	1,00	1,00	0,4444	0,6667
1	0,50	0,00	0,2439	0,1408	0,50	0,00	0,2222	0,1333
2	0,30	0,49	0,1463	0,2912	0,30	0,00	0,1333	0,0800
3	0,25	0,00	0,1220	0,0742	0,25	0,00	0,1111	0,0667
4	-	-	-	-	0,20	0,00	0,0889	0,0533

3.2.4 Grau de reconstrução

O algoritmo proposto utiliza um procedimento de variação do grau de reconstrução com regras que garantem a geração de soluções distintas da corrente (Glover, 2000). Este procedimento possui algumas semelhanças com a vizinhança do algoritmo busca tabu de Bortfeldt e Gehring (1998) e a busca local de Moura e Oliveira (2005). A melhor solução obtida em um número de iterações, determinado pelo parâmetro do algoritmo *fre_gr*, é selecionada para dar inicio ao procedimento que varia o grau de reconstrução, cujas principais idéias são detalhadas a seguir.

Considere uma solução S que tenha sido construída em p passos por uma heurística construtiva, e seja $LRC_j = \{ s_1, s_2, \ldots, s_{n_j} \}$ a lista restrita de candidatos do passo j com n_j candidatos, ordenada segundo um critério de avaliação. Se a solução S é representada por uma seqüência $S = (s_1, s_2, \ldots, s_p)$ então s_j denota o índice do cubóide escolhido no passo j, $s_{s_j} \in LRC_j$, $s_j \leq n_j$. Dado um passo j_0, o procedimento consiste em construir uma solução parcial $S_{parcial}$ com os primeiros j_0-1 passos de S, inserir um cubóide $s_{s'_{j_0}} \in LRC_{j_0}$, $s'_{j_0} \neq s_{j_0}$, e construir novas soluções a partir de $S_{parcial}$ por um número de iterações de construção limitado pelo parâmetro *iter_parcial*. O cubóide inserido na solução parcial no passo j_0 compreende o que se denomina regra de diferenciação, ou seja, garante que soluções diferentes de S serão geradas. De forma a reduzir o esforço computacional deste procedimento, somente os primeiros max_c

cubóides de cada lista de candidatos podem ser utilizados no processo de diferenciação. Com o mesmo objetivo, as soluções parciais utilizadas para gerar novas soluções devem ter volume inferior ao valor do parâmetro max_vp multiplicado pelo volume ocupado da solução incumbente, com $0 < max\_vp < 1$.

A Figura 3.10 exemplifica o procedimento grau de reconstrução utilizando o conjunto de caixas e a solução do Exemplo 2. A solução em (a) é construída em quatro passos e a solução parcial com dois passos é apresentada em (b). A Figura 3.10 (c) representa a lista de candidatos do passo $j_0 = 3$ composta por três candidatos, $LRC_3 = \{c_{113}, c_{133}, c_{243}\}$. Observe que a solução apresentada em (a) foi construída com o primeiro cubóide da lista, ou seja, c_{113}. Seguindo o procedimento, o segundo cubóide da lista de candidatos c_{243} é escolhido para ser inserido na solução parcial, de acordo com a regra de diferenciação que exige um cubóide distinto daquele originalmente escolhido no passo $j_0 = 3$, ver cubóide cinza escuro em (d). Finalmente, os cubóides da solução parcial em (d) são fixados e o algoritmo de construção gera novas soluções a partir desta solução parcial por *iter_parcial* iterações de construção.

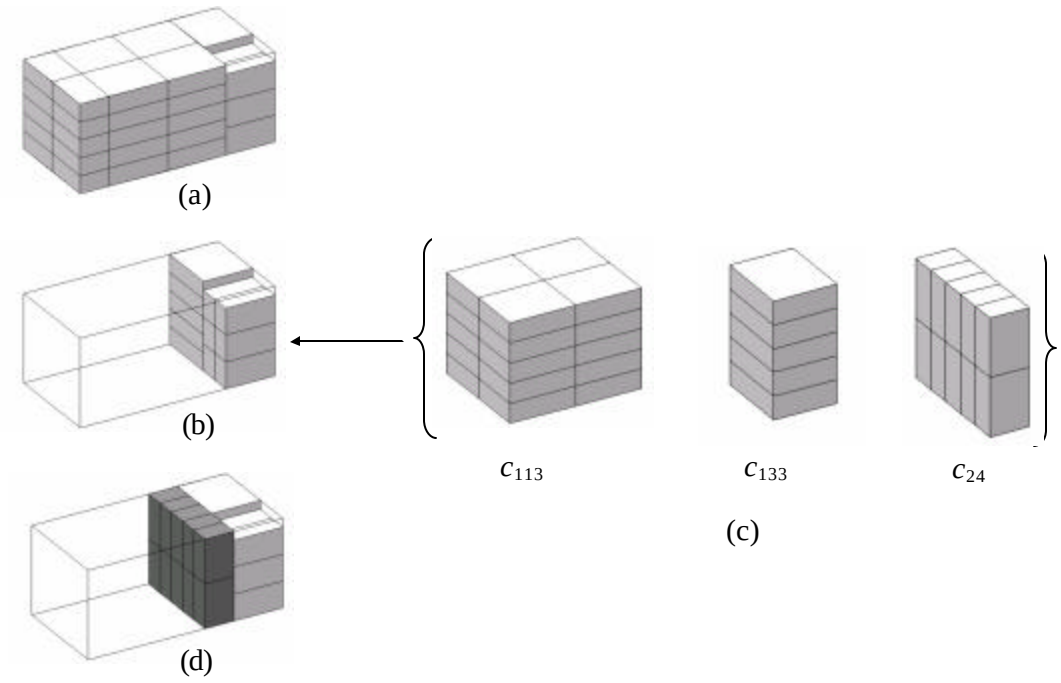


Figura 3.10. Variação do grau de reconstrução

3.2.5 Algoritmo proposto

A Figura 3.11 apresenta o algoritmo de múltiplos inícios para o problema de carregamento de contêiner em pseudocódigo. Na linha 1 dados da instância e parâmetros do algoritmo são processados. Nas linhas 2 a 4 as estruturas que armazenam as soluções S_{freq_gr} e $S_{incumbente}$ são inicializadas, bem como a variável *iter* destinada a contar o número de iterações. O laço principal é compreendido entre as linhas 5 e 17. Na linha 7 um conjunto de critérios de avaliação dos cubóides é escolhido com o mecanismo proposto por Prais e Ribeiro (2000). Na linha 8 é invocado o procedimento que constrói soluções a partir do conjunto vazio e na linha 9 o conjunto elite é atualizado. A solução incumbente é atualizada na linha 10 e a melhor solução das últimas *freq_gr* iterações na linha 11. Na linha 12 é verificado se o procedimento que constrói soluções a partir de soluções parciais deve ser invocado (linha 13), se isto ocorrer, a solução incumbente é atualizada na linha 14 e S_{freq_gr} é reiniciada na linha 15.

	Procedimento múltiplos inícios para carregamento de contêineres
1	Leia e processe os dados da instância e parâmetros do algoritmo
2	$S_{freq\_gr} \leftarrow \{\}$
3	$S_{incumbente} \leftarrow \{\}$
4	$iter \leftarrow 0$
5	**Enquanto** (critério de parada não satisfeito) **faça**
6	$iter \leftarrow iter + 1$
7	*Critérios* ← Escolha um conjunto de critérios de avaliação dos cubóides
8	$S \leftarrow$ Construção_Aleatorizada({}, *Critérios*)
9	Atualiza conjunto elite com S
10	**Se** $f(S_{incumbente}) < f(S)$ **então** $S_{incumbente} \neg S$ **Fim_Se**
11	**Se** $f(S_{freq\_gr}) < f(S)$ **então** $S_{freq\_gr} \neg S$ **Fim_Se**
12	**Se** $iter$ MOD $freq\_gr = 0$ **então**
13	$S \leftarrow$ Grau_Reconstrução(S_{freq_gr}, *Critérios*)
14	**Se** $f(S_{incumbente}) < f(S)$ **então** $S_{incumbente} \neg S$ **Fim_Se**
15	$S_{freq\_gr} \leftarrow \{\}$
16	**Fim_Se**
17	**Fim_Enquanto**
18	**Retorna**($S_{incumbente}$)

Figura 3.11. Algoritmo Múltiplos inícios para PCE3D

O algoritmo da heurística construtiva é detalhado na Figura 3.12. A solução $S_{inicial}$ é um dado de entrada que determina se a construção da solução se dá a partir de um conjunto vazio ou de uma solução parcial. Na linha 2 os espaços vazios são identificados utilizando as estruturas e procedimentos descritos na seção 3.2.1. As combinações de todos os espaços vazios e tipos de caixas, em todas as orientações permitidas, são avaliadas para criar a lista restrita de candidatos LRC, ver linhas 5 a 16. O número máximo de elementos da LRC é fixo e determinado pelo parâmetro denominado t. Os elementos da lista LRC são ordenados de acordo com o critério escolhido, ver linha 11. Na linha 17 é feita a escolha de um elemento da lista restrita de candidatos LRC para ser adicionado à solução em construção. Para tanto, é utilizada uma função de tendência, baseada na posição que o cubóide candidato ocupa na LRC, que favorece os elementos melhores classificados. O cálculo das reduções do cubóide escolhido não utiliza memória nas primeiras K iterações, ver linhas 18 e 19. Na linha 21 o contador de passos p é incrementado e na linha 22 a LRC do passo corrente é armazenada em $CLRC_p$ para ser utilizada, se for o caso, no procedimento para variação do grau de reconstrução. Após a inclusão do cubóide na solução, a lista de espaços vazios é atualizada, ver linha 23.

A Figura 3.13 apresenta o algoritmo do procedimento para variação do grau de reconstrução. A melhor solução gerada nas últimas *freq_gr* e um conjunto de critérios para avaliação dos cubóides são recebidos como parâmetros. Na linha 1 a variável j, utilizada para controlar o grau de reconstrução das soluções, é inicializada com valor 1. O laço principal é compreendido entre as linhas 2 e 22. As variáveis l, h e c são inicializadas a cada iteração do laço principal nas linhas 3, 4 e 5. A variável l recebe o índice do cubóide escolhido no passo j da construção da solução S, a variável h é utilizada para percorrer os índices da LRC associada ao passo j, e a variável c é um contador dos cubóides inseridos na solução parcial pela regra de diferenciação. Para um dado grau de reconstrução, soluções distintas são construídas no laço representado nas linhas 6 a 20. Na linha 8 os primeiros j-1 passos da solução S são copiados para $S_{parcial}$ e a regra de diferenciação é aplicada na linha 9, na qual $CLRC_j$ representa a LRC associada ao passo j. Nas linhas 11 a 16 são geradas soluções a partir da solução parcial por *iter_parcial* iterações. Observe que S é atualizada sempre que uma solução com melhor avaliação é gerada, ver linhas 12 e 13. As soluções geradas também são utilizadas para atualizar a

memória, ver linha 14. O procedimento termina quando o volume do carregamento da solução parcial é superior ao valor do parâmetro *max_vp* multiplicado pelo volume ocupado da solução incumbente, $max\_vp \cdot V(S_{incumbente})$, ou quando todos os passos utilizados para construir a solução *S* são examinados, ver linha 22.

	Procedimento Construção_Aleatorizada($S_{inicial}$, *Critérios*)
1	$S \leftarrow S_{inicial}$
2	*lista _ espaços _ vazios* ← espaços vazios de *S*
3	$p \leftarrow 0$
4	**Enquanto** $\sum_{t=1}^{m} \hat{q}_t > 0$ **E** $\lvert lista\_espaços\_vazios \rvert > 0$ **faça**
5	**Para** *i* ¬ 1**até** $\lvert lista\_espaços\_vazios \rvert$ **faça**
6	**Para** *t* ← 1 **até** *m* **faça**
7	**Se** $\hat{q}_t > 0$ **E** a caixa do tipo *t* couber no espaço vazio *i* **então**
8	**Para** *k* ← 1 **até** 6 **faça**
9	**Se** a orientação *k* da caixa do tipo *t* é factível **então**
10	Calcula o cubóide $\overline{c}_{tk}$ conforme seção 3.2.2
11	Atualiza a lista de candidatos *LRC* utilizando *Critérios*
12	**Fim_Se**
13	**Fim_Para**
14	**Fim_Se**
15	**Fim_Para**
16	**Fim_Para**
17	Escolha um elemento da *LRC* utilizando uma função de tendência baseada na posição de cada elemento na lista.
18	***Se*** *iter* < *K* **então** Calcula reduções conforme seção 3.2.2
19	**Senão** Calcula reduções conforme seção 3.2.3 **Fim_Se**
20	Inclua o cubóide escolhido na solução *S*
21	$p \leftarrow p$ +1
22	$CLRC_p$ ¬ *LRC*
23	Atualiza *lista_espaços_vazios* e número de caixas não empacotadas
24	**Fim_Enquanto**
25	**Retorna**(*S*)

Figura 3.12. Algoritmo da heurística construtiva aleatorizada

```
   Procedimento Grau_Reconstrução(S, Critérios)
1  j ← 1
2  Repita
3     l ¬ índice do cubóide escolhido no passo j na construção da solução S
4     h ¬ 1
5     c ¬ 1
6     Enquanto h ≤ n_j E c ≤ max_c faça
7        Se h ¹ l então
8           Copia a solução S para S_parcial até o passo j-1
9           S_parcial ← S_parcial ∪ { c_{t_h k_h i_h} ∈ CLRC_j }
10          r ← 0
11          Enquanto r < iter_parcial faça
12             S_iteração ← Construção_Aleatorizada(S_parcial, Critérios)
13             Se f(S) < f(S_iteração) então S ← S_iteração Fim_Se
14             Atualiza conjunto elite com S_iteração
15             r ← r + 1
16          Fim_Enquanto
17          c ← c + 1
18       Fim_Se
19       h ← h + 1
20    Fim_Enquanto
21    j ← j + 1
22 Até volume de S no passo j > max_vp·V(S_incumbente) OU
        j > número de passos utilizados para construir S
23 Retorna(S)
```

Figura 3.13. Algoritmo para variação do grau de reconstrução da solução

A Tabela 3.6 lista os parâmetros do algoritmo com uma breve descrição de cada um. É importante ressaltar que testes computacionais demonstraram que o ajuste da maioria destes parâmetros é pouco sensível. De fato, apenas quatro (n_1, n_2 *freq_gr* e *max_c*) foram ajustados com diferentes valores para propiciar um maior grau de aleatoriedade na geração de soluções para instâncias em que as dimensões das caixas

são comparáveis as dimensões do contêiner. Pode-se afirmar que não ocorre prejuízo com relação à qualidade das soluções quando o valor dos demais parâmetros é fixado.

Tabela 3.6. Descrição dos parâmetros do algoritmo de múltiplos inícios para PCE3D

Parâmetro	Descrição
K	número iterações a partir do qual a memória passa a influenciar a heurística construtiva. Também é utilizado no procedimento de escolha dos conjuntos de critérios de seleção dos cubóides
$Emax$	número máximo de soluções do conjunto elite
λ	controle da influência da memória no cálculo dos cubóides
r	altera a função de tendência
n_1 e n_2	expoentes utilizados na função de tendência polinomial
f	utilizado no cálculo da escolha entre diferentes conjuntos de critérios de seleção dos cubóides
t	número máximo de elementos da lista restrita de candidatos
$freq\_gr$	numero de iterações que determina a freqüência com a qual o procedimento que varia o grau de reconstrução é aplicado
max_vp	Determina o grau de reconstrução máximo
max_c	número máximo de cubóides utilizados na regra de diferenciação no procedimento que varia o grau de reconstrução
$iter\_parcial$	número de vezes que a solução parcial é reconstruída no procedimento que varia o grau de reconstrução

3.2.6 Adaptações do algoritmo proposto para diferentes conjuntos de restrições

De acordo com o exposto na subseção 2.2.1 do capítulo 2, as restrições consideradas são: orientação, estabilidade, empilhamento e distribuição de peso dentro do contêiner.

As restrições de orientação são trivialmente tratadas pelo algoritmo proposto, enquanto as demais restrições demandam desenvolvimento de procedimentos novos. Estes procedimentos, para cada conjunto de restrições, são descritos a seguir.

Estabilidade

A estabilidade vertical exige que uma porcentagem da área da base de cada caixa seja apoiada pelo piso do contêiner ou por outras caixas. Devido à definição utilizada de espaço vazio, o algoritmo naturalmente satisfaz a restrição de estabilidade vertical com 100% de apoio da base das caixas. No entanto, se projeção da base de apoio é permitida, então os procedimentos que identificam os espaços vazios devem ser modificados. Para exemplificar os novos procedimentos, considere a visão superior do carregamento composto por duas caixas ilustrado na Figura 3.14. Em (a), as dimensões da base do contêiner e das caixas são apresentadas em uma dada unidade de comprimento, bem como a cota de cada superfície. Observe que quatro espaços vazios são identificados tomando como referência as coordenadas $(0,0,0)$. Se a altura do contêiner for 50, os espaços 1 e 2, que oferecem suporte total da base, possuem dimensões $25\times 50\times 20$ e $30\times 25\times 20$, representados respectivamente pelas áreas hachuradas em (b) e (c). Os espaços 3 e 4 são extensões dos dois primeiros e possuem dimensões $(25+5)\times 50\times 20$ e $(30+70)\times(25+25)\times 20$, representados respectivamente pelas áreas hachuradas em (d) e (e). O comprimento do espaço 3, $(25+5)$, expressa o fato do espaço vazio, ao longo desta dimensão, possuir um comprimento com apoio de 25 e uma extensão, sem apoio, de 5 unidades de comprimento. O mesmo raciocínio é aplicado para a largura do espaço vazio.

A forma de testar se uma caixa cabe em um espaço vazio também deve ser modificada. É necessário verificar não somente se o espaço é suficientemente grande, mas também se oferece suporte ao centro de gravidade e se o suporte da base da caixa está acima do mínimo estipulado.

Mais formalmente, dado um espaço vazio, sejam X_c, X_s, W_c e W_s o comprimento com e sem apoio e a largura com e sem apoio, respectivamente. A base de um espaço vazio possui dimensões $X\times W$, em que $X = X_c + X_s$ e $W = W_c + W_s$. Uma caixa com dimensões $d_{1tk}\times d_{2tk}\times d_{3tk}$ só pode ser designada para um espaço vazio se, além de caber no espaço, também atender a seguinte condição:

$$\frac{(d_{1tk} - X_c)(d_{2tk} - W_c)}{d_{1tk}d_{2tk}} \leq AP$$

em que AP representa a porcentagem máxima da área da base que pode ser projetada. Quando AP é nulo, $AP = 0$, não é permitido projeção da área da base.

Esta condição limita a área da base projetada e sempre que $AP < 0{,}5$ existe a garantia que o centro de gravidade da caixa, que coincide com o centro geométrico, tem apoio.

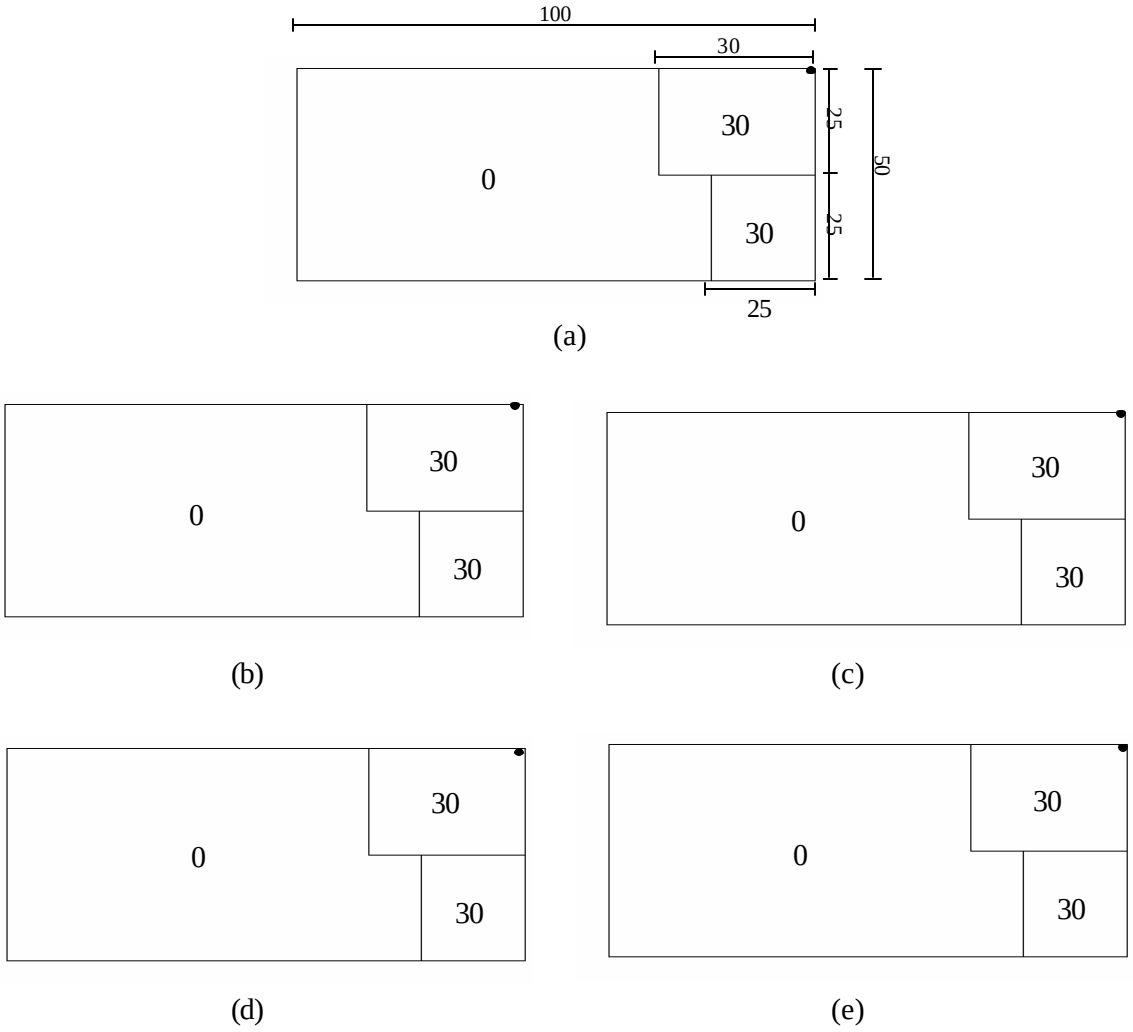


Figura 3.14. Identificação das dimensões dos espaços vazios

Empilhamento

Restrições de empilhamento limitam a pressão que pode ser exercida na superfície superior de cada tipo de caixa. Para incluir este conjunto de restrições no algoritmo basta modificar a estrutura de dados que armazena a solução, de forma a

permitir a avaliação da pressão máxima que pode ser exercida em cada segmento da superfície de carregamento. Para tanto, uma adaptação da representação espacial de Ngoi et al. (1994), proposta por Bischoff (2006), é utilizada. Esta adaptação, como dito antes, é composta por duas matrizes bidimensionais, uma para descrever a posição das caixas e outra para armazenar a capacidade de empilhamento de cada espaço vazio, conforme descrito na subseção 2.3.3 do capítulo 2. A expressão (3.1), utilizada no cálculo dos cubóides, também deve ser modificada de modo a respeitar a capacidade de empilhamento, conforme segue:

$$\bar{c}^{z}_{tki} = \min\left\{\left\lfloor \frac{Z_i}{d_{3tk}} \right\rfloor, \left\lfloor \frac{d_{1tk} \cdot d_{2tk} \cdot b_{tk}}{w_t} \right\rfloor, \left\lfloor \frac{d_{1tk} \cdot d_{2tk} \cdot B_i}{w_t} \right\rfloor, \hat{q}_t \right\},$$

em que w_t representa o peso associado à caixa do tipo t, b_{tk} representa a pressão, em unidades de peso por área, que a face superior do tipo de caixa t com orientação k pode suportar e B_i a pressão máxima que a base do espaço vazio i pode suportar.

O segundo termo entre chaves determina o número máximo de caixas que podem compor uma pilha de caixas do tipo t com orientação k. De forma similar, o terceiro termo determina o número máximo de caixas que podem ser empilhadas sobre a base do espaço vazio i.

Distribuição de peso dentro do contêiner

Restrições de distribuição de peso dentro do contêiner exigem que uma distância máxima entre o centro de gravidade calculado e esperado seja satisfeita. Os trabalhos da literatura que tratam estas restrições inicialmente constroem um padrão de carregamento com camadas verticais isoladas e, em seguida, permutam as camadas verticais para satisfazer as restrições de distribuição de peso. Em geral, é considerado o centro de gravidade ao longo dos eixos x e y.

Dada a natureza do algoritmo proposto, no decorrer da busca várias soluções são geradas e avaliadas. Desta forma, é possível escolher, dentre as soluções com uma distância viável entre o centro de gravidade e o ponto exigido, aquela que apresenta melhor ocupação do volume do contêiner. Se nenhuma solução atende a restrição de distribuição de peso, escolhe-se aquela que possui o centro de gravidade mais próximo do esperado.

3.3 Resultados Computacionais

Todos os algoritmos foram implementados em linguagem C++ e compilados com GCC versão 3.3.3 com a opção de otimização –O3. Os testes computacionais foram realizados em um PC Intel Pentium 4 2.8 GHZ com sistema operacional Linux Fedora Core 2.

Os parâmetros do algoritmo de múltiplos inícios proposto são ajustados com o conjunto de instâncias proposto por Bischoff e Ratcliff (1995a), e utilizados em todos os testes computacionais.

Diferentes instâncias da literatura são utilizadas para validar o desempenho do algoritmo de múltiplos inícios. Em todos os experimentos, salvo menção em contrário, o valor de AP é igual a zero, ou seja, é exigido que as caixas carregadas tenham 100% de suporte da base. Os resultados dos testes computacionais são apresentados para cada variante do problema de carregamento de contêiner, conforme segue.

3.3.1 Restrições de estabilidade

Duas versões do algoritmo proposto são apresentadas, uma versão restrita, denominada AAR, e uma versão completa, denominada AA. A primeira versão utiliza poucas iterações, é desprovida de memória e sempre constrói as soluções a partir do conjunto vazio. Com isto, tem-se um algoritmo rápido e eficiente que pode ser inserido em outros algoritmos mais complexos. No entanto, se o desejo é dar mais ênfase à qualidade da solução então o algoritmo completo deve ser utilizado.

Versão Restrita

O número máximo de soluções avaliadas na versão restrita é limitado em 240. Uma função de tendência polinomial $bias(r) = r^{-n}$ é utilizada no cálculo das reduções nos cubóides e na escolha do elemento da LRC. Seja $f(S_{incumbente})$ o valor da solução incumbente e $f(S_{parcial})$ o valor de uma solução parcial. O expoente n é definido conforme segue:

$$n = \begin{cases} n_1 \text{ para } f(S_{parcial}) < r \cdot f(S_{incumbente}) \\ n_2 \text{ caso contrário} \end{cases}$$

em que r = 0,8, $n_1 = 2$ e $n_2 = 3$ para instâncias com menos de oito tipos de caixas e $n_1 = 3$ e $n_2 = 4$ para os demais casos. O tamanho máximo da LRC é τ = 10.

Seis critérios de avaliação são combinados em quatros conjuntos, ver Tabela 3.7. Na avaliação dos cubóides, os critérios são utilizados seqüencialmente, em que o primeiro é o principal e os demais servem como desempate. Nas versões AAR estes conjuntos são escolhidos aleatoriamente em cada iteração.

Tabela 3.7. Conjunto de critérios de avaliação

Conjunto	Critérios
*Critérios*1	C_1, C_2, C_5, C_6
*Critérios*2	C_3, C_4, C_5, C_6
*Critérios*3	C_1, C_2, C_4, C_6
*Critérios*4	C_3, C_2, C_4, C_6

O desempenho do algoritmo mostrou-se pouco sensível quando os valores dos parâmetros variam. Desta forma, um número reduzido de testes computacionais foi necessário para determinar a configuração apresentada acima.

Se a identificação dos espaços vazios ocorre dos fundos para frente do contêiner o algoritmo é identificado como AAR1, e se ocorre simultaneamente dos fundos para frente e da frente para os fundos do contêiner o algoritmo é identificado como AAR2.

Os resultados referentes a AAR1 e AAR2 representam a média de dez execuções realizadas com sementes diferentes para o gerador de números pseudo-aleatórios. O desempenho da heurística de múltiplos inícios é comparado com nove abordagens da literatura, listadas a seguir:

- **BJR** – heurística construtiva (Bischoff et al.,1995)
- **BR** – heurística construtiva (Bischoff e Ratcliff, 1995a)
- **BG_1** – algoritmo genético (Gehring e Bortfeldt,1997)
- **DB** – heurística construtiva (Davies e Bichoff,1999)
- **E** – heurística com busca em árvore (Eley, 2002)
- **CD** – heurística construtiva (Chieng e Deng, 2004)
- **CM** – conjunto de cinco heurísticas construtivas (Cecílio e Morabito, 2004)
- **LZ** – metaheurística *squeaky wheel optimization* (Lim e Zhang, 2005)
- **MO** – GRASP (Moura e Oliveira, 2005)

O código computacional de Cecilio e Morabito (2004) foi utilizado para obter os resultados da Tabela 3.8, que reporta o melhor resultado das cinco heurísticas que compõem CM. Observe que duas das abordagens listadas acima são metaheurísticas (MO e LZ), as demais metaheurísticas que constam na literatura são comparadas com a versão completa do algoritmo.

O conjunto de instâncias proposto por Bischoff e Ratcliff (1995a) compreende sete classes, BR1 a BR7, com 100 instâncias cada. O contêiner possui dimensões $587 \times 233 \times 220$, em centímetros. O número de tipos de caixa em cada uma das sete classes é 3, 5, 8, 10, 12, 15 e 20. Na classe BR1 existem, em média, 50,2 caixas para cada tipo, enquanto na classe BR7 este número cai para 6,51 caixas por tipo. De acordo com a média decrescente de número de caixas por tipo, as instâncias variam gradualmente de fracamente heterogêneas para fortemente heterogêneas. Restrições de orientação proíbem que uma caixa assuma uma posição em que a dimensão que determina a altura seja maior ou igual ao dobro de uma das dimensões da base, ou seja, para uma caixa do tipo t com orientação k tem-se $u_{tk} = 0$ se $d_{3tk} / \left(\min_{j=1,2} d_{jtk} \right) \geq 2$.

A Tabela 3.8 apresenta a média percentual do volume ocupado do contêiner obtido por AAR1, AAR2 e as demais abordagens listadas anteriormente. Os resultados são apresentados em ordem crescente da média geral. O algoritmo AAR2 obtém sistematicamente o melhor resultado dentre todas as abordagens (destaque em negrito) e AAR1 o segundo melhor resultado. O tempo computacional médio para resolver todas as instâncias foi de 0,29 segundos para AAR2 e 0,14 para AAR1. Devido as diferentes plataformas de hardware utilizadas é difícil comparar o tempo computacional com as demais abordagens, mas para propiciar uma base comparativa, o tempo computacional médio para MO é de 33,5 segundos em um Pentium 4 2.4 GHz.

Tabela 3.8. Resultados para instâncias de Bischoff e Ratcliff (1995a)

Método	Instâncias							Média
	BR1	BR2	BR3	BR4	BR5	BR6	BR7	
BJR	81,76	81,70	82,98	82,60	8276	81,50	80,51	81,97
BR	83,79	84,44	83,94	83,71	83,80	82,44	82,01	83,45
DB	84,10	84,50	85,00	84,70	84,60	83,70	82,70	84,19
BG_1	85,80	87,26	88,10	88,04	87,86	87,85	87,68	87,51
CM	89,05	87,40	87,21	86,75	87,09	86,05	84,82	88,34
E	88,05	88,44	89,23	89,24	88,99	88,91	88,36	88,75
MO	89,07	90,43	90,86	90,42	89,57	89,71	88,05	89,07
LZ	87,4	88,7	89,3	89,7	89,7	89,7	89,4	89,13
AAR1	90,86	90,88	90,94	90,67	90,40	90,14	89,46	90,48
AAR2	**91,73**	**91,60**	**91,47**	**91,06**	**90,90**	**90,46**	**89,54**	**90,96**

Bischoff e Ratcliff (1995a) propõem duas medidas para avaliar a estabilidade de um carregamento. A primeira medida, denominada Medida 1, fornece o número médio de caixas que proporcionam suporte para caixas que não estão sobre o piso do contêiner (quanto maior melhor). Alternativamente é proposta a Medida 1a, uma extensão da Medida 1 que descarta áreas de contatos inferiores a 5% da área da face superior da caixa suporte. A segunda medida, denominada Medida 2, fornece a média percentual de caixas que não possuem três faces laterais com apoio (quanto menor melhor).

A Tabela 3.9 apresenta os resultados comparativos relativos à estabilidade do carregamento de seis abordagens que constam na literatura juntamente com AAR1 e AAR2. Os melhores resultados aparecem em negrito. O algoritmo BJR, projetado para obter padrões de carregamento estáveis no que se refere à Medida 1, obtém os melhores resultados para esta medida. O algoritmo AAR1 apresenta os melhores resultados para a Medida 2, muito provavelmente devido ao bom desempenho na ocupação do volume do contêiner. Os resultados de AAR2 são ligeiramente superiores para a Medida 1 e significativamente inferiores para a Medida 2 quando comparados com AAR1. Esta diferença é devida ao procedimento de carregar o contêiner em duas direções utilizado por AAR2, que resulta em espaços vazios entre as duas frentes de carregamento.

A Tabela 3.10 apresenta resultados comparativos com a heurística proposta por Chien e Deng (2004) para onze instâncias reais de empresas de transporte de Taiwan, em que as caixas podem assumir qualquer uma das 6 possíveis orientações. A versão AAR2 consegue os melhores resultados para todas as instâncias com um tempo computacional máximo inferior a 0,3 segundos. Excepcionalmente, a versão AAR1,

considerando as 10 execuções, obtém uma solução com 100% de ocupação do volume do contêiner para as instâncias 9 e 10. A versão AAR2 obtém três soluções com 100% de ocupação do volume do contêiner para a instância 10. A Figura 3.15 apresenta graficamente as soluções ótimas das instâncias 9 e 10.

Resultados complementares para AAR1 e AAR2 são inclusos no apêndice deste trabalho.

Tabela 3.9. Resultados estabilidade do carregamento, instâncias de Bischoff e Ratcliff(1995a)

Métodos	Instâncias													
	BR1		BR2		BR3		BR4		BR5		BR6		BR7	
	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
BJR	**2,02**	8,50	**2,22**	11,21	**2,20**	15,93	**2,10**	17,51	**2,09**	21,60	**2,04**	22,13	**1,92**	27,07
BR	1,13	10,36	1,10	14,60	1,08	19,67	1,07	23,53	1,06	26,03	1,06	31,04	1,04	35,99
BG_1	-	11,00	-	16,00	-	18,50	-	21,50	-	22,50	-	25,00	-	28,50
CM	1,14	7,57	1,12	10,75	1,10	13,72	1,10	14,99	1,10	16,50	1,10	19,58	1,10	21,76
E	-	9,80	-	13,50	-	18,00	-	20,50	-	21,50	-	22,90	-	26,00
MO	1,07	11,53	1,10	12,67	1,09	17,75	1,10	20,03	1,10	22,75	1,10	26,50	1,11	28,86
AAR1	1,15	**6,00**	1,15	**9,22**	1,12	**10,35**	1,11	**12,48**	1,11	**13,70**	1,10	**15,70**	1,08	**18,24**
AAR2	1,18	10,77	1,18	13,72	1,15	16,17	1,13	18,09	1,13	19,51	1,12	20,91	1,10	23,91

Tabela 3.10. Resultados para as instâncias de Chien e Deng (2004)

Métodos	Instâncias										
	1	2	3	4	5	6	7	8	9	10	11
CD	83,08	95,22	83,43	87,68	80,49	81,16	91,30	90,55	95,55	93,08	92,02
AAR1	83,94	97,26	93,29	98,66	**83,04**	**85,25**	99,56	**92,40**	98,09	99,57	97,57
AAR2	**83,94**	**97,32**	**94,73**	**98,64**	**83,04**	**85,25**	**99,74**	**92,40**	**99,16**	**99,82**	**97,81**

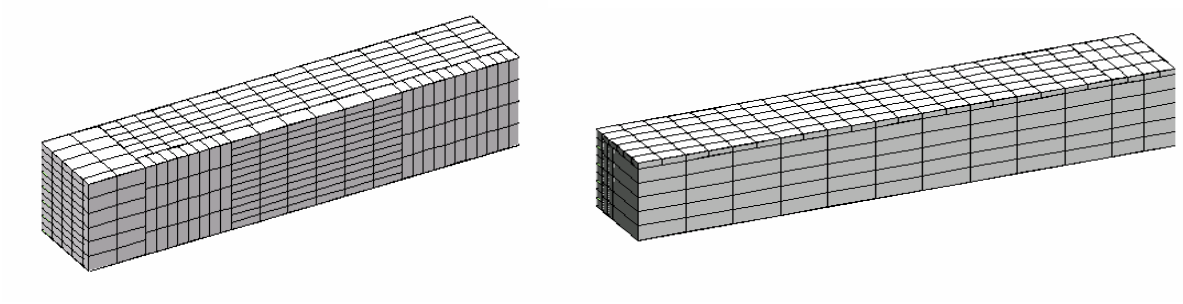


Figura 3.15. Soluções ótimas para as instâncias 9 e 10 de Chien e Deng (2004)

Versão Completa

A versão do algoritmo de múltiplos inícios proposto que utiliza memória e diferentes graus de reconstrução é denominado AA1 ou AA2, dependendo da forma como são identificados os espaços vazios. Nesta versão, os conjuntos de critérios para escolha dos cubóides são escolhidos com o mecanismo adaptativo proposto por Prais e Ribeiro (2000).

Os parâmetros em comum com a versão restrita assumem os mesmos valores e os demais, também calibrados com as instâncias de Bischoff e Ratcliff (1995a), são exibidos na Tabela 3.11.

Tabela 3.11. Valores dos parâmetros para os algoritmos AA1 e AA2

Parâmetro	Valor
K	100
f	10
$emax$	10
λ	0,8
$freq\_gr$	400
max_vp	0,8
max_c	1
$Iter\_parcial$	50

Os valores dos três primeiros parâmetros listados na Tabela 3.11 foram determinados de acordo com o utilizado em trabalhos da literatura (Prais e Ribeiro, 2000 e Armentano e Araújo, 2006). Para os demais parâmetros os valores testados são os que seguem. Para λ 0,2; 0,5; 0,8; 1, para $Freq\_gr$ 50, 100, 200, 400, 800, para max_vp 0,3; 0,5; 0,8; 1, para max_c 1, 2, 3 e para $iter\_parcial$ 10, 50, 100.

As Figuras 3.16 e 3.17 apresentam a contribuição dos diferentes componentes no desempenho do algoritmo proposto para as instâncias BR1-BR7. Para ambos os tipos de identificação de espaços vazios, a memória e grau de reconstrução são acrescidos separadamente à versão restrita. Deste modo, AAM acresce os procedimentos relativos a memória ao método restrito AAR, AAG acresce os procedimentos relativos ao uso de diferentes graus de reconstrução e AA é o algoritmo com todos os componentes. Todos os resultados correspondem à melhor solução encontrada em 10.000 soluções avaliadas. A memória possui uma atuação mais acentuada nas instâncias fracamente heterogêneas, enquanto o grau de reconstrução é mais influente para as instâncias fortemente heterogêneas. Em instâncias fracamente heterogêneas existe mais possibilidade de gerar

cubóides de tamanhos distintos, daí a atuação mais acentuada da memória. O algoritmo completo obtém o melhor desempenho para todas as classes de instâncias.

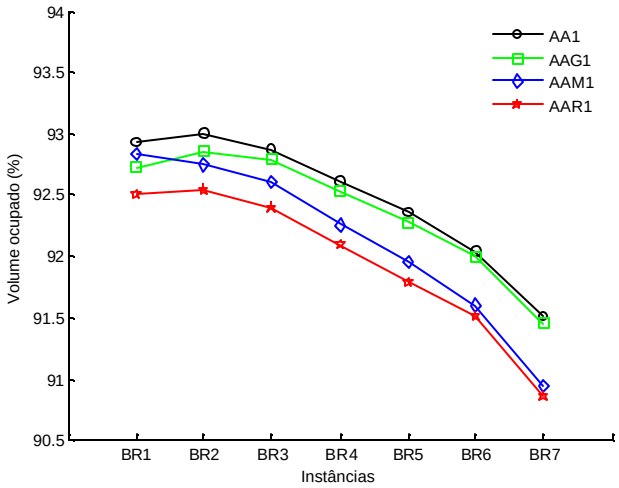


Figura 3.16. Contribuição dos diferentes componentes no desempenho do algoritmo AA1

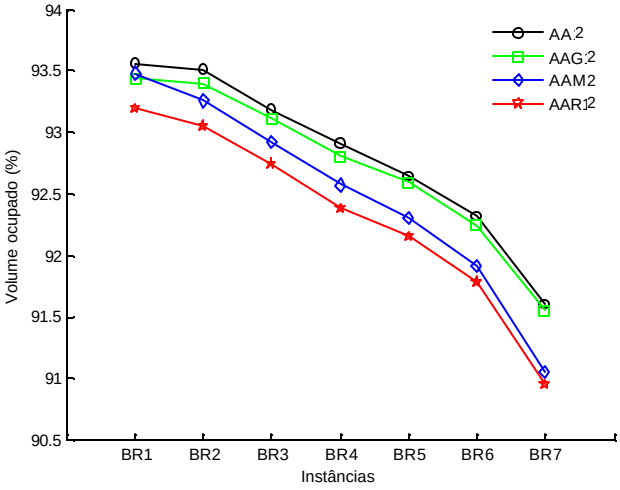


Figura 3.17. Contribuição dos diferentes componentes no desempenho do algoritmo AA2

As seguintes abordagens encontradas na literatura científica são utilizadas nos testes comparativos de desempenho das versões AA1 e AA2:

- **BG_2** – busca tabu (Bortfeldt Gehring, 1998)
- **BG_3** – algoritmo genético híbrido (Bortfeldt e Gehring, 2001)

- **GB** – algoritmo genético paralelo (Gehring e Bortfeldt, 2002)
- **BGM_1** – busca tabu sequencial (Bortfeldt et al., 2003)
- **BGM_2** – busca tabu paralela(Bortfeldt et al., 2003)
- **MGB_1** – metaheurística híbrida (Mack et al., 2004)
- **B** – heurística de múltiplos inícios (Bischoff, 2006)

A Tabela 3.12 apresenta os resultados de AA1, AA2 e das abordagens citadas acima para as instâncias BR1 – BR7. Os resultados dos algoritmos AA1 e AA2 referem-se a 10.000 soluções avaliadas. Os autores de BGM_1, BGM_2 e MGB_1 só reportam o desempenho médio de todas as instâncias agrupadas. A versão AA2 apresenta o melhor desempenho e a versão AA1 o segundo melhor desempenho para todas as classes de instâncias. O tempo computacional médio total é de 10,45 segundos para AA2 e 6,01 segundos para AA1. Os autores de MBG_1 reportam um tempo computacional médio de 205 segundos em um Pentium 2 GHz.

Os resultados de B referem-se à avaliação de 1.000 soluções. Utilizando o mesmo limite de soluções avaliadas para AA2, a média geral passa a ser 91,84% superando B em todas as classes de instâncias, com exceção da classe BR7 na qual AA2 obtém 90,49% de ocupação do contêiner. Neste caso, o tempo computacional médio de AA2 é de 0,95 segundos e para B em torno de 3,5 minutos em um Pentium 4 1.7 GHz.

Tabela 3.12. Resultados adicionais para instâncias de Bischoff e Ratcliff (1995a)

Métodos	Instâncias							Média
	BR1	BR2	BR3	BR4	BR5	BR6	BR7	
BG_3	87,81	89,40	90,48	90,63	90,73	90,72	90,65	90,06
GB	88,10	89,56	90,77	91,03	91,23	91,28	91,04	90,43
BG_2	92,41	92,33	91,97	91,26	90,40	89,57	88,18	90,87
B	90,57	90,84	91,43	91,21	91,25	91,04	90,81	91,02
BGM_1	-	-	-	-	-	-	-	91,60
BGM_2	-	-	-	-	-	-	-	92,20
MBG_1	-	-	-	-	-	-	-	92,41
AA1	92,93	93,00	92,86	92,60	92,35	92,04	91,50	92,47
AA2	**93,57**	**93,51**	**93,19**	**92,91**	**92,63**	**92,32**	**91,60**	**92,82**

A Tabela 3.13 apresenta os resultados quando AP assume valor 0,45, ou seja, até 45% de projeção da base de apoio é permitida. Os algoritmos AA1 e AA2 avaliam no máximo 30.000 soluções. A abordagem MGB_2 é a versão paralela de MGB_1. É

possível observar que AA2 apresenta o melhor resultado médio e os melhores resultados para cada classe de instâncias, com exceção da classe BR3. O tempo computacional médio para resolver todas as instâncias é de 59,78 segundos para AA1, 129,61 segundos para AA2 e 222 segundos para MGB_2 em um Pentium 2 GHz.

Tabela 3.13. Resultados para instâncias de Bischoff e Ratcliff (1995a) com restrições de projeção da base de apoio

Métodos	Instâncias							Média
	BR1	BR2	BR3	BR4	BR5	BR6	BR7	
MGB_1	93,26	93,56	93,71	93,30	92,78	92,20	91,20	92,86
MGB_2	93,41	93,82	**94,02**	93,68	93,18	92,64	91,68	93,20
AA1	93,63	93,53	93,50	93,14	92,95	92,64	92,12	93,07
AA2	**94,37**	**94,36**	94,00	**93,72**	**93,49**	**93,15**	**92,49**	**93,65**

3.3.2 Restrições de empilhamento

Para tratar as restrições de empilhamento, além das adaptações expostas na seção 3.2.6, uma nova seleção de conjuntos para a avaliação dos cubóides é utilizada, conforme apresentado na Tabela 3.14. Nos dois primeiros conjuntos, o critério utilizado para diminuir a complexidade da frente de carregamento (C_5) é substituído pelo critério que considera a perda relativa na capacidade de empilhamento (C_7), uma vez que o último critério considera as restrições de empilhamento e os padrões de carregamento para este problema tendem a ser mais complexos. Especificamente no primeiro conjunto, a substituição do critério que favorece a construção de superfícies planas (C_2) pelo critério que avalia a área da base dos cubóides (C_4) procura favorecer a geração de superfícies de carregamento mais homogêneas no que se refere à capacidade de empilhamento.

Tabela 3.14. Conjunto de critérios de avaliação dos cubóides - restrições de empilhamento

Conjunto	Critérios
*Critérios*1E	C_1, C_4, C_7, C_6
*Critérios*2E	C_3, C_4, C_7, C_6
*Critérios*3	C_1, C_2, C_4, C_6
*Critérios*4	C_3, C_2, C_4, C_6

O conjunto de instâncias gerado por Bischoff (2006), que acresce um peso e uma capacidade de empilhamento às caixas definidas em BR1-BR7, é utilizado nos testes apresentados a seguir. O peso de cada caixa é proporcional ao seu volume, ou seja, assume-se que todas as caixas possuem a mesma densidade.

Na Tabela 3.15, a abordagem proposta por Bischoff (2006), identificada por B, é comparada com as versões AA1E e AA2E do algoritmo proposto adaptado para tratar restrições de empilhamento. Para B, AA1E e AA2E é imposto o limite de 1.000 soluções avaliadas, enquanto AA1E* e AA2E* utilizam o limite de 60.000 soluções avaliadas. Verifica-se que AA1 apresenta melhor desempenho para BR1 e AA2 para BR1 e BR2 em comparação com B. Quando mais soluções são avaliadas, AA1E* e AA2E* obtêm os melhores resultados para todas as classes de instâncias, com de BR7.

Tabela 3.15. Resultados para instâncias de Bischoff e Ratcliff (1995a) com restrições de empilhamento

Métodos	Instâncias							Média
	BR1	BR2	BR3	BR4	BR5	BR6	BR7	
B	80,44	83,70	85,84	85,84	85,95	86,04	**86,30**	84,87
AA1E	80,66	83,66	85,07	84,47	84,16	84,12	83,69	83,69
AA2E	80,89	84,18	85,32	84,88	84,29	84,35	83,61	83,93
AA1E*	82,64	86,04	87,25	86,74	86,44	86,35	85,86	85,90
AA2E*	**82,68**	**86,29**	**87,54**	**87,02**	**86,78**	**86,70**	85,99	**86,14**

A Tabela 3.16 apresenta os tempos computacionais, em segundos, para os resultados apresentados na Tabela 3.15. Bischoff (2006) relata a média geral para resolver as instâncias BR1-BR7 em um Pentium 4 1.7 GHz com 256 Mb de RAM. Apesar da diferença dos computadores, não é difícil inferir que B tem um tempo computacional superior às versões AAR1E* e AAR2E*. É possível justificar essa diferença devido a B construir o padrão de carregamento do contêiner caixa a caixa, enquanto os outros métodos utilizam cubóides.

Tabela 3.16. Tempo computacional, em segundos, para instâncias de Bischoff e Ratcliff (1995a) com restrições de empilhamento

Métodos	Instâncias							Média
	BR1	BR2	BR3	BR4	BR5	BR6	BR7	
B	-	-	-	-	-	-	-	♦210,00
AA1E	0,31	0,66	0,97	1,44	1,92	2,78	4,38	1,78
AA2E	0,56	1,21	1,83	2,71	3,76	5,58	8,88	3,51
AA1E*	19,51	40,59	60,30	88,03	116,64	170,59	266,94	105,71
AA2E*	29,75	64,45	96,69	143,24	197,57	295,07	470,08	185,26

♦ Pentium 4 1.7 GHz com 256 Mb de RAM

♠ Pentium 4 2.8 GHZ com 512 Mb de RAM

As Tabelas 3.17 e 3.18 apresentam os resultados quando reduções nas capacidades de empilhamento utilizadas anteriormente são consideradas. Na primeira

tabela a capacidade de empilhamento é reduzida à metade e na segunda à quarta parte. Conforme esperado, os resultados demonstram que quanto menor a capacidade de empilhamento menor é a ocupação do volume do contêiner. Considerando o limite de 1.000 soluções avaliadas, AA1E e AA2E superam B nas classes de instâncias fracamente heterogêneas (BR1 a BR4) quando a capacidade de empilhamento é reduzida à metade. Para uma redução à quarta parte, AA1E e AA2E superam B em todas as classes de instâncias, com exceção de BR5 e BR7. Quando um número maior de soluções são avaliadas, AA1E* e AA2E* apresentam os melhores resultados para ambos os casos estudados.

Tabela 3.17. Resultados para instâncias de Bischoff e Ratcliff (1995a) com restrições de empilhamento reduzidas à metade

Métodos	Instâncias							Média
	BR1	BR2	BR3	BR4	BR5	BR6	BR7	
B	61,41	67,67	71,00	72,35	72,35	73,22	72,77	70,11
AA1E	63,61	68,59	71,53	72,38	71,96	72,73	71,53	70,33
AA2E	63,69	68,78	71,56	72,66	72,21	72,92	71,65	70,50
AA1E*	**65,10**	**70,97**	73,94	75,31	74,96	75,65	74,60	72,93
AA2E*	65,04	70,96	**74,06**	**75,49**	**75,09**	**75,74**	**74,85**	**72,96**

Tabela 3.18. Resultados para instâncias de Bischoff e Ratcliff (1995a) com restrições de empilhamento reduzidas à quarta parte

Métodos	Instâncias							Média
	BR1	BR2	BR3	BR4	BR5	BR6	BR7	
B	43,95	49,69	52,38	53,85	54,58	54,86	55,01	52,05
AA1E	46,22	51,17	53,32	53,99	54,21	54,90	54,01	52,55
AA2E	46,14	51,12	53,39	54,21	54,54	55,01	54,19	52,66
AA1E*	**46,95**	**52,12**	**54,75**	55,88	56,26	56,81	56,04	54,12
AA2E*	46,87	52,09	**54,75**	**56,02**	**56,47**	**56,84**	**56,31**	**54,19**

3.3.3 Restrições de distribuição de peso

Como não existem instâncias disponíveis na literatura para este problema, o conjunto de testes foi gerado de acordo com a proposta de Davies e Bischoff (1999). Nesta metodologia, os itens das instâncias BR1-BR7 pertencem a duas categorias, ou são muito leves ou muito pesados. A densidade de cada tipo de caixa é escolhido, com igual probabilidade, nos intervalos 0,01-0,05 g/cm^3 ou 0,96-1,00 g/cm^3, de acordo com uma distribuição uniforme. Os autores observam que a motivação deste método é gerar

instâncias que sejam desafiadoras do ponto de vista de distribuição de peso, e não necessariamente a configuração resultante reflete o que ocorre em problemas reais.

Para resolver este problema é utilizada a versão restrita do algoritmo de múltiplos inícios (AAR1 e AAR2). As restrições de distribuição são interpretadas de quatro formas, conforme explicado a seguir:

1. Livre – as restrições de distribuição de peso não são consideradas, os resultados para o volume ocupado são iguais aos apresentados na Tabela 3.8.
2. **1% (*x*)** – é exigido que a distância da coordenada x do centro de gravidade em relação ao ponto médio não ultrapasse 1% do comprimento do contêiner. Como o comprimento do contêiner considerado é de 587 cm, a distância da coordenada x do centro de gravidade em relação ao ponto médio do comprimento do contêiner deve ser menor que 5,87 cm, ou ficar o mais próximo possível deste valor.
3. **1% (*y*)** – semelhante à restrição anterior, mas considerando a largura do contêiner. Neste caso, a distância da coordenada y do centro de gravidade em relação ao ponto médio da largura do contêiner deve ser menor ou o mais próximo possível de 2,33 cm, uma vez que o contêiner considerado possui 233 cm de largura.
4. **1% (*x*,*y*)** – as duas restrições descritas anteriormente são consideradas em conjunto.

A Tabela 3.19 apresenta os resultados de AAR1 e AAR2 para os quatro tipos de restrições de distribuição de peso considerados. Para as restrições denominadas Livre, verifica-se que a distância do centro de gravidade em relação ao ponto médio do contêiner nas duas direções horizontais não ultrapassa 10% da correspondente dimensão**.** A única exceção é o centro de gravidade médio obtido por AAR1 ao longo do eixo x da classe BR2. Este resultado já é suficiente para atender a maioria das aplicações práticas e revela que é possível obter soluções com boa ocupação de volume do contêiner associadas a uma boa distribuição de peso.

Ainda na Tabela 3.19, para as restrições 1% (*x*) e 1% (*y*) a degradação do volume ocupado não excede 1,83 %, sendo mais acentuada para a restrição do tipo

1% (x). Quando é considerada a restrição 1% (x,y), a degradação do volume ocupado é de 4,65 % e 4,46 % para AAR1 e AAR2 , respectivamente. Em geral, o algoritmo AAR2 apresenta resultados superiores para todas as medidas. Esse comportamento pode ser atribuído a diferença na forma de identificar os espaços vazios utilizado em AAR1 e AAR2. Também é possível verificar que a distribuição de peso para as instâncias mais heterogêneas é de melhor qualidade. Este fato pode ser facilmente explicado lembrando que nestas instâncias existe um maior número de tipos de caixas, o que promove um maior número de opções para distribuir o peso no carregamento.

Tabela 3.19. Resultados para instâncias de Bischoff e Ratcliff (1995a) com restrições de distribuição de peso

Versão	Restrição	Medida	Instâncias							Média
			BR1	BR2	BR3	BR4	BR5	BR6	BR7	
AAR1	Livre	V (%)	90,86	90,88	90,94	90,67	90,40	90,14	89,46	90,48
		x (cm)	50,72	61,63	54,68	46,75	40,57	42,08	36,92	47,62
		y(cm)	13,60	14,86	16,19	14,90	14,35	12,60	12,00	14,07
	1% (x)	V (%)	88,68	88,55	88,96	88,97	88,87	88,68	87,91	88,66
		x (cm)	3,22	3,06	3,07	3,04	2,95	2,87	3,00	3,03
		y(cm)	14,99	17,00	15,87	16,38	14,45	14,15	12,43	15,04
	1% (y)	V (%)	89,26	89,52	89,94	89,66	89,38	89,22	88,55	89,36
		x (cm)	55,69	69,34	60,27	50,65	42,96	45,97	37,34	51,74
		y(cm)	1,62	1,39	1,33	1,25	1,29	1,22	1,23	1,33
	1% (x, y)	V (%)	84,99	84,75	86,44	86,25	86,41	86,40	85,56	85,83
		x (cm)	3,60	3,41	3,44	3,30	3,05	3,13	3,14	3,30
		y(cm)	7,65	7,12	6,68	6,18	5,24	5,24	3,97	6,01
AAR2	Livre	V (%)	91,73	91,60	91,47	91,06	90,90	90,46	89,54	90,96
		x (cm)	45,51	56,84	52,17	45,37	43,90	44,26	40,47	46,93
		y(cm)	14,68	17,26	17,70	16,70	14,30	13,90	11,57	15,16
	1% (x)	V (%)	89,61	89,36	89,29	89,30	89,26	88,88	88,22	89,13
		x (cm)	3,08	2,96	2,85	3,01	2,92	2,96	2,92	2,96
		y(cm)	16,80	19,63	17,58	15,61	14,80	14,30	12,45	15,88
	1% (y)	V (%)	90,20	89,99	89,77	89,58	89,51	89,25	88,56	89,55
		x (cm)	52,68	64,13	55,68	48,76	45,86	45,37	38,66	50,16
		y(cm)	1,20	1,27	1,33	1,28	1,26	1,20	1,21	1,25
	1% (x, y)	V (%)	86,80	85,96	86,54	86,76	86,66	86,50	86,22	86,49
		x (cm)	3,33	3,26	3,25	3,08	3,01	2,93	2,85	3,10
		y(cm)	5,41	5,52	6,68	5,39	4,55	4,38	3,34	5,04

Capítulo 4

Problema *bin packing* 3D

4.1 Introdução

O problema *bin packing* 3D abordado neste capítulo concentra-se na questão de como carregar um conjunto de caixas de tamanhos diferentes em um número mínimo de contêineres idênticos. Este problema é NP-difícil no sentido forte e extremamente trabalhoso de resolver na prática (Martello et al., 2000). Para minimizar o número de *bins* requerido, muitas heurísticas propostas na literatura resolvem repetidos problemas de carregamento de um único contêiner. Deste modo, pode-se concluir que um elemento importante em heurísticas para o problema *bin Packing* 3D é o procedimento que carrega as caixas, no entanto, o projeto do procedimento que supervisiona a utilização dos *bins* tem importância capital. Os termos *bin* e contêiner são utilizados indistintamente.

Uma vez definido o algoritmo para carregamento de contêiner, duas estratégias podem ser facilmente implementadas para a resolução do problema *bin packing* 3D. Estas estratégias podem ser definidas conforme segue:

- Estratégia seqüencial: os contêineres são preenchidos um após o outro em seqüência.
- Estratégia paralela: um número dado de contêineres é preenchido ao mesmo tempo, os espaços vazios de todos os contêineres são considerados em uma única lista e o espaço vazio com melhor avaliação recebe itens para carregamento. O número de contêineres deve ser definido *a priori*.

As estratégias seqüencial e paralela são extensões naturais do algoritmo de carregamento de contêiner, mas são demasiadas simples para apreender os elementos necessários para construir um algoritmo robusto para o problema *bin packing* 3D.

Neste trabalho é proposto um método para resolução do problema *bin packing* 3D inspirado em descida em vizinhança variável (DVV), conhecido na língua inglesa como *variable neighborhood search*. As estruturas de vizinhança utilizadas são semelhantes às de Lodi et al. (2004). Estas vizinhanças definem subproblemas que são resolvidos com estratégia seqüencial e paralela utilizando o algoritmo para carregamento de contêiner apresentado no Capítulo 3.

O método de DVV, proposto nos trabalhos de Mladenovic e Hansen (1997) e Hansen e Mladenovic (1999, 2001, 2003), é uma busca local que explora o espaço de soluções de uma estrutura de vizinhança com tamanho variável. Para construir as vizinhanças e realizar uma busca sistemática, faz-se necessário definir uma métrica no espaço de soluções de modo a possibilitar que soluções de boa qualidade "distantes" da solução atual sejam encontradas. Isto habilita o método a escapar de ótimos locais com respeito a uma vizinhança de menor tamanho. A Figura 4.1 mostra um pseudocódigo para o método DVV. Nesta figura, na linha 1 é feita a seleção das estruturas de vizinhança e na linha 2 a variável kn, que determina o índice da vizinhança utilizada, é inicializada com valor 1. Uma solução inicial é obtida na linha 3. O laço nas linhas 4 a 12 é iterado até que todas as vizinhanças sejam analisadas sem sucesso. Na linha 5 é aplicada a busca local com a vizinhança de índice kn. Sempre que uma solução de melhor qualidade é obtida a busca recomeça com a primeira vizinhança, caso contrário, a próxima vizinhança é examinada, ver linhas 6 a 11.

Na próxima seção é detalhado o algoritmo para o problema *bin packing* 3D inspirado em DVV.

```
    Procedimento descida em vizinhança variável
1   Selecione o conjunto de estruturas de vizinhança kn = 1, 2,..., kn_max
2   kn ← 1
3   Construa uma solução inicial x
4   Repita
5       Aplique busca local com a vizinhança kn para melhorar x e seja x' a solução obtida
6       Se x' é melhor que x então
7           x ← x'
8           kn ← 1
9       Senão
10          kn ¬ kn + 1
11      Fim_Se
12  Até que kn > kn_max
13  Retorna x
```

Figura 4.1. Pseudocódigo para o método de descida em vizinhança variável

4.2 Algoritmo de descida em vizinhança variável para o problema *bin packing* 3D

No algoritmo proposto, dada uma solução, um *bin* alvo é escolhido e desmontado de forma a inicializar o conjunto B de caixas não carregadas. Cada vizinhança considera o conjunto B e todas as kn-túplas dos *bins* remanescentes, em que kn é o parâmetro que determina o tamanho da vizinhança. Um movimento corresponde à resolução de um subproblema do *bin packing* 3D original, no qual se busca carregar as caixas de uma kn-túpla mais as caixas do conjunto B em no máximo kn *bins*. Sempre que um movimento diminui o volume do conjunto de caixas não carregadas, este é imediatamente executado e o processo reinicia com a vizinhança de índice 1. Caso contrário, o processo continua com a vizinhança de índice $kn+1$.

Os subproblemas gerados pelas kn-vizinhanças são resolvidos com o algoritmo para carregamento de contêiner apresentado no capítulo 3 com estratégia paralela e sequencial.

A Figura 4.2 apresenta o pseudocódigo do procedimento descida em vizinhança variável para *bin packing* 3D. Dados da instância e parâmetros do algoritmo são processados na linha 1. Nas linhas 2 a 6 uma solução inicial é calculada. Em primeiro lugar uma solução é obtida com a estratégia seqüencial e então é atribuída à solução incumbente. A variável $maxvol$ é inicializada com o volume ocupado do *bin* mais carregado da solução incumbente. Uma vez determinado o número máximo de *bins* necessário para carregar todo o conjunto de caixas com a estratégia sequencial, outra solução é calculada com a estratégia paralela. Em seguida, a solução incumbente e a variável $maxvol$ são atualizadas. Na linha 7 o conjunto B recebe as caixas do *bin* com menor volume ocupado (*bin* alvo). Na linha 8 o tamanho da vizinhança é inicializado com valor 1 e $\overline{B}$ recebe o conjunto B. O laço principal, exibido nas linhas 9 a 28, termina quando todas as vizinhanças são analisadas, o número de *bins* da $S_{incumbente}$ é igual ao limitante inferior ou um terceiro critério de parada é satisfeito, usualmente tempo computacional. O limitante inferior é trivialmente calculado como o menor inteiro maior que o quociente entre o volume total das caixas e o volume de um *bin*. As caixas do conjunto B e da kn-túpla definem um subproblema que só é resolvido se houver a possibilidade de se obter uma solução cujo volume ocupado médio seja menor ou igual à $maxvol$ ou a vizinhança em uso for a de índice 1, ver linha 11. Todos os subproblemas da vizinhança de índice 1 são considerados na intenção de minimizar o volume do conjunto B com pouco esforço computacional. O procedimento para resolver os subproblemas é invocado na linha 12, na qual S é uma solução com no máximo kn *bins* e $\overline{B}$ o conjunto de caixas não carregadas após a resolução do subproblema em foco. Ainda na linha 12, os algoritmos que resolvem os subproblemas não são determinísticos e os parâmetros $iter\_p$ e $iter\_s$ são utilizados para limitar o número de soluções avaliadas. Se o conjunto de caixas $\overline{B}$ tem volume total menor que B (linha 10), o fluxo do algoritmo é desviado para a linha 16. Se $\overline{B}$ é vazio a solução incumbente e a variável $maxvol$ são atualizadas, um novo *bin* alvo é escolhido e $\overline{B}$ recebe o conjunto B, caso contrário, o conjunto B recebe $\overline{B}$. Em qualquer caso, o índice da vizinhança é inicializado com valor 1 e o conjunto $\overline{B}$ recebe B, ver linhas 16 a 23. Se nenhum conjunto $\overline{B}$ obtém melhor avaliação que B, o tamanho da vizinhança é incrementado, linha 25. Ao final a melhor solução gerada, $S_{incumbente}$, é retornada.

	Procedimento descida em vizinhança variável para *bin packing* 3D
1	Leia e processa os dados da instância
2	$S \leftarrow$ calcula uma solução inicial com estratégia sequencial
3	$S_{incumbente} \leftarrow S$
4	*maxvol* $\leftarrow$ volume ocupado do *bin* mais carregado da $S_{incumbente}$
5	$S \leftarrow$ calcula uma solução inicial com estratégia paralela com o número de *bins* da $S_{incumbente}$
6	**Se** todas as caixas empacotadas **então** $S_{incumbente} \leftarrow S$; atualiza *maxvol*; **Fim_Se**
7	$B \leftarrow$ conjunto de caixas do *bin* com menor volume carregado (*bin* alvo)
8	$kn \leftarrow 1$; $\overline{B} \leftarrow B$
9	**Enquanto** $(kn \leq kn_{\max})$ **E** (número de *bins* da $S_{incumbente}$ > limitante inferior) **E** (critério de parada não satisfeito) f**aça**
10	**Enquanto** (existir kn-túplas não incluindo o *bin* alvo) **E** $(V(\overline{B}) \geq V(B))$ f**aça**
11	**Se** $(V(B)+V(kn-\text{túpla}) < kn \cdot maxvol)$ **OU** $(kn=1)$ **então**
12	Resolva com estratégia paralela (*iter_p*) e sequencial (*iter_s*) o subproblema induzido por B e kn-túpla e encontre a solução S e o respectivo conjunto de caixas não carregadas $\overline{B}$.
13	**Fim_Se**
14	**Fim_Enquanto**
15	**Se** $(V(\overline{B}) < V(B))$ **então**
16	**Se** $\overline{B} = \{\}$ **então**
17	Atualiza $S_{incumbente}$ e *maxvol*
18	$B \leftarrow$ conjunto de caixas do *bin* com menor volume carregado (*bin* alvo)
19	**Senão**
20	$B \leftarrow \overline{B}$
21	**Fim_Se**
22	$\overline{B} \leftarrow B$
23	$kn \leftarrow 1$
24	**Senão**
25	$kn \leftarrow kn+1$
26	**Fim_Se**
27	**Fim_Enquanto**
28	**Retorna (**$S_{incumbente}$**)**

Figura 4.2. Algoritmo de busca em vizinhança variável para o problema *bin packing* 3D

Quando o problema possui restrições de múltiplos destinos, a solução inicial é obtida pelo carregamento das demandas em ordem não crescente de volume. No carregamento de cada *bin*, sempre que uma demanda não pode ser carregada por inteiro, tenta-se carregar a próxima na seqüência, e assim por diante. Uma vizinhança definida por movimentos de troca de pares adjacentes é explorada quando os subproblemas são resolvidos. Esta vizinhança considera todas as possíveis trocas entre pares de demandas que sejam consecutivos na seqüência de carregamento da solução atual. Por exemplo, suponha que a seqüência de carregamento da solução atual é 1,2,3,4 (isto significa que a primeira demanda a ser carregada é a 1, seguida da demanda 2, depois a 3 e por fim a demanda 4). Os vizinhos são **2,1,**3,4 ; 1,**3,2,**4 e 1,2,**4,3,**. O tamanho da vizinhança é $n-1$. É importante observar que neste caso a identificação dos espaços vazios acontece sempre dos fundos para frente do contêiner.

Uma vez que os algoritmos de carregamento de contêiner com estratégia paralela e sequencial não são determinísticos, o algoritmo da Figura 4.2 pode ser iniciado várias vezes de forma a propiciar diferentes configurações de empacotamento na solução inicial e nos subproblemas. Neste caso, a solução incumbente passa a ser a melhor solução gerada em todo o processo de busca. Além dos parâmetros dos algoritmos para carregamento de contêiner com estratégia paralela e sequencial, os seguintes parâmetros devem ser definidos:

Tabela 4.1. Descrição dos parâmetros do algoritmo de descida em vizinhança variável

Parâmetro	Descrição
$iter\_i$	número máximo de inícios
kn_{max}	determina o tamanho máximo das vizinhanças
$iter\_p$	número máximo de soluções avaliadas com estratégia paralela na resolução dos subproblemas em cada vizinhança
$iter\_s$	número máximo de soluções avaliadas com estratégia sequencial na resolução dos subproblemas em cada vizinhança

4.3 Resultados computacionais

Todos os algoritmos foram implementados em linguagem C++ e compilados com GCC versão 3.3.3 com a opção de otimização –O3. Os testes computacionais foram realizados em um PC Intel Pentium 4 2.8 GHz com sistema operacional Linux Fedora Core 2. O algoritmo proposto é denominado AA1bp se a identificação dos espaços vazios ocorre dos fundos para frente do contêiner, e AA2bp se ocorre simultaneamente dos fundos para frente e da frente para os fundos do contêiner.

Os parâmetros dos algoritmos de carregamento de contêiner com estratégia sequencial e paralela para resolver os subproblemas foram ajustados com as instâncias BR1-BR7. A Tabela 4.2 sumariza os parâmetros e seus respectivos valores, que são os mesmos utilizados nos testes computacionais do capítulo 3.

Tabela 4.2. Valores dos parâmetros do algoritmo para carregamento de contêiner

Parâmetro	Valor
K	100
f	10
emax	10
λ	0,8
r	0,8
t	10
freq_gr	400
max_vp	0,8
max_c	1
iter_parcial	50

Os parâmetros da função de tendência utilizada nos cálculos das reduções dos cubóides e na escolha do elemento da LRC são definidos de acordo com o número de tipos de caixas. Para instâncias com menos de oito tipos de caixas, tem-se $n_1 = 2$ e $n_2 = 3$ e para os demais casos $n_1 = 3$ e $n_2 = 4$.

Poucos testes computacionais foram necessários para determinar a configuração do algoritmo de descida em vizinhança variável apresentada na Tabela 4.3.

Quando necessário, é discutido o ajuste dos parâmetros para os diferentes experimentos realizados.

Tabela 4.3. Valores dos parâmetros do algoritmo para *bin packing* 3D

Parâmetro	Valor
$iter\_i$	20
kn_{max}	3
$iter\_p$	1000
$iter\_s$	2000

As abordagens utilizadas para comparar o desempenho são listadas a seguir:

- **IMM** – heurística baseada em programação inteira (Ivancic et al., 1989)
- **BR_1** – heurística construtiva (Bischoff e Ratcliff, 1995a)
- **BR_2** – heurística construtiva (Bischoff e Ratcliff, 1995b)
- **E_1** – heurística com busca em árvore com estratégia sequencial (Eley, 2002)
- **E_2** – heurística com busca em árvore com estratégia paralela (Eley, 2002)
- **E_3** – heurística baseada em programação inteira (Eley, 2003)
- **LZ** – metaheurística *squeaky wheel optimization* (Lim e Zhang, 2005)

Considerando restrições de estabilidade, orientação e empilhamento, seis conjuntos de instâncias são utilizados nos testes computacionais. O primeiro conjunto foi proposto por Ivancic et al. (1989), os quatro seguintes por Lim e Zhang (2005) e o último conjunto de instâncias por Gendreau et al. (2006a). As características e os resultados computacionais para cada um destes conjuntos são apresentados a seguir.

4.3.1. Resultados computacionais para o conjunto de instâncias de Ivancic et al. (1989)

O conjunto de Ivancic et al. (1989) compreende 47 instâncias em que as caixas, distribuídas em 2 a 5 diferentes tipos, podem ser carregadas com qualquer orientação. Restrições de projeção da base de apoio e empilhamento não são consideradas. Em muitas destas instâncias as dimensões das caixas são comparáveis às dimensões dos contêineres, e não é raro existir soluções em que alguns *bins* possuem 100% do volume ocupado. A Figura 4.3 apresenta a solução ótima para a instância

número 8. Nesta solução, os três primeiros *bins* apresentam 100% de ocupação do volume.

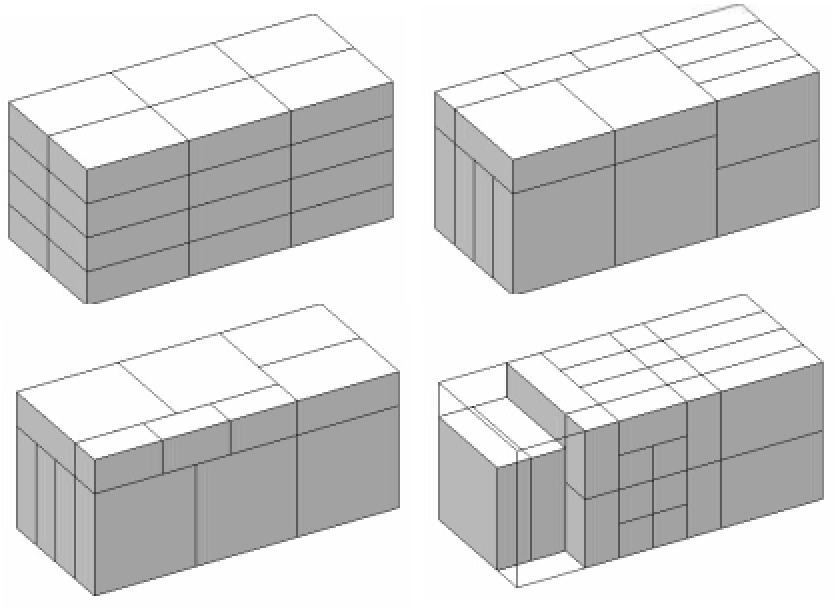


Figura 4.3. Solução ótima para instância número 8 de Ivancic et al. (1989)

Os resultados referentes às versões AA1bp e AA2bp representam a média de dez execuções realizadas com sementes diferentes para o gerador de números pseudo-aleatórios. Além do número máximo de inícios e o número de *bins* igual ao limitante inferior, o tempo computacional máximo é limitado em 1.000 segundos. Testes computacionais demonstraram que este último critério é suficiente para garantir a obtenção de soluções de boa qualidade em um tempo computacional razoável.

Uma vez que muitas caixas deste conjunto de instâncias têm dimensões comparáveis às dimensões do contêiner, é razoável ajustar alguns parâmetros do algoritmo para obter um maior grau de aleatoriedade. Deste modo, para o presente teste computacional os valores dos seguintes parâmetros foram ajustados: $n_1 = 0$ e $n_2 = 1$ para todas instâncias (existem no máximo cinco tipos distintos de caixas); $max\_c = 6$; e $freq\_gr = 100$. A idéia básica deste ajuste é aumentar o grau de aleatoriedade na escolha dos cubóides e, ao mesmo tempo, invocar mais vezes o procedimento que varia o grau de reconstrução. Quando n_1 assume valor zero a

construção dos cubóides passa ser exclusivamente aleatória, a menos da influência da memória baseada nas soluções do conjunto elite.

A Tabela 4.4 apresenta os resultados para AA1pb, AA2pb e mais sete abordagens. O número de tipos de caixas e o limitante inferior de *bins* de cada instância aparecem na segunda e terceira coluna, respectivamente. Considerando o número total de *bins*, AA1pb obtém o melhor resultado e AA2pb o segundo melhor resultado. O algoritmo LZ resolve na otimalidade 29 das 47 instâncias, o mesmo número de AA2pb, enquanto AA1pb obtém 30 soluções ótimas. AA1pb e AA2pb são as únicas abordagens a encontrar o valor ótimo da instância de número 27 e encontram a mesma solução nas dez execuções para 44 instâncias. Se comparados os resultados das duas versões, verifica-se que elas diferem apenas em 4 instâncias. O tempo computacional médio para resolver todas as instâncias é de 395,60 segundos para AA1pb e 417,62 segundos para AA2pb, consideravelmente superior aos 6,43 segundos de LZ em um Pentium 4, 2.4 GHz. No entanto, Lim e Zhang (2005) não deixam claro o critério de parada do algoritmo LZ.

Com os parâmetros ajustados para as instâncias BR1-BR7 (ver Tabela 4.2), em média, AA1pb obtém o total de 695,4 e AA2bp 694,6 *bins.*

Tabela 4.4. Resultados para o conjunto de instâncias de Ivancic et al. (1989)

Inst.	# tipos	L.I.	IMM	BR_1	BR_2	E_1	E_2	E_3	LZ	AA1pb	AA2pb
1	2	19	26	27	27	27	26	25[a]	25	25,0	25,0
2	2	7	11	11	11	11	10	10	10	10,0	10,0
3	4	19	20	21	26	21	22	20	19	19,0	19,2
4	4	26	27	29	27	29	30	26	26	26,0	26,0
5	4	46	65	61	59	55	51	51	51	51,6	51,6
6	3	10	10	10	10	10	10	10	10	10,0	10,0
7	3	16	16	16	16	16	16	16	16	16,0	16,0
8	3	4	5	4	4	4	4	4	4	4,0	4,0
9	2	16	19	19	19	19	19	19[a]	19	19,0	19,0
10	2	37	55	55	55	55	55	55[a]	55	55,0	55,0
11	2	14	18	19	25	17	18	17	16	17,0	17,0
12	3	45	55	55	55	53	53	53[a]	53	53,0	53,0
13	3	20	27	25	27	25	25	25	25	25,0	25,0
14	3	27	28	27	28	27	27	27	27	27,0	27,0
15	3	11	11	11	15	12	12	11	11	11,0	11,0
16	3	21	34	28	29	28	26	26[a]	26	26,0	26,0
17	3	7	8	8	10	8	7	7	7	7,0	7,0
18	3	1	3	3	2	1	1	2[a]	2	2,0	2,0
19	3	1	3	3	3	2	2	3[a]	3	3,0	3,0
20	3	2	5	5	5	2	2	5[a]	5	5,0	5,0
21	5	17	24	24	26	24	26	20	20	20,0	20,0
22	5	8	10	11	11	9	9	8	9	8,0	8,0
23	5	17	21	22	22	21	21	20	20	20,0	20,0
24	4	5	6	6	7	6	6	6	5	5,0	5,0
25	4	4	6	5	5	6	5	5	5	5,0	5,0
26	4	3	3	3	4	3	3	3	3	3,0	3,0
27	3	4	5	5	5	5	5	5	5	4,0	4,0
28	3	9	10	11	12	11	10	10	9	9,4	9,6
29	4	15	18	17	23	18	18	17	17	17,0	17,0
30	4	18	24	24	26	22	23	22	22	22,0	22,0
31	4	11	13	13	14	13	14	13	12	12,0	12,0
32	3	4	5	4	4	4	4	4	4	4,0	4,0
33	3	4	5	5	5	5	5	5	4	4,0	4,0
34	3	5	9	9	8	5	9	8	8	8,0	8,0
35	2	2	3	3	3	2	2	2	2	2,0	2,0
36	2	10	18	19	14	18	14	14[a]	14	14,0	14,0
37	3	12	26	27	23	26	23	23[a]	23	23,0	23,0
38	3	25	50	56	45	46	45	45	45	45,0	45,0
39	3	12	16	16	18	15	15	15	15	15,0	15,0
40	4	7	9	10	11	9	9	8	9	8,1	8,0
41	4	14	16	16	17	16	15	15	15	15,0	15,0
42	3	4	4	5	5	4	4	4	4	4,0	4,0
43	3	3	3	3	3	3	3	3	3	3,0	3,0
44	3	3	4	4	4	4	4	4	3	3,0	3,0
45	4	2	3	3	3	3	3	3	3	3,0	3,0
46	4	2	2	2	2	2	2	2	2	2,0	2,0
47	4	3	4	3	4	3	3	3	3	3,0	3,0
Total		572	763	763	777	725	716	699	694	693,1	693,4

[a] Solução ótima segundo Eley (2003)

4.3.2. Resultados computacionais para os conjuntos de instâncias de Lim e Zang (2005)

Lim e Zhang (2005) propõem conjuntos de instâncias para o problema *bin packing* 3D gerados a partir das instâncias BR1-BR7 de Bischoff e Ratcliff (1995a). A metodologia utilizada é similar a dos últimos autores, com o acréscimo de um parâmetro R, denominado razão alvo, que corresponde à soma do volume das caixas dividido pelo volume $V = L \times W \times H$ de um único *bin*. Em cada instância gerada, o volume total das caixas T_c é maior ou igual a $T_c = R \times V$ e menor ou igual a $T_c + \max_{t=1,\ldots,m}(v_t)$. Em razão do procedimento adotado, o número mínimo de *bins* necessário para conter todas as caixas de uma dada instância é $\lfloor R \rfloor + 1$. As restrições de orientação das instâncias BR1-BR7 são reproduzidas de acordo com a metodologia original. Restrições de projeção da base de apoio e empilhamento não são consideradas.

A razão alvo R assume valores {5,00 ; 5,25 ; 5,50 ; 5,75} e origina quatro conjuntos de instâncias, a saber, BR-5,00-1 a BR-5,00-7; BR-5,25-1 a BR-5,25-7; BR-5,50-1 a BR-5,50-7 e BR-5,75-1 a BR-5,75-7. Cada conjunto consiste de 700 casos e ao todo são geradas 2.800 instâncias que exigem no mínimo seis *bins* para conter todas as caixas. Segundo Lim e Zhang (2005), os quatros valores acima para a razão alvo R fornecem instâncias testes mais desafiadoras e representativas do que as propostas por Eley (2002), que utiliza somente R = 5,00. De fato, para R = 5,00 a média do volume ocupado por *bin* é de 83,3% e para R = 5,75 é de 95,83%.

O algoritmo para carregamento de contêiner que utiliza carregamento simultâneo em dois sentidos apresenta melhor desempenho para as instâncias BR1-BR7. Sendo assim, por causa do grande número de instâncias proposto por Lim e Zhang (2005), somente a versão AA2bp é utilizada nestes testes computacionais. Como terceiro critério de parada é utilizado tempo computacional máximo de 4.000 segundos, que é aproximadamente o tempo máximo gasto pelo algoritmo LZ para resolver um problema de carregamento de contêiner em um computador similar (Pentium 4, 2.4 GHz). Como uma forma de agilizar testes computacionais, os resultados para AAcr representam a média dos experimentos realizados conforme segue. Todas as instâncias são resolvidas com a

primeira semente e se o número de *bins* encontrado é igual a seis (limitante mínimo) o resultado é confirmado com as demais nove sementes.

Tabela 4.5. Resultados para os conjuntos de instâncias de Lim e Zhang (2005)

Instâncias	Valor da razão alvo *R*							
	5,00		5,25		5,50		5,75	
	Método		Método		Método		Método	
	LZ	AA2bp	LZ	AA2bp	LZ	AA2bp	LZ	AA2bp
BR-R-1 (3)	601	600,0	605	602,7	629	617,3	691	677,5
BR-R-2 (5)	600	600,0	600	600,0	612	606,5	689	671,4
BR-R-3 (8)	600	600,0	600	600,0	600	600,0	692	671,1
BR-R-4 (10)	600	600,0	600	600,0	600	600,0	690	677,1
BR-R-5 (12)	600	600,0	600	600,0	600	600,0	692	683,8
BR-R-6 (15)	600	600,0	600	600,0	600	600,0	696	686,0
BR-R-7 (20)	600	600,0	600	600,0	600	600,0	700	695,6
LZ – AA2bp	1		2,3		17,2		87,5	

Como mencionado, são necessários no mínimo seis *bins* para conter todas as caixas de cada instância, de tal modo que o limitante inferior do número total de *bins* para cada classe de instâncias é 600.

A Tabela 4.5 apresenta uma comparação de desempenho com a metaheurística *squeaky wheel optimization* proposta por Lim e Zhang (2005), denotada por LZ. Para cada classe de instância é apresentado o número total de *bins* utilizado. O algoritmo AA2pb apresenta desempenho igual ou melhor em todas as classes de todos os conjuntos de instâncias, em especial, para o conjunto BR-5,00, todas as soluções encontradas são ótimas. Observa-se, na última linha da tabela, que a diferença de desempenho entre AA2pb e LZ aumenta com o acréscimo da razão alvo R.

Os tempos computacionais de AA2bp, em segundos, constam na Tabela 4.6. O resultados para as seis últimas classes do conjunto BR-5,75 são maiores que os demais resultados apresentados porque, predominantemente, neste caso o critério de parada mais utilizado é o tempo computacional máximo de 4.000 segundos. Para a classe BR-5,75-1 e os conjuntos com valor de razão alvo igual a 5,00, 5,25 e 5,50, os critérios de parada utilizados são o número máximo de 20 inícios ou número de *bins* igual ao limitante inferior. Desafortunadamente, Lim e Zhang (2005) não apresentam informação sobre o tempo computacional para LZ, o que impossibilita uma análise comparativa.

Tabela 4.6. Tempo computacional do algoritmo AA2pb para os conjuntos de instâncias de Lim e Zhang (2005)

Instâncias	Valor da razão alvo *R*			
	5,00	5,25	5,50	5,75
	Tempo (s)	Tempo (s)	Tempo (s)	Tempo (s)
BR-R-1 (3)	3,96	63,57	350,73	1.523,04
BR-R-2 (5)	4,19	6,77	298,20	3.182,47
BR-R-3 (8)	8,05	9,94	70,74	3.110,48
BR-R-4 (10)	10,24	11,02	58,70	3.290,53
BR-R-5 (12)	12,14	12,41	28,72	3.585,09
BR-R-6 (15)	15,59	16,12	21,99	3.612,97
BR-R-7 (20)	21,68	22,20	33,47	3.898,28

4.3.3. Resultados computacionais para os conjuntos de instâncias de Gendreau et al. (2006a)

As instâncias de Gendreau et al. (2006a) são originalmente propostas para o problema de roteamento de veículos capacitados 3D. O contêiner tem dimensões $60 \times 25 \times 30$ e as demandas são compostas por no mínimo uma e no máximo três caixas, cujas dimensões assumem um valor escolhido aleatoriamente, seguindo uma distribuição uniforme, entre 20% e 60% da correspondente dimensão do contêiner. A cada demanda também é atribuído um peso igual ao de vinte e sete instâncias para o problema de roteamento de veículos capacitado (Toth e Vigo, 2002). Não existe relação entre o peso das demandas e o volume das caixas. As caixas são classificadas em frágeis ou não frágeis. Um carregamento para ser factível deve satisfazer as seguintes restrições:

R1. **Orientação**: uma ou duas dimensões de algumas caixas não podem ser orientadas verticalmente. Se nenhuma restrição sobre a orientação de uma caixa for imposta, então esta pode ser carregada com uma das seis possíveis orientações.

R2. **Estabilidade**: uma caixa deve ter uma área de suporte maior que um limite mínimo estipulado sobre uma porcentagem da base.

R3. **Múltiplos destinos:** grupos de itens ou demandas com diferentes destinos devem ser posicionados próximos dentro do mesmo contêiner. Quando um cliente é visitado, deve ser possível descarregar

cada item da sua demanda com um único movimento direto e paralelo à direção do comprimento do contêiner, ou seja, a caixa a ser entregue deve ser completamente visível quando o observador é posicionado na frente do carregamento em uma altura adequada.

R4. **Empilhamento:** Nenhum item não frágil pode ser carregado sobre um item frágil, no entanto, pilhas de itens frágeis são permitidas.

R5. **Peso**: a soma dos pesos das demandas não deve ultrapassar a capacidade de peso do contêiner.

Neste experimento computacional são consideradas as quatro primeiras restrições, uma vez que o objetivo é investigar tão somente o aspecto tridimensional do carregamento.

As vinte e sete instâncias são divididas igualmente em três grupos de acordo com o número de clientes, a saber, $n \leq 25$, $25 < n < 50$ e $n \geq 50$. Como a identificação simultânea dos espaços vazios não faz sentido com restrições de múltiplos destinos, somente os resultados para o algoritmo AA1bp são apresentados na Tabela 4.7. A primeira coluna apresenta a denominação original das instâncias, em que o número de clientes e veículos pode ser identificado, e a segunda coluna apresenta o número total de caixas. A terceira coluna apresenta o número ótimo de *bins* para o caso unidimensional (1D) e as duas últimas o número de *bins* encontrado para o caso tridimensional quando as restrições R1 a R4 são consideradas (3DR) e quando estas são relaxadas (3D). Neste último caso, todas as orientações possíveis das caixas são permitidas, a restrição R2 é mantida e as restrições R3 e R4 são completamente desconsideradas,

A diferença entre o número de *bins* encontrados em 3DR e 3D aumenta gradativamente ao longo dos três grupos de instâncias (6, 8 e 15, respectivamente). Ao todo, o caso 3D necessita 29 *bins* a menos que 3DR e em apenas 5 instâncias o número de *bins* permanece inalterado. Comparando os resultados entre 1D e 3DR, observa-se que duas instâncias apresentam o mesmo número e oito um número menor de *bins* no caso 3DR. Para 3D existem duas instâncias com o número igual e nove com o número menor de *bins* em relação à 1D. Isto é possível porque a escolha das dimensões das caixas não possui relação com o peso das demandas e no caso unidimensional somente as restrições de peso são consideradas, o que não ocorre para 3DR e 3D.

O tempo computacional médio para resolver todas as instâncias para o caso 3DR é de 84,99 segundos e 65,37 segundos para 3D.

Tabela 4.7. Resultados para os conjuntos de instâncias de Gendreau et al. (2006a)

	Instância	# caixas	1D	3DR	3D
01	E016-03m	32	3	4	3
02	E016-05m	26	5	3	3
03	E021-04m	37	4	4	4
04	E021-06m	36	6	4	3
05	E022-04g	45	4	5	5
06	E022-06m	40	6	5	4
07	E023-03g	46	3	5	4
08	E023-05s	43	5	5	4
09	E026-08m	50	8	6	5
10	E030-03g	62	3	7	6
11	E030-04s	58	4	7	6
12	E031-09h	63	9	7	6
13	E033-03n	61	3	6	6
14	E033-04g	72	4	8	7
15	E033-05s	68	5	8	6
16	E036-11h	63	11	6	6
17	E041-14h	79	14	8	7
18	E045-04f	94	4	10	9
19	E051-05e	99	5	10	9
20	E072-04f	147	4	14	13
21	E076-07s	155	7	15	13
22	E076-08s	146	8	15	13
23	E076-10e	150	10	15	13
24	E076-14s	143	14	13	12
25	E101-08e	193	8	17	16
26	E101-10c	199	10	21	18
27	E101-14s	198	14	20	18
Total			181	248	219

Capítulo 5

Problema de carregamento e roteamento de veículos

5.1 Introdução

Neste capítulo é considerada uma extensão do problema de roteamento de veículos capacitados (PRVC) em que restrições de empacotamento tridimensionais são introduzidas. Este problema é denominado problema de roteamento de veículos capacitados com restrições de empacotamento tridimensionais (PRVC3D).

O PRVC é um dos mais estudados problemas de roteamento de veículos por ser de interesse prático e de difícil resolução. O objetivo é a otimização do despacho de lotes de mercadorias demandados por um conjunto de clientes e operado por uma frota homogênea de veículos que partem de um depósito central. Para a resolução deste problema, em geral, as demandas dos clientes são representadas pelo peso total das mercadorias despachadas, enquanto em problemas reais as demandas consistem de itens discretos. Quando as restrições de empacotamento são descritas por um problema unidimensional, assume-se implicitamente que cada demanda ocupa uma determinada seção do contêiner, ou ainda, que a carga molda-se de acordo com o formato do contêiner. A primeira situação acarreta desperdício de espaço, enquanto a segunda pode levar a soluções infactíveis quando a carga é formada por objetos rígidos que não alteram a forma sob pressão. Se restrições de empacotamento mais reais são consideradas, a acoplagem entre as estruturas de empacotamento e roteamento torna-se mais crítica e um tratamento especial deve ser utilizado para evitar custos computacionais proibitivos.

O PRVC possui muitas variações, tais como janelas de tempo, múltiplos depósitos e frota heterogênea de veículos. Um número de algoritmos exatos e

metaheurísticas estão disponíveis na literatura para o PRVC (ver Laporte et al., 2000 e Toth e Vigo, 2002 para uma revisão da literatura). Enquanto algoritmos exatos podem resolver problemas de 'pequeno porte, metaheurísticas têm sucesso em problemas maiores. Estudos recentes são apresentados por Toth e Vigo (2003), Reimann et al. (2004), Fukasawa et al. (2004) e Wassan (2006). No que se refere ao PRVC3D, existe apenas o estudo de Gendreau et al. (2006a) na literatura, como já discutido no capítulo 2.

Para resolver o PRVC3D é proposto um algoritmo de busca tabu no qual as restrições de empacotamento tridimensionais são satisfeitas utilizando os algoritmos para empacotamento tridimensional descritos nos capítulos anteriores deste trabalho. Na próxima seção o PRVC3D é formalmente definido e o algoritmo de busca tabu para a sua resolução é detalhado.

5.2 Algoritmo de busca tabu para o problema de roteamento de veículos capacitados 3D

O PRVC3D é definido em um grafo completo não direcionado $G=(V,E)$, em que V é um conjunto de $n+1$ vértices que correspondem ao depósito (vértice 0) e aos clientes (vértices $1, ..., n$), e E é o conjunto de arestas (i, j), com um custo não negativo c_{ij} associado. É dado um número máximo de u veículos, com uma capacidade máxima de peso C e um espaço de armazenamento com dimensões $L \times W \times H$. A demanda do cliente i $(i=1,\ldots,n)$ consiste de itens agrupados em m tipos que podem assumir $k = 1,..., 6$ orientações, cada tipo t $(t=1,\ldots,m)$ é identificado por três dimensões espaciais $d_{1tk}, d_{2tk}, d_{3tk}$, um volume v_t, um peso w_t e um parâmetro binário f_t para fragilidade (1 se a caixa for frágil e 0 caso contrário). Denotando por q_{it} o número de caixas do tipo t que compõem a demanda do cliente i, o volume total da demanda é dado por $\sum_{t=1}^{m} q_{it} v_t$ e o peso total por $\sum_{t=1}^{m} q_{it} w_t$. O carregamento dos itens deve obedecer, além das restrições sobreposição, as seguintes restrições:

R1. **Orientação**: uma ou duas dimensões de algumas caixas não podem ser orientadas verticalmente. Se nenhuma restrição sobre a orientação de uma caixa for imposta, então esta pode ser carregada com uma das seis orientações possíveis.

R2. **Estabilidade**: uma caixa deve ter uma área de suporte maior que um limite mínimo estipulado sobre uma porcentagem da base.

R3. **Múltiplos destinos:** o carregamento de demandas formadas pelo agrupamento de diferentes itens deve respeitar a ordem com que estes devem ser entregues nos respectivos destinos. Quando um cliente é visitado, deve ser possível descarregar cada item da sua demanda com um único movimento direto e paralelo à direção do comprimento do contêiner, ou seja, a caixa a ser entregue deve ser completamente visível quando o observador é posicionado na frente do carregamento em uma altura adequada.

R4. **Empilhamento:** Nenhum item não frágil pode ser carregado sobre um item frágil, no entanto, pilhas de itens frágeis são permitidas.

R5. **Peso**: a soma dos pesos das demandas não deve ultrapassar a capacidade de peso do contêiner.

Assume-se que cada demanda pode ser carregada por inteiro dentro do espaço de armazenamento dos veículos, e que é possível realizar todas as entregas com no máximo u veículos.

O objetivo é encontrar uma partição de clientes em no máximo u subconjuntos de modo a minimizar o número de veículos e o custo total. As rotas associadas aos subconjuntos devem ter início e término no depósito (vértice 0), tal que cada cliente seja visitado uma única vez, as restrições descritas acima sejam satisfeitas e a soma dos pesos das demandas não exceda a capacidade C dos veículos.

O algoritmo de busca tabu proposto possui algumas similaridades com o algoritmo de Gendreau et al. (2006a), que por sua vez utiliza técnicas introduzidas por Gendreau et al. (1994) e Grendreau et al. (2006b). No entanto, é diferente em alguns

componentes fundamentais, como a vizinhança e o algoritmo para o carregamento tridimensional.

Uma solução inicial é obtida a partir da heurística construtiva paralela de Clarke e Wright (1964). A única adaptação necessária é que, ao aceitar a união de quaisquer duas rotas, deve-se verificar a factibilidade do carregamento tridimensional. A Figura 5.1 apresenta um pseudocódigo para a heurística construtiva de Clarke e Wright. Inicialmente n rotas são construídas (linha 1) e, em seguida, as economias são calculadas (linhas 2 a 5). Após ordenar as economias (linha 6), tenta-se conectar rotas distintas na ordem estipulada (linha 7). Ao final a solução obtida é retornada (linha 8).

	Procedimento heurística construtiva Clarke e Wright
1	$S \leftarrow$ Criar n rotas, uma para cada cliente
2	**Para** cada par de cidades **faça**
3	$s_{ij} = c_{0i} + c_{0j} - c_{ij}$
4	**Se** $s_{ij} > 0$ **então** *lista_savings* $\leftarrow s_{ij}$
5	**Fim_Para**
6	*lista_savings* $\leftarrow$ *lista_saving* em ordem não crescente de valor
7	$S \leftarrow$ Percorra a *lista_savings* e faça a ligação entre as cidades i e j conectadas ao depósito, desde que a rota resultante respeite as restrições de peso e empacotamento tridimensional
8	**Retorna (**S)

Figura 5.1. Pseudocódigo para a heurística construtiva Clarke e Wright (1964)

Caso o número de veículos da solução inicial seja maior que o máximo permitido u, a rota com menor número de clientes é desmontada e estes são inseridos em outras rotas de modo a minimizar o custo. Este procedimento é iterado até que o número de veículos máximo seja obtido e, invariavelmente, leva uma solução infactível quanto ao peso ou comprimento do carregamento.

No processo de busca, duas vizinhanças são definidas pelos seguintes movimentos:

1. Inserção: um cliente é removido de sua posição corrente e inserido em posições diferentes na mesma rota ou em outra.
2. Troca de um par de clientes: dois clientes trocam de posição dentro da mesma rota ou em rotas distintas.

O movimento é denominado intra-rota se os clientes envolvidos pertencerem à mesma rota e inter-rota se os clientes pertencerem a rotas distintas. Ordinariamente são utilizados somente movimentos inter-rotas, com exceção das três iterações seguintes à atualização da solução quando são permitidos os dois tipos de movimentos. Em qualquer caso, as vizinhanças sempre são consideradas em conjunto e a cada iteração da busca tabu o melhor movimento é executado. Dado que soluções infactíveis são aceitas durante a busca, a composição dos dois movimentos permite factibilizar soluções mais facilmente.

O tempo requerido para explorar uma vizinhança depende não só do tamanho da mesma como também do tempo necessário para verificar a factibilidade das soluções geradas. No PRVC3D é necessário calcular um padrão de carregamento tridimensional a cada avaliação de movimento, o que pode levar a custos computacionais proibitivos. Uma forma de contornar este problema é lançar mão de técnicas de redução de vizinhança. Estas técnicas compreendem regras que são aplicadas com o intuito de excluir movimentos com pouca chance de causar melhoria no valor da função objetivo. Com isto, é possível explorar um número maior de vizinhanças e avaliar um número menor de soluções. A restrição de vizinhança utilizada considera, para cada cliente i, os arcos incidentes c_{ij} tomados em ordem crescente de custo e um parâmetro p. Define-se como candidatos de i os p clientes correspondentes aos p menores custos. A cada movimento, um cliente i só pode ser designado para uma rota se esta contiver pelo menos um dos p candidatos de i. Quando é executado um movimento que leva a uma atualização da solução incumbente o valor do parâmetro p é duplicado nas três iterações seguintes. Da mesma forma, se o número de iterações sem atualização da solução incumbente for maior que um limiar definido pelo parâmetro $não\_atualiza$, o valor de p é duplicado por três iterações consecutivas.

Quando um movimento é executado, a re-inserção de um cliente na rota da qual ele foi retirado (pode ser na mesma) é declarado tabu por q iterações, a menos que isso leve a uma melhora da solução incumbente (critério de aspiração).

Durante a exploração das vizinhanças uma solução S é avaliada pela seguinte função objetivo:

$$f(S) = (\text{distância total}) + a \cdot (\text{excesso de peso}) + b \cdot (\text{excesso de comprimento}) + g \cdot frq(i, j)$$

em que $frq(i, j)$ representa a razão entre o número de vezes que um movimento designou o cliente i para a rota j e o número total de movimentos realizados. O segundo e o terceiro termo da expressão de avaliação penalizam infactibilidades, enquanto o quarto termo promove uma diversificação na busca. A distância total e o excesso de peso são trivialmente calculados, no entanto, o excesso de comprimento exige que um padrão de carregamento tridimensional seja gerado. Isto é feito utilizando o algoritmo de múltiplos inícios para carregamento de contêiner descrito no capítulo 3, modificado para trabalhar com um contêiner com comprimento variável, ou seja, resolve-se o problema *strip packing* 3D. O parâmetro g penaliza as designações mais freqüentes de clientes para veículos, enquanto o excesso de peso e o excesso no comprimento do contêiner são penalizados pelos parâmetros a e b, respectivamente.

Os parâmetros a e b são ajustados no decorrer da busca conforme segue. Seja Δ_α a variação do excesso de peso. Se Δ_a é positivo, então a é incrementado em 10%, enquanto se Δ_a for negativo, α é decrementado em 10%. Ainda, se o número de soluções consecutivas e infactíveis quanto ao peso ultrapassar o valor do parâmetro *infac_peso*, então o valor de a é incrementado em 50%. A mesma estratégia é aplicada para o ajuste do parâmetro b, em que a variação do excesso de comprimento Δ_b e o parâmetro *infac_comp* são considerados.

Para minimizar a distância total percorrida pela frota de veículos é importante calcular padrões de carregamento tridimensional factíveis. Por isso, a intensificação da busca é restrita especificamente aos procedimentos de empacotamento. Dada uma solução, se o movimento corrente diminui o valor de função objetivo, o número soluções avaliadas pelo algoritmo de empacotamento 3D é $iter\_3D_1$, caso contrário este número é $iter\_3D_2$, com $iter\_3D_1 > iter\_3D_2$. Se após $iter\_3D_1$ iterações a solução encontrada for infactível e o excesso de comprimento do carregamento não exceder em 10% o comprimento do contêiner, então o algoritmo avalia mais $iter\_3D_1$ soluções no intuito de tentar obter um carregamento factível.

A Figura 5.2 apresenta um pseudocódigo do algoritmo busca tabu para o problema de roteamento de veículo capacitados 3D. A restrição de vizinhança e a forma de avaliar as soluções são idênticas às utilizadas em (Gendreau et al., 2006a). Na linha 1 dados da instância e parâmetros do algoritmo são processados. Nas linhas 2 a 4, a

solução inicial é calculada, se necessário o número de veículos é factibilizado e a solução incumbente recebe a solução inicial. Observe que a solução inicial pode ser infactível com relação ao peso ou ao comprimento do contêiner. O contador de iterações é inicializado na linha 5. O laço principal é apresentado nas linhas 6 a 26, e termina quando um critério de parada é satisfeito. O melhor movimento das vizinhanças é escolhido na linha 7 e aplicado à solução corrente na linha 8. Após atualizar a memória de curto prazo, verifica-se se a solução incumbente deve ser atualizada, ver linhas 9 a 12. O parâmetro p tem seu valor temporariamente duplicado se a solução incumbente não for atualizada por um número de iterações superior ao valor do parâmetro *não_atualiza*, ver linhas 14 a 17. Os valores dos parâmetros de penalidade a e b são ajustados nas linhas 18 a 26. A solução incumbente é retornada na linha 28.

A adaptação do algoritmo de carregamento de contêiner para resolver o problema *strip packing* 3D é de fácil implementação, basta acrescentar o controle do comprimento do contêiner. O primeiro padrão de carregamento é obtido a partir do empacotamento de todas as caixas das demanda em um contêiner com comprimento superestimado. O comprimento máximo ocupado do carregamento da solução inicial, denominado *max_x,* é decrementado de uma unidade e passa a determinar o comprimento do contêiner da próxima solução a ser construída. Este procedimento é iterado até que não seja possível encontrar soluções factíveis, situação na qual o comprimento assume valores discretos dentro do intervalo [*max_x–amplitude; max_x+amplitude*], em que a *amplitude* é um parâmetro do algoritmo que controla a oscilação do comprimento. É importante observar que as demandas devem ser carregadas obedecendo à seqüência da rota a qual pertence.

O algoritmo é detalhado no pseudocódigo apresentado na Figura 5.3. Na linha 1 as variáveis utilizadas para controlar a oscilação do comprimento do contêiner são inicializadas com valor zero, e o valor inicial deste é inicializado com infinito na linha 2. A solução inicial é calculada na linha 3 e atribuída com infactibilidade de comprimento à solução incumbente na linha 4. A variável *max_x* recebe o comprimento máximo ocupado pelo carregamento da solução inicial decrementado de uma unidade na linha 5. O laço principal do algoritmo, linhas 7 a 22, termina quando um critério de parada é satisfeito. A cada iteração as demandas são carregadas e uma nova solução é gerada, ver linha 8. Sempre que a solução incumbente é atualizada o comprimento do contêiner é

decrementado de uma unidade, ver linhas 10 a 12. Caso contrário, se o número de tentativas em melhorar a solução incumbente ultrapassar o valor do parâmetro

	Procedimento busca tabu para o problema de roteamento de veículo capacitados 3D
1	Leia e processe os dados da instância e parâmetros do algoritmo
2	$S \leftarrow$ construa solução inicial
3	**Se** número de veículos de $S > u$ **então** $S \leftarrow$ factibiliza número de veículos **Fim_Se**
4	$S_{incumbente} \leftarrow S$
5	$iter \leftarrow 0$
6	**Enquanto** (critério de parada não satisfeito) **faça**
7	$mov \leftarrow$ melhor movimento não tabu das vizinhanças de troca e inserção
8	$S \leftarrow$ executa movimento *mov* em S
9	atualiza *lista_tabu*
10	**Se** (S é factível **E** distância total de $S <$ distância total de $S_{incumbente}$) **OU** (infactiblidade de $S <$ infactibilidade de $S_{incumbente}$) **então**
11	$S_{incumbente} \leftarrow S$
12	dobra o valor de p e inclua movimentos intra-rotas nas vizinhanças de troca e inserção pelas próximas 3 iterações
13	**Fim_Se**
14	**Se** ($S_{incumbente}$ não foi atualizada nas últimas *não_atualiza* iterações) **então**
15	dobra o valor de p pelas próximas 3 iterações;
16	inicializa contador de iterações sem atualização da $S_{incumbente}$
17	**Fim_Se**
18	**Se** (as últimas *infac_peso* soluções são infactíveis com relação ao peso) **então**
19	$b \leftarrow 1{,}5 \cdot b$; inicializa contador de soluções infactíveis com relação ao peso
20	**Fim_Se**
21	**Se** (as últimas *infac_comp* soluções são infactíveis com relação ao comprimento) **então**
22	$a \leftarrow 1{,}5 \cdot a$; inicializa contador de soluções infactíveis com relação ao comprimento
23	**Fim_Se**
24	**Se** $\Delta_a > 0$ **então** $a \leftarrow 1{,}1 \cdot a$ **Senão Se** $\Delta_a < 0$ **então** $a \leftarrow 0{,}9 \cdot a$ **Fim_Se**
25	**Se** $\Delta_b > 0$ **então** $b \leftarrow 1{,}1 \cdot b$ **Senão Se** $\Delta_b < 0$ **então** $b \leftarrow 0{,}9 \cdot b$ **Fim_Se**
26	$iter \leftarrow iter + 1$
27	**Fim_Enquanto**
28	**Retorna** ($S_{incumbente}$)

Figura 5.2. Pseudocódigo do algoritmo busca tabu para o problema de roteamento de veículo capacitados 3D

```
    Procedimento múltiplos inícios para strip packing 3D
1   delta ← 0; não_melhora ← 0 ;
2   inicializa comprimento do contêiner com ∞
3   S ← Carrega_demandas()
4   S_incumbente ← S
5   max_x ← (comprimento máximo ocupado em S) – 1
6   iter ← 0
7   Enquanto (critério de parada não satisfeito) faça
8     S ← Carrega_demandas()
9     Se (comprimento máximo ocupado em S < comprimento máximo ocupado em S_incumbente) então
10      S_incumbente ← S
11      não_melhora ← 0;   max_x ← max_x –1
12      atualiza comprimento do contêiner com max_x
13    Senão
14      não_melhora ← não_melhora +1
15      Se (não_melhora > max_não_melhora ) então
16        delta ← delta –1
17        Se (delta < - amplitude ) então  delta ← amplitude  Fim_Se
18        atualiza comprimento do contêiner com max_x + delta
19      Fim_Se
20    Fim_Se
21  Fim_Enquanto
22  Retorna (S_incumbente)
```

Figura 5.3. Pseudocódigo do algoritmo de múltiplos inícios para *strip packing* 3D

max_não_melhora, o comprimento de contêiner passa a oscilar em torno do valor de *max_x* com uma amplitude determinada pelo parâmetro *amplitude*, ver linhas 14 a 18. Para tanto, a variável *delta*, utilizada para ajustar o comprimento do contêiner, é decrementada a cada iteração na linha 16 e inicializada na linha 17 quando assume o valor *–amplitude*. Ao final a melhor solução encontrada é retornada, ver linha 23.

A Tabela 5.1 lista os parâmetros do algoritmo de busca tabu com uma breve descrição de cada um.

Tabela 5.1. Descrição dos parâmetros do algoritmo busca tabu para o problema de roteamento de veículo capacitados 3D

Parâmetro	Descrição
p	número máximo de candidatos de cada cliente
$iter\_3D_1$ e $iter\_3D_2$	número máximo de soluções avaliadas no empacotamento tridimensional, com $iter\_D_1 > iter\_D_2$.
q	tamanho máximo da lista tabu (memória de curto prazo)
a	penalidade para infactibilidade quanto ao peso do carregamento
b	penalidade para infactibilidade quanto ao comprimento do carregamento
não_atualiza	número máximo de iterações sem atualização da solução incumbente a partir do qual o valor de p é duplicado por três iterações
infac_peso	número de soluções consecutivas infactíveis quanto ao peso do carregamento a partir do qual a é ajustado.
infac_comp	número de soluções consecutivas infactíveis quanto ao comprimento do carregamento a partir do qual b é ajustado.

No que se refere ao algoritmo utilizado para gerar os padrões de carregamento tridimensional, além dos parâmetros descritos na Tabela 3.6 do Capítulo 3, dois novos parâmetros são definidos: *max_não_melhora* e *amplitude*. O primeiro determina o início do processo de oscilação do comprimento do contêiner e o segundo determina a amplitude máxima da oscilação.

5.3 Resultados computacionais

Todos os algoritmos foram implementados em linguagem C++ e compilados com GCC versão 3.3.3 com a opção de otimização –O3. Os testes computacionais foram realizados em um PC Intel Pentium 4 28 GHZ com sistema operacional Linux Fedora Core 2.

O algoritmo de busca tabu é denominado AAcr e o seu desempenho é validado com vinte e sete instâncias divididas igualmente em três grupos de acordo com o

número de clientes, a saber, $n \leq 25$, $25 < n < 50$ e $n \geq 50$ (Gendreau et al., 2006a). Os resultados dos testes computacionais são apresentados tomando as restrições R1 a R5 em conjunto ou separadamente.

As instâncias possuem o grafo, os pesos das demandas e a capacidade de peso dos veículos derivados de instâncias para PRVC com distâncias euclidianas. O espaço para carregamento possui dimensões $60 \times 25 \times 30$. As dimensões das caixas que compõem as demandas foram geradas a partir de um valor escolhido aleatoriamente, seguindo uma distribuição uniforme, entre 20% e 60% da correspondente dimensão do contêiner. Isto é feito para propiciar caixas de tamanho variado. As demandas de cada cliente i (i = 1, ..., n) são compostas por no mínimo uma e no máximo três caixas, e estas podem ser frágeis ou não frágeis. As caixas devem ser carregadas com não mais que 25% de projeção da base de apoio. O número de veículos máximo foi determinado de modo a garantir a existência de soluções factíveis.

O ajuste dos parâmetros mostrou-se pouco sensível e um número reduzido de testes computacionais foi necessário para determinar a configuração apresentada a seguir. Os valores dos parâmetros da busca tabu apresentados na Tabela 5.2 foram encontrados através de experimentos computacionais. O termo $\bar{c}$ que consta na inicialização dos parâmetros a e b representa a média do custo dos arcos.

Tabela 5.2. Valores dos parâmetros do algoritmo AAcr

Parâmetro	Valor
q	$\min\{7, \lceil n/4 \rceil\}$
a^*	$20 \cdot \bar{c}/C$
b^*	$20 \cdot \bar{c}/L$
$iter\_3D_1$	1000
$iter\_3D_2$	100
p	$\lceil n/4 \rceil$
não_atualiza	20
infac_peso	10
infac_comp	10

* valor inicial

Dado que o conjunto de caixas considerado é fortemente heterogêneo e o desempenho com diferentes graus de reconstrução não promoveu melhora significativa

no desempenho, o empacotamento tridimensional é calculado com a versão restrita do algoritmo de múltiplos inícios, ou seja, sem os procedimentos relativos à memória e diferentes graus de reconstrução. Os valores dos parâmetros são apresentados na Tabela 5.3 e os conjuntos de critérios de avaliação dos cubóides na Tabela 5.4. Os parâmetros r, t e K apresentam valores idênticos aos utilizados nos testes computacionais do Capítulo 3.

Tabela 5.3. Valores dos parâmetros do algoritmo para carregamento tridimensional

Parâmetro	Valor
n_1	1
n_2	2
r	0,80
t	10
K	100
max_não_melhora	50
amplitude	$0{,}10 \cdot L$

Tabela 5.4. Conjuntos de critérios de avaliação

Conjunto	Critérios
*Critérios*1	C_1, C_4, C_5, C_6
*Critérios*2	C_5, C_3, C_4, C_6
*Critérios*3	C_5, C_1, C_4, C_6
*Critérios*4	C_3, C_5, C_4, C_6

O desempenho de AAcr é comparado com o do algoritmo proposto por Gendreau et al. (2006a), que possui duas versões diferentes, uma com um único início (GILM1) e outra com múltiplos inícios (GILM2). O critério de parada utilizado é de meia hora para $n \leq 25$, uma hora para $25 < n < 50$ e duas horas para $n \geq 50$.

A Tabela 5.5 apresenta os resultados do algoritmo AAcr para as instâncias em questão. A duas primeiras colunas mostram a denominação e o número de caixas de cada instância, respectivamente. Na coluna seguinte, a distância total da solução inicial é apresentada se esta for factível e, a seguir, tem-se o número de veículos e a distância total da solução final. A última coluna apresenta o tempo computacional, em segundos, utilizado para encontrar a solução incumbente. O algoritmo AAcr encontra 17 soluções iniciais factíveis, enquanto os autores de GILM1 e GILM2 relatam o total de 15. Considerando o tempo computacional para encontrar a solução incumbente, verifica-se

que 15 minutos são suficientes para AAcr resolver as nove primeiras instâncias. Gendreau et al. (2006a) utilizam um Pentium 4 3.0 GHz e reportam um tempo computacional médio de 2.058,9 segundos para GILM1 e 1.358,2 segundos para GILM2, o que possibilita concluir que o tempo computacional médio de AAcr para encontrar a melhor solução é condizente com os destas abordagens. Para resolver todas as instâncias AAcr utiliza 306 veículos.

Tabela 5.5. Resultados do algoritmo AAcr para as instâncias de Gendreau et al. (2006a)

Instância	# caixas	Solução Inicial	# v	Solução Final	Tempo (s)
E016-03m	32	320,12	4	304,13	73,52
E016-05m	26	–	5	334,96	224,92
E021-04m	37	397,369	5	391,65	180,89
E021-06m	36	–	6	447,98	111,99
E022-04g	45	494,79	6	454,14	12,86
E022-06m	40	–	6	499,07	707,16
E023-03g	46	–	6	865,77	27,47
E023-05s	43	886,74	6	823,21	13,13
E026-08m	50	–	8	645,14	331,95
E030-03g	62	953,58	8	849,33	275,43
E030-04s	58	938,55	8	822,17	353,23
E031-09h	63	–	9	625,92	474,61
E033-03n	61	3.091,97	7	2.807,56	154,98
E033-04g	72	1.779,91	8	1.494,67	589,40
E033-05s	68	1.641,36	8	1.467,70	1.914,75
E036-11h	63	–	11	702,70	88,28
E041-14h	79	–	14	879,57	2.114,28
E045-04f	94	1.338,06	11	1.316,12	2.227,61
E051-05e	99	913,26	11	823,48	2.921,06
E072-04f	147	672,91	17	623,35	7.153,52
E076-07s	155	1.183,92	17	1.168,79	2.408,49
E076-08s	146	1.287,58	19	1.252,62	1.218,94
E076-10e	150	-	17	1.216,21	5.566,58
E076-14s	143	1.255,19	16	1.193,12	7.188,71
E101-08e	193	1.545,11	22	1.499,02	6.976,25
E101-10c	199	1.840,23	26	1.782,83	7.157,83
E101-14s	198	-	25	1.675,28	2.304,7
Média				998,76	1.954,54
Total			306		

A Tabela 5.6 apresenta os resultados comparativos de AAcr com GILM1 e GILM2. As duas primeiras colunas identificam as instâncias e indicam o número total de

caixas. As três colunas seguintes mostram o número de veículos e a distância total das soluções obtidas por GILM1 e GILM2, respectivamente. O número de veículos para AAcr é apresentado na sexta coluna e a distância total na sétima. As duas últimas colunas exibem a diferença de ganho percentual entre as abordagens AAcr e GILM1 (d1) e AAcr e GILM2 (d2), dada por $d = 100 \cdot (A - G) / \min\{A, G\}$, em que d é o ganho percentual, A é o valor da solução encontrada por AAcr e G o valor encontrado por GILM1 ou GILM2, conforme o caso.

Tabela 5.6. Resultados comparativos para as instâncias de Gendreau et al. (2006a)

Instância	# caixas	# v	GILM1	GILM2	# v	AAcr	d1 (%)	d2 (%)
E016-03m	32	5	316,32	316,32	4	304,13	-4,01	-4,01
E016-05m	26	5	350,58	350,58	5	334,96	-4,66	-4,66
E021-04m	37	5	447,73	447,73	5	391,65	-14,32	-14,32
E021-06m	36	6	448,48	448,48	6	447,98	-0,11	-0,11
E022-04g	45	7	464,24	464,24	6	454,14	-2,22	-2,22
E022-06m	40	6	504,46	504,46	6	499,07	-1,08	-1,08
E023-03g	46	6	831,66	831,66	6	865,77	4,10	4,10
E023-05s	43	8	871,77	873,14	6	823,21	-5,90	-6,07
E026-08m	50	8	666,10	676,60	8	645,14	-3,25	-4,88
E030-03g	62	10	911,16	893,61	8	849,33	-7,28	-5,21
E030-04s	58	9	819,36	818,65	8	822,17	0,34	0,43
E031-09h	63	9	651,58	717,74	9	625,92	-4,10	-14,67
E033-03n	61	9	2.928,34	2.816,93	7	2.807,56	-4,30	-0,33
E033-04g	72	11	1.559,64	1.548,41	8	1.494,67	-4,35	-3,60
E033-05s	68	10	1.452,34	1.488,79	8	1.467,70	1,06	-1,44
E036-11h	63	11	707,85	714,37	11	702,70	-0,73	-1,66
E041-14h	79	14	920,87	909,99	14	879,57	-4,70	-3,46
E045-04f	94	14	1.400,52	1.452,02	11	1.316,12	-6,41	-10,33
E051-05e	99	13	871,29	827,99	11	823,48	-5,81	-0,55
E072-04f	147	20	732,12	732,12	17	623,35	-17,45	-17,45
E076-07s	155	18	1.275,20	1.226,20	17	1.168,79	-9,10	-4,91
E076-08s	146	19	1.277,94	1.291,53	19	1.252,62	-2,02	-3,11
E076-10e	150	18	1.258,16	1.271,40	17	1.216,21	-3,45	-4,54
E076-14s	143	18	1.307,09	1.317,86	16	1.193,12	-9,55	-10,45
E101-08e	193	24	1.570,72	1.616,39	22	1.499,02	-4,78	-7,83
E101-10c	199	28	1.847,95	1.839,12	26	1.782,83	-3,65	-3,16
E101-14s	198	25	1.747,52	1.773,50	25	1.675,28	-4,31	-5,86
Média			1.042,26	1.043,33		998,76	-4,52	-4,87
Total		336			306			

AAcr apresenta um melhor desempenho médio nos três grupos de instâncias. A única instância para a qual a distância total encontrada é maior e o número de veículos

é idêntico ao de GILM1 e GILM2 ocorre em E023-03g. Para E030-04s e E033-05s a distância total de AAcr é maior que GILM1 e/ou GILM2, mas o número de veículos utilizados é menor. No último grupo de instâncias AAcr encontra um número de veículos menor para todas as instâncias, com exceção de E076-08s. Mesmo assim, em todas as instâncias deste grupo a distância total de AAcr é menor, com a diferença percentual de –17,45 % para E072-04f. No total, AAcr necessita de 30 veículos a menos que GILM1 e GILM2 e obtém, em média, rotas com distância aproximadamente 5% menor.

O efeito nos resultados de AAcr quando algumas restrições são eliminadas pode ser observado na Tabela 5.7. Nesta tabela, a primeira coluna apresenta a identificação das instâncias e as duas seguintes os resultados de AAcr quando todas as restrições são consideradas (3DR). As demais colunas apresentam o número de veículos, a distância total e a diferença percentual em relação à 3DR quando uma ou mais restrições são eliminadas. Os resultados sem as restrições de empilhamento são identificados por F, sem as restrições de múltiplos destinos por MD e sem as restrições de empilhamento, múltiplos destinos e todas as orientações das caixas possíveis por 3D.

Utilizando os resultados em 3DR como referência, tanto o número total de veículos como a distância percorrida diminuem quando as restrições de empilhamento e múltiplos destinos são desconsideradas, uma de cada vez (F e MD) ou as duas ao mesmo tempo (3D). Em F, seis instâncias apresentam distância percorrida maior do que em 3DR e uma delas, E076-07s, necessitou de um veículo a mais. E076-10e precisa de um veículo a mais, no entanto, a distância total percorrida é menor. Em MD, duas instâncias apresentam distância total percorrida maior, mas somente uma delas, E022-06m, necessita do mesmo número de veículos utilizado em 3DR. As instâncias E101-10c, E101-08e e E076-14s apresentam um veículo a mais com uma distância total percorrida menor. Em 3D, todos os resultados são melhores ou iguais aos obtidos em 3DR. A diferença percentual média em relação à 3DR é maior em 3D, seguido por MD e F. O mesmo ocorre para número total de veículos utilizados, em que neste caso o desejável é obter o menor número possível.

Tabela 5.7. Resultados do algoritmo AAcr com diferentes configurações de restrições

Instância	3DR		F			MD			3D		
	# v	Dist. Total	# v	Dist. Total	d (%)	# v	Dist. Total	d (%)	# v	Dist. Total	d (%)
E016-03m	4	304,13	4	304,13	0,00	4	301,74	-0,79	4	282,96	-7,48
E016-05m	5	334,96	5	334,97	0,00	5	334,96	0,00	5	334,96	0,00
E021-04m	5	391,65	5	396,05	1,12	5	377,23	-3,82	5	373,92	-4,74
E021-06m	6	447,98	6	448,24	0,06	6	440,94	-1,60	6	430,89	-3,97
E022-04g	6	454,14	6	467,79	3,01	5	462,66	1,88	5	407,51	-11,44
E022-06m	6	499,07	6	499,08	0,00	6	504,23	1,03	6	499,08	0,00
E023-03g	6	865,77	5	775,87	-11,59	5	793,78	-9,07	5	732,52	-18,19
E023-05s	6	823,21	6	821,98	-0,15	5	781,52	-5,33	5	702,28	-17,22
E026-08m	8	645,14	8	658,41	2,06	8	630,13	-2,38	8	631,09	-2,23
E030-03g	8	849,33	7	838,61	-1,28	7	824,99	-2,95	6	739,66	-14,83
E030-04s	8	822,17	7	806,52	-1,94	7	788,99	-4,21	6	712,11	-15,46
E031-09h	9	625,92	9	621,56	-0,70	9	617,36	-1,39	9	614,24	-1,90
E033-03n	7	2.807,56	7	2.782,60	-0,90	7	2.639,11	-6,38	6	2.509,12	-11,89
E033-04g	8	1.494,67	8	1.499,10	0,30	8	1.456,81	-2,60	7	1.251,81	-19,40
E033-05s	8	1.467,70	7	1.438,38	-2,04	7	1.382,36	-6,17	6	1.210,69	-21,23
E036-11h	11	702,70	11	698,61	-0,59	11	698,61	-0,59	11	698,61	-0,59
E041-14h	14	879,57	14	878,14	-0,16	14	877,26	-0,26	14	864,79	-1,71
E045-04f	11	1.316,12	10	1.270,66	-3,58	10	1.237,85	-6,32	9	1.088,10	-20,96
E051-05e	11	823,48	11	780,02	-5,57	11	804,55	-2,35	9	690,58	-19,24
E072-04f	17	623,35	17	621,00	-0,38	17	608,90	-2,37	15	539,21	-15,61
E076-07s	17	1.168,79	18	1.185,67	1,44	16	1.081,78	-8,04	14	1.008,40	-15,91
E076-08s	19	1.252,62	17	1.220,85	-2,60	16	1.197,13	-4,64	14	1.085,64	-15,38
E076-10e	17	1.216,21	18	1.186,89	-2,47	17	1.150,64	-5,70	14	1.038,54	-17,11
E076-14s	16	1.193,12	16	1.164,17	-2,49	17	1.150,19	-3,73	15	1.071,76	-11,32
E101-08e	22	1.499,02	22	1.494,14	-0,33	23	1.431,71	-4,70	18	1.292,02	-16,02
E101-10c	26	1.782,83	26	1.736,01	-2,70	27	1.754,58	-1,61	21	1.424,92	-25,12
E101-14s	25	1.675,28	24	1.592,29	-5,21	24	1.578,57	-6,13	19	1.388,67	-20,64
Média		998,76		982,29	-1,36		959,58	-3.34		874,96	-12,21

Tabela 5.8. Resultados comparativos com diferentes configurações de restrições

Instância	F			MD			3D		
	GILM1	Aacsimr	% d	GILM1	Aacr	% d2	GILM1	Aacr	% d3
E016-03m	316,32	304,13	-4,01	301,74	301,74	0,00	297,65	282,96	-5,19
E016-05m	334,96	334,96	0,00	334,96	334,96	0,00	334,96	334,96	0,00
E021-04m	430,02	396,05	-8,58	392,44	377,23	-4,03	362,27	373,92	3,22
E021-06m	440,68	448,24	1,72	430,88	440,94	2,33	430,88	430,89	0,00
E022-04g	462,59	467,79	1,12	422,99	462,66	9,38	395,64	407,51	3,00
E022-06m	498,32	499,08	0,15	495,85	504,23	1,69	495,85	499,08	0,65
E023-03g	801,03	775,87	-3,24	732,51	793,78	8,36	742,23	732,52	-1,33
E023-05s	864,54	821,98	-5,18	810,65	781,52	-3,73	735,14	702,28	-4,68
E026-08m	677,06	658,41	-2,83	630,13	630,13	0,00	630,13	631,09	0,15
E030-03g	843,33	838,61	-0,56	827,11	824,99	-0,26	717,90	739,66	3,03
E030-04s	819,36	806,52	-1,59	767,22	788,99	2,84	718,24	712,11	-0,86
E031-09h	669,16	621,56	-7,66	635,16	617,36	-2,88	614,60	614,24	-0,06
E033-03n	2.753,91	2.782,60	1,04	2.549,68	2639,11	3,51	2.316,56	2.509,12	8,31
E033-04g	1.518,26	1.499,10	-1,28	1.389,31	1456,81	4,86	1.276,60	1.251,81	-1,98
E033-05s	1.414,19	1.438,38	1,71	1.346,44	1382,36	2,67	1.196,55	1.210,69	1,18
E036-11h	711,11	698,61	-1,79	703,57	698,61	-0,71	698,61	698,61	0,00
E041-14h	920,87	878,14	-4,87	906,42	877,26	-3,32	906,42	864,79	-4,81
E045-04f	1.433,51	1.270,66	-12,82	1.331,71	1237,85	-7,58	1.124,33	1.088,10	-3,33
E051-05e	853,05	780,02	-9,36	781,77	804,55	2,91	680,29	690,58	1,51
E072-04f	653,47	621,00	-5,23	629,77	608,90	-3,43	529,00	539,21	1,93
E076-07s	1.185,67	1.185,67	0,00	1.210,78	1081,78	-11,92	1.004,40	1.008,40	0,40
E076-08s	1.243,22	1.220,85	-1,83	1.160,67	1197,13	3,14	1.068,96	1.085,64	1,56
E076-10e	1.248,25	1.186,89	-5,17	1.153,60	1150,64	-0,26	1.012,51	1.038,54	2,57
E076-14s	1.187,80	1.164,17	-2,03	1.154,51	1150,19	-0,38	1.063,51	1071,76	0,78
E101-08e	1.673,08	1.494,14	-11,98	1.420,51	1431,71	0,79	1.371,32	1292,02	-6,14
E101-10c	1.775,52	1.736,01	-2,28	1.605,54	1754,58	9,28	1.557,12	1424,92	-9,28
E101-14s	1.662,10	1.592,29	-4,38	1.556,33	1578,57	1,43	1.378,52	1388,67	0,74
Média	1.014,50	982,29	-3,37	951,19	959,58	0,54	876,30	874,96	-0,32

Gendreau et al. (2006a) não divulgam o número de veículos utilizados quando as restrições do empacotamento tridimensional são seletivamente eliminadas. Assumindo que o número seja o mesmo quando todas as restrições são consideradas, os resultados comparativos com AAcr e GILM1 podem ser observados na Tabela 5.8. Em F, AAcr obtém uma diferença percentual de –3,37 % e utiliza 30 veículos a menos que GILM1. Em MD, existe uma diferença de 33 veículos a favor de AAcr, mas a distância percorrida média é ligeiramente superior. No que se refere aos resultados em 3D, cabe lembrar que GILM1 respeita as restrições de orientação, mas as caixas podem ter até 100% de projeção da base sem restrições de múltiplos destinos. Neste último caso, AAcr utiliza 74 veículos a menos que GILM1 e também consegue uma distância percorrida média levemente menor.

Capítulo 6

Conclusões e trabalhos futuros

6.1 Conclusões

O estudo apresentado neste trabalho refere-se a problemas de corte e empacotamento tridimensional e a integração destes com roteamento de veículos.

Inicialmente, foram abordados os problemas de carregamento de contêiner e *bin packing* 3D. Após a revisão bibliográfica, apresentou-se a fundamentação da escolha de heurísticas construtivas com múltiplos inícios para o problema de carregamento de contêiner. Basicamente, este método foi escolhido devido à dificuldade de implementação de procedimentos de busca local que atuem sobre a representação direta da solução, ou seja, o padrão de carregamento propriamente dito. Uma vez definido o método, uma nova abordagem baseada em cubóides de tamanho variável foi proposta. Nesta abordagem, o grau de ramificação da árvore construtiva é consideravelmente maior que o das abordagens com carregamento caixa a caixa ou com cubóides de tamanho fixo. Para contornar esta dificuldade, utilizou-se lista restrita de candidatos com escolha tendenciosa, conjunto de soluções elite, grau de reconstrução controlado e escolha adaptativa de parâmetros. As três últimas técnicas são diferentes formas do uso de memória adaptativa, que possui a função básica de evidenciar elementos que devem aparecer na construção das soluções. O algoritmo assim desenvolvido é inserido na implementação de um método de descida em vizinhança variável para tratar o problema *bin packing* 3D. O desempenho destes algoritmos foi avaliado a partir de diversos conjuntos de instâncias com diferentes grupos de restrições que usualmente aparece em

problemas reais. Os testes computacionais revelaram que o uso de cubóides de tamanho variável é uma abordagem efetiva, mesmo para conjuntos de caixas com um alto grau de heterogeneidade, e que o uso de heurísticas construtivas é uma alternativa simples e competitiva quando comparada com outras metaheurísticas disponíveis na literatura para estes problemas. As heurísticas apresentaram desempenho superior, quando comparadas com as demais disponíveis na literatura, e mostraram-se bastante robustas, com um ajuste de parâmetros não crítico. O uso do conjunto de soluções elite foi particularmente útil para instâncias fracamente heterogêneas, enquanto diferentes graus de reconstrução apresentaram melhor desempenho quando aplicados a instâncias fortemente heterogêneas.

Nas implementações de heurísticas para estes problemas, grande parte do tempo computacional é gasto na identificação dos espaços vazios, principalmente quando restrições de cunho prático são levadas em consideração. Com relação a este fato, o uso da representação espacial com matrizes bidimensionais mostrou-se importante para agilizar a identificação dos espaços vazios, bem como facilitar a incorporação de restrições. A partir da observação que as instruções para carregamento não necessariamente precisam seguir a ordem com a qual a solução é construída, foi proposta uma identificação simultânea de espaços vazios dos fundos para frente e da frente para os fundos do contêiner.

Por último, abordou-se o problema da integração de empacotamento tridimensional com roteamento de veículos, para o qual foi proposto um algoritmo de busca tabu que invoca os métodos desenvolvidos anteriormente para carregamento de contêiner. O problema de roteamento de veículos capacitados é largamente estudado na literatura, no entanto, do conhecimento do autor, somente o trabalho de Gendreau et al. (2006a) formula o problema de modo a inserir restrições de empacotamento tridimensional. O algoritmo de busca tabu proposto neste trabalho apresentou melhor desempenho tanto para a distância total percorrida como número de veículos utilizados. Muito embora a formulação considere um número fixo de veículos, os resultados comparativos com outras abordagens e com diferentes configurações do conjunto de restrições de empacotamento revelaram que o procedimento que carrega as caixas é essencial não só para minimizar a distância total percorrida como também o número de veículos utilizados.

6.2 Trabalhos futuros

O campo de pesquisa para problemas de corte e empacotamento tridimensional, integrados ou não com roteamento de veículos, é muito vasto. A partir do estágio alcançado pela pesquisa desenvolvida na presente tese, algumas extensões são possíveis para a sua continuidade:

1. Incorporar outras restrições de cunho prático além das já estudadas.

2. Com base nos resultados obtidos, é interessante considerar a aplicação dos métodos desenvolvidos em problemas correlatos, como carregamento de múltiplos contêineres, *strip packing* 3D e programação de projetos com disponibilidade restrita de recursos.

3. Estudar uma formulação bi-objetivo hierárquico do PRVC3D e restrições adicionais para o roteamento como, por exemplo, janelas de tempo.

4. Gerar instâncias teste com um número maior de caixas por clientes e com volume vinculado, de alguma forma, ao peso total da demanda.

Referências Bibliográficas

Ahmadi, S. e Osman, I. H. 2003. Greedy adaptive memory programming search for the capacitated clustering problem, working paper.

Armentano, V. A. e Araújo, O. C. B. 2006. Grasp with memory-based mechanisms for minimizing total tardiness in single machine scheduling with setup times, *Journal of Heuristics*, 12 (6), pp. 427-446.

Baker, B. S., Coffman Jr., E. G. e Rivest, R. L. 1980. Orthogonal packing in two dimensions, *SIAM Journal on Computation*, 9, pp. 846-855.

Ballou, R. H. 1999. *Logística empresarial*, Sonopress, Porto Alegre.

Binato, S., Hery J., Loewenstern, D. M e Resende, M. G. C. 2002. A greedy randomized adaptive search procedure for job shop scheduling, in *Essays and Surveys on Metaheuristics Essays and Surveys on Metaheuristics*, pp. 58-79, C.C. Ribeiro and P. Hansen, Eds., Kluwer.

Bischoff, E. E. 2006. Three-dimensional packing of items with limited load bearing strength, *European Journal of Operational Research*, 168 (3), pp 952-966.

Bischoff, E. E. e Dowsland, W. B. 1982. An Application of the Micro to Product Design and Distribution. *Operational Research Society Journal*, 33, pp. 271-280.

Bischoff, E. E. e Marriott, M. D. 1990. A Comparative Evaluation of Heuristics for Container Loading. *European Journal of Operational Research,* 44, pp. 267-276.

Bischoff, E. E. e Ratcliff, M. S. W. 1995a. Issues in the development of approaches to container loading", *Omega*, 4, pp 377-390.

Bischoff, E. E., Janetz, F. e Ratcliff, M. S. W. 1995. Loading Pallets with Non-Identical Items*. European Journal of Operational Research,* 84, pp. 681-692.

Blum, C. e Andrea, R. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Computing Survey, 35 (3), pp. 268-308.

Bortfeldt, A. e Gehring, H. 1998. Ein Tabu Search-Verfahren für Containerbeladeprobleme mit schwach heterogenem Kistenvorrat. *OR Spektrum*, 20 (4), pp. 237-250.

Bortfeldt, A. e Gehring, H. 1999. Two Metaheuristics for Strip Packing Problems. In: Despotis, D.K.; Zopounidis, C. (Hrsg.): *Proceedings of the 5th International Conference of the Decision Sciences Institute*, Athen, 2, pp. 1153-1156.

Bortfeldt, A. e Gehring, H. 2001. A Hybrid Genetic Algorithm for the Container Loading Problem. *European Journal of Operational Research*, 131, pp. 143-161.

Bortfeldt, A., Gehring, H. e Marck D. 2003. A parallel tabu search algorithm for solving the container loading problem, *Parallel Computing,* 29 (5), pp. 641-662.

Bresina, J. 1996. Heuristic-Biased Stochastic Sampling, *In Proceedings of the 13th national Conference on Artificial Intelligence,* pp. 271-278.

Carpenter, H. e Dowsland, W. B. 1985. Practical Considerations of the pallet loading problem, *Journal of Operational Research Society*, 36, pp. 489-497.

Carvalho, M. H., Cerioli, M. R., Dahab, R., Feofiloff, P., Fernandes, C. G., Ferreira, C. E., Guimarães, K. S., Miyazawa, F. K., Pina, J. C., Soares, J. e Wabayashi, Y. 2001. *Uma Introdução Sucinta a Algoritmos de Aproximação*, livro do Colóquio Brasileiro de Matemática.

Cecilio, F. O. 2003. *Heurísticas para o problema de carregamento de carga dentro de contêineres*, São Carlos - SP, Dissertação de Mestrado, UFSCar.

Cecilio, F. O. e Morabito, R. 2004. Refinamentos na heurística de George e Robinson para o problema de carregamento de caixas dentro de contêineres, *Transportes*, 12 (1), pp. 32-45.

Chen, C. S., Lee, S. M. e Shen, Q. S. 1995. An Analytical Model for the Container Loading Problem, *European Journal of Operational Research*, 80, pp.68-76.

Chiang, W. C. e Russell, R. A. 1996. Simulated annealing metaheuristics for the vehicle routing problems with time windows, *Annals of Operations Research,* 63, pp. 3-27.

Chien C. F. e Deng J. F. 2004. A container packing support system for determining and visualizing container packing patterns, *Decision Support Systems*, 37 (1), pp. 23-34.

Chien, C. F. e Wu, W. T. 1999. A framework of modularized heuristics for determining the container loading patterns, *Computers & Industrial Engineering*, 37 (1-2), pp. 339-342.

Christofides, N. e Whitlock, C. 1977. An algorithm for two-dimensional cutting problems, *Operations Research*, 25 (1), pp 30-44.

Clarke, G. e Wright, J. W. 1964. Scheduling of vehicles from a central depot to a number of delivery points, *Operations Research*, 12, pp. 568-581.

Corcoran, A. L. e Wainwright, R. L. 1992. A Genetic Algorithm for Packing in Three Dimensions, in *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, ACM Press, pp. 1021-1030,.

Cordeau, J-F, Gendreau, M. e Laporte, G. 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problems, *Networks*, 30, pp. 105-119.

Davies, A. P. e Bischoff, E. E. 1999. Weight Distribution Considerations in Container Loading. *European Journal of Operational Research*, 114, pp.509-527.

Dowsland, K. A. e Dowsland, W. B. 1992. Packing Problems. *European Journal of Operational Research*, 56, pp. 2-14.

Duin, C. W. e Voss, S. 1999. The Pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs, *Networks*, 34, pp. 181-191.

Dyckhoff, H. 1990. A typology of cutting and packing problems, *European Journal of Operational Research*, 44, pp. 145-159.

Eley, M. 2002. Solving container loading problems by block arrangement, *European Journal of Operational Research*, 141 (2), pp. 393-409

Eley, M. 2003. A botleneck assignment approach to the multiple container loading problem, *OR Stpectrum*, 45, pp 45-60.

Faroe O., Pisinger D. e Zachariasen M. 1999. Guided Local Search for the Three-Dimensional Bin-Packing Problem, Technical paper, DIKU, University of Copenhagen.

Faroe O., Pisinger D. e Zachariasen M. 2003. Guided Local Search for the Three-Dimensional Bin-Packing Problem, *INFORMS Journal on Computing*, 15 (3), pp. 267-283.

Fleurent, C. e Glover, F. 1999. Improved constructive mult-start strategies for the Quadratic assignment problem using adaptive memory, *INFORMS Journal on Computing*, 11(2), pp.198-204.

Fukasawa, R., Longo, H., Lysgaard, J. L, Poggi de Aragão, M., Reis, M., Uchoa, E. e Werneck, F. 2004. Robust branch-and-cut-and-price for the capatitated vehicle routing problem. *Proceedings X IPCO,* Springer Lecture Notes in Computer Science, New York, pp. 1-15.

Gehring, H. e Bortfeldt, A. 1997. A Genetic Algorithm for Solving the Container Loading Problem. *International Transactions in Operations Research*, 4 (5-6), pp. 401-418.

Gehring, H. e Bortfeldt, A. 2002. A parallel genetic algorithm for solving the container loading problem, *International Transactions of Operational Research*, 9 (4), pp. 497-511.

Gehring, H., Menschner, K. e Meyer, M. 1990. A Computer-Based Heuristic for Packing Pooled Shipment Containers. *European Journal of Operational Research*, 44, pp. 277-288.

Gendreau, M., Hertz, A. e Laporte, G. 1992. New insertion and postoptimization procedures for the trabeling salesman problem, *Operations Research*, 40, pp. 1086-1094.

Gendreau, M., Hertz, A. e Laporte, G. 1994. A Tabu Search Heuristic for the Vehicle Routing Problem, *Management Science*, 40, pp. 1276-1290.

Gendreau, M., Iori, M., Laporte, G. e Martello, S. 2006a. A Tabu Search Algorithm for a Routing and Container Loading Problem, *Transportation Science*, 40, pp. 342-350.

Gendreau, M., Iori, M., Laporte, G. e Martello, S. 2006b. A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints, *Networks*, submetido.

George, J. A. e Robinson, D. F. 1980. A Heuristic for Packing Boxes into a Container, *Computers and Operations Research*, 7, pp.147-156.

Gilmore, P. C. e Gomory, R. 1961. A linear programming approach to the cutting-stock problem, *Operations Research*, 9 (6), pp. 848-859.

Gilmore, P. C. e Gomory, R. 1963. A linear programming approach to the cutting-stock problem – part II, *Operations Research*, 11 (6), pp. 863-888.

Glover, F. 1986. Future Paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research*, 13 (5), pp. 533-549.

Glover, F. 2000. Multi-Start and Strategic Oscillation Methods - Principles to Exploit Adaptive Memory, Computing Tools for Modeling, Optimization and Simulation: Interfaces in *Computer Science and Operations Research*, M. Laguna and J.L. Gozales Velarde, Eds., Kluwer Academic Publishers, pp. 124.

Haessler, R. W. e Talbot, F. B. 1990. Load Planning for Shipments of Low Density Products. *European Journal of Operational Research*, 44, pp. 289-299.

Han, C. P., Knott, K. e Egbelu, P. J. 1989. A Heuristic Approach to the Three-Dimensional Cargo Loading Problem. *International Journal of Production Research*, 27 (5), pp. 757-774.

Hansen, P. Mladenovic, N., 2001. Variable neighborhood search: Principles and applications. *European Journal of Operational Research,* 130, pp. 449-467.

Hansen, P. e Mladenovic, N. 1999. An introduction to variable neighbourhood search, in *Metaheuristics: Advances and trends in local search procedures for optimization*, Vos, S., Martello, S., Osman, I., Roucairol, C., Eds., Kluwer Academic Publishers, pp. 433-458.

Hansen, P. e Mladenovic, N. 2003. Variable neighbourhood search, in *Handbook of metaheuristics,* Glover, F., Kochenberger, G., Eds., Kluwer Academic Publishers, pp. 145-184.

Hassamontr, J. 2003. On decomposing 3D Packing Problem in Wooden Furniture Industry, *In Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp 497-502.

He, D. Y. e Cha, J. Z. 2002. Research on Solution to Complex Container Loading Problem Based on Genetic Algorithm, In: *Proceeding of First International Conference on Machine Learning and Cybernetics*, 1, pp. 78-82, Beijing, November.

Hifi, M. 2004. Exact algorithms for unconstrained three-dimensional cutting problems: a comparative study, *Computers and Operations Research,* 31 (5), pp. 657-674.

Iori M. 2004. *Metaheuristic algorithms for combinatorial optimization problems*, Bologna – Itália, Tese de Doutorado, DEIS – Universidade de Bologna.

Ivancic, N., Mathur K., e Mohanty B. B. 1989. An integer programming based heuristic approach to the three-dimensional packing problem. *Journal of Manufacturing and Operations Management,* 2, pp. 268-298.

Joslin, D. e Clements, D. 1999. "Squeaky wheel Optimization", *Journal of Artificial Intelligence Research*, 10, pp. 353-373.

Lai, K. K., Xue, J. e Xu, B. 1998.Container packing in a multi-customer delivering operation, *Computers & Industrial Engineering*, 35 (1-2), pp. 323-326.

Laporte, G., Gendreau, M., Potvin, J-Y. e Semet, F. 2000. Classical and moderns heuristics for the vehicle routing problem, *International Transactions in Operatonal Research,* 7, pp. 285-300.

Li, K. e Cheng, K. H. 1992. Heuristic algorithms for on-line packing in three dimensions, *Journal of algorithms*, 13, pp. 589-605.

Lim, A. e Zhang, X. 2005. The container loading problem, *ACM Symposium on Applied Computing*, pp. 913-917.

Lim, A., Rodrigues, B. e Wang Y. 2003. A multi-faced buildup algorithm for three-dimensional packing problems, *Omega*, 31 (6), pp. 471-481.

Lim, A., Rodrigues, B. e Yang, Y. 2005. 3-d container packing heuristics, *Applied Intelligence*, 22 (2), pp. 125-135.

Lodi A., Martello S. e Vigo D. 1999. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, *INFORMS Journal on Computing*, 11, pp. 345-357.

Lodi A., Martello S. e Vigo D. 2002. Heuristic algorithms for the three-dimensional bin packing problem, *European Journal of Operational Research*, 141 (2), pp 410-420.

Lodi A., Martello S. e Vigo D. 2004. TSpack: A Unified Tabu Search Code for Mult-Dimensional Bin Packing Problems, *Annals of Opertions Research*, 131, pp 203-213.

Loh, T. H. e Nee, A. Y. C. 1992. *A packing algorithm for hexahedral boxes*, Proceedings of the Conference of Industrial Automation, Singapore, pp. 115-126.

Mack, D., Bortfeldt, A. e Gehring, H. (2004). A parallel hybrid local search algorithm for the container loading problem. Accepted by *ITOR* (3/2004).

Martello, S. e Toth, P. 1990. *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester.

Martello, S., Pisinger, D. e Vigo, D. 2000. The Three Dimensional Bin Packing Problem, *Operations Research*, 48, pp. 256-267.

Martí, R. 2003. Multi-Start Methods, Handbook of MetaHeuristics, Glover and Kochenberger Eds., Kluwer, pp. 355-368.

Miyazawa, F. K. e Wakabayashi Y. 1997. An algorithm for the three-dimensional packing problem with asymptotic performance analysis, *Algorithmica*, 18 (1), pp. 122-144.

Miyazawa, F. K. e Wakabayashi Y. 2000. Approximation algorithms for the orthogonal *z*-oriented 3-D packing problem, *SIAM J. Computing*, 29(3), pp. 1008-1029.

Miyazawa, F. K. e Wakabayashi Y. 2004. Packing Problems with Orthogonal Rotations. LATIN'2004: Theoretical Informatics. *Lecture Notes in Computer Science*, LNCS 2976, pp. 359-368, *Springer-Verlag*. Buenos Aires, Argentina.

Mladenovic, N. e Hansen, P. 1997. Variable Neighborhood Search, *Computers and Operations Research*, 24, pp. 1097-1100.

Montgomery, J., Randall, M. e Hendtlass, T. 2004. Search Bias in Constructive Metaheuristics and Implications for Ant Colony Optimisation. *Proceedings of the Fourth International Workshop on Ant Algorithms*, ANTS2004, Brussels, Belgium.

Morabito, R. e Arenales, M. 1994. An And/Or-graph Approach to the Container Loading Problem. *International Transactions in Operations Research,* 1 (1), pp.59-73.

Moura, A. e Oliveira, J. F. 2005. A GRASP Approach to the Container-Loading Problem, *IEEE Intelligent Systems*, 4(20), pp. 50-57.

Nelder, J. A. e Mead, R. 1965. A Simplex Method for Function Minimization, *Computer Journal*, 7, pp. 308-313.

Ngoi, B. K. A., Tay, M. L. e Chua, E. S. 1994. Applying spatial representation techniques to the container packing problem, *International Journal of Production Research*, 32, pp. 111-123.

Nilsson, N. 1982. *Principles of Artificail Intelligence*, Springer-Verlang.

Patterson, R., Pirkul, H. e Rolland, E. 1999. Adaptive Reasoning Technique for the Capacitated Minimum Spanning Tree Problem, *Journal of Heuristics,* 5, pp. 159-180.

Pearl, J. 1984. *HeuristicsI*: *Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley Publishing Company, 2a. Ed.,

Pisinger, D. 2002. Heuristics for the container loading problem, *European Journal of Operational Research*, 141, pp. 143-153.

Prais, M. e Ribeiro, C. C. 2000. Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment, *INFORMS Journal on Computing*, 12, pp. 164-176.

Reimann, M., Doerner, K. e Hartl, R. F. 2004. D-Ants: Savings Based Ants divide and conquer the vehicle routing problem, *Computers and Operations Research*, 4, pp. 563-591.

Rodrigues Neto, L., L. 2005. *Um algoritmo genético para solução do problema de carregamento de container,* Rio de Janeiro – RJ, Dissertação de Mestrado, COPPE /UFRJ.

Russel, R. A. 1995. Hybrid Heuristic for the vehicle routing problem with time windows, *Transportation Science*, 29(2), pp. 156-166.

Silva, J. L. C. e Soma, N. Y., 2003. Um algoritmo polinomial para o problema de empacotamento de contêineres com estabilidade estática da carga, *Pesquisa Operacional,* 23 (1), pp. 79-98.

Steudel, H. 1979. Generating pallet loading patterns: A special case of the two-dimensional cutting stock problem, *Management Science*, 10, pp. 997-1004.

Toth, P. e Vigo, D. 2002. *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematicsand Applications, Philadelphia.

Toth, P. e Vigo, D. 2003. The Granular Tabu Search and its Application to the Vehicle Routing Problem, *Informs Journal on Computing*, 15(4), pp. 333-346.

Verweij, A. M. 1996. Multiple destination bin packing. UU-CS (Ext. r. no. 1996-39). Utrecht, the Netherlands: Utrecht University: Information and Computing Sciences.

Wassan, N. A. 2006. A reactive tabu search for vehicle routing problem, *Journal of the Operational Research Society,* 57 (1), pp. 111-116.

Apêndice

A seguir é transcrito na íntegra o artigo intitulado “A Multi-start Random Constructive Heuristic for the Container Loading Problem” submetido para publicação na revista Pesquisa Operacional.

A MULTI-START RANDOM CONSTRUCTIVE HEURISTIC FOR THE CONTAINER LOADING PROBLEM

Olinto César Bassi de Araújo *
Colégio Politécnico da Universidade Federal de Santa Maria
Universidade Federal de Santa Maria
olinto@smail.ufsm.br

Vinícius Amaral Armentano
Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas
vinicius@densis.fee.unicamp.br
* Corresponding author/autor para quem as correspondências devem ser encaminhadas

Abstract

This paper deals with the container loading problem which involves the selection of a subset of boxes, each box with a given volume, such that they fit in a single container and maximize its volume utilization subject to orientation and stability constraints. We propose a multi-start random constructive heuristic with a load arrangement that is based on maximal cuboids that fit in given empty spaces. Each instance is adaptively evaluated by a set of criteria, and at each step of the construction process one maximal cuboid is chosen probabilistically from a restricted list of candidates. In order to enhance the flexibility in the construction of a solution, a probabilistic reduction on such cuboids is allowed. Computational tests on several instances from the literature show that the proposed method performs better than other approaches.

Keywords: container loading; cuboid arrangement; multi-start random constructive heuristic.

Resumo

Neste trabalho abordamos o problema de carregamento de contêiner que trata da seleção de um subconjunto de caixas, cada caixa com um dado volume, de forma a maximizar o volume ocupado de um único contêiner sujeito a restrições de orientação e estabilidade. Propomos uma heurística construtiva aleatória com múltiplos inícios que utiliza um arranjo de carga baseado em cubóides que maximizam a ocupação de espaços vazios. Cada instância é avaliada de forma adaptativa por um conjunto de critérios, e em cada passo do processo construtivo um cubóide é selecionado probabilisticamente de uma lista restrita de candidatos. Para aumentar a flexibilidade na construção de uma solução, permite-se uma redução probabilística no tamanho dos cubóides. Resultados computacionais em instâncias da literatura

mostram que o método proposto apresenta um desempenho superior a outros enfoques sugeridos na literatura.

Palavras-chave: carregamento de contêiner; arranjo com cubóides; heurística construtiva aleatória com múltiplos inícios.

1. Introduction

This paper addresses the problem of optimizing the loading of rectangular boxes of different sizes into a rectangular container of given dimensions so that their edges lie parallel to the edges of the container and that no two items overlap. This is one of the problems with numerous applications in the cutting and packing industry and a general classification of such problems is provided by Dyckhoff (1990). The problem is particularly important in companies whose logistic activities involve storage, distribution and/or collection of goods, for a better space utilization allows reduction of cost and time in loading and unloading containers.

Three-dimensional cutting and packing problems are extensions of their one-dimensional counterparts, and therefore belong to the NP-hard class. This implies that most likely optimal methods are not able solve real problems in a reasonable time, and for this reason the literature on optimal methods and approximate algorithms is scarce, whereas the literature on heuristic methods is fairly vast.

Chen *et al.* (1995) propose a mixed integer linear programming for the problem of loading multiple containers. In addition to constraints that avoid overlapping, constraints that control the weight imbalance along one of the dimensions are also modeled. The model is tested on one instance with one container and six boxes. Lai *et al*. (1998) propose a graph-based model for the loading problem with multiple costumers orders such that cargoes belonging to the same costumer are packed together in the container. An exact algorithm and a heuristic are proposed for solving the model. Martello *et al.* (2000) propose a branch-and-bound algorithm for loading a single container, which is then used in an exact algorithm for the three-dimensional bin packing. Hifi *et al*. (2004) suggest two optimal algorithms for solving unconstrained three-dimensional cutting problems, in which there is an unlimited quantity of pieces of each type to be cut.

Heuristic approaches represent viable alternatives to obtain good solutions for practical problems in a reasonable time. Pisinger (2002) classifies heuristic approaches according to the loading building pattern, namely wall building, stack building, guillotine cutting, and cuboid arrangement.

The wall building approach constructs vertical or horizontal layers which reduce the solution space and allows the use of simple data structure in the implementation of algorithms.

Such an approach was introduced by George & Robinson (1980) who suggested a sophisticated constructive heuristic based on vertical layers such that spaces not occupied in a layer can be use in subsequent layers. The ideas proposed by George & Robinson (1980) are the base for heuristics developed by Bischoff & Marriot (1990), Gehring *et al.* (1990), Bortfeldt & Gehring (2001), Pisinger (2002), Cecílio & Morabito (2004) and Moura & Oliveira (2005). Bischoff *et al.* (1995) and Lim & Zhang (2005) use horizontal layers in order to build the loading pattern.

The stack building approach allows the decomposition of the original problem into two subproblems: the three-dimensional problem of packing the boxes into suitable stacks and the two-dimensional problem of locating the stacks at the floor of the container. A stack is built from the selection of a base box that is positioned at the floor of the container. The next box that is placed in the stack must have its base fully supported by one or more the boxes that lie below it, as shown in Figure 1. This approach favors the treatment of weight constraints, but in general gives rise to loading patterns with poor horizontal stability, and when the cargo is weakly heterogeneous it results in a low utilization of the container space. The use of this approach can be found in the heuristics proposed by Haessler & Talbot (1990), Gehring & Bortfeldt (1997).

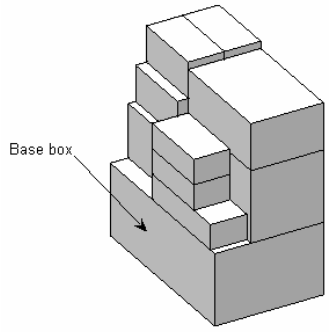


Figure 1. Example of stack packing

A guillotine cut is a constraint imposed to the problem, i.e., each cut on any parallelepiped produces two new parallelepipeds. In the guillotine cutting approach all boxes can be distinguished as a sequence of guillotine cuts. Hassamontr (2003) develops a software tool for three-dimensional cuts in a wooden furniture industry. Morabito & Arenales (1994) propose a heuristic which makes use of guillotine cuts, however, this constraint is unusual in three-dimensional packing problems, unlike in two-dimensional cutting problems.

In the cuboid arrangement, the container is filled by homogeneous blocks made up of boxes of the same type and with identical orientation. Bortfeld & Gehring (1998) propose a heuristic that makes use of local arrangements with one or two blocks. A parallel version of this heuristic is suggested in (Bortfeldt *et al.*, 2003). Eley (2002) points out that the ease of

arrangement and lower complexity in terms of load bearing strength are some of the advantages of this approach.

Ngoi *et al.* (1994) develop a spatial representation technique to model the loading process which allows the evaluation of all potential placement locations. Such a representation is used by Bischoff & Ratcliff (1995), Chien & Deng (2004) and an adaptation of the representation is utilized by Bischoff (2006). Lim *et al.* (2003) propose a heuristic in which the walls of the container are used as the ground to build up a load arrangement, called multi-faced buildup.

Real life container loading problems are complex and usually other constraints and/or objectives must be taken into account, in addition to the overlapping constraint and the volume maximization. For example, loading stability and weight distribution are important factors in some applications. Bischoff & Ratcliff (1995) list twelve factors that play an important role in container loading problems. Gehring *et al.* (1990) suggest a load arrangement that takes into account constraints in weight distribution. Gehring & Bortfeldt (1997) address the problem with several constraints, namely, orientation, load bearing, maximum weight, stability and weight distribution. Davies & Bischoff (1999) propose a heuristic that is able to produce loading arrangements which combine high space utilization with an even weight distribution of the cargo. Bortfeldt *et al.* (2003) and Mack *et al.* (2004) consider constraints of orientation and stability in terms of sections of the cargo that overhang beyond the edge of the box(es) supporting it. He & Cha (2002) consider the sum of weighted objectives involving volume maximization, weight maximization and the minimization of the height of the gravity center. Bischoff (2006) develops an algorithm for tackling problems where the load bearing strength of the cargo is a key factor.

In this paper we put forward a multi-start random constructive heuristic for the container loading problem with a load arrangement that is based on maximal cuboids that fit in given empty spaces, such that they fit in a single container and maximize its volume utilization subject to orientation and stability constraints.

At each step of the construction process one maximal cuboid is chosen probabilistically from a restricted list of candidates. In order to enhance the flexibility in the construction of a solution, we allow a probabilistic reduction on such cuboids. This approach can be viewed as a generalization of procedures that load one box at a time and the cuboid approach. The paper contains five sections. The following section defines the problem in more detailed terms. Section 3 of the paper describes the proposed multi-start heuristic, and in section 4 the heuristic performance is tested on standard benchmarks and additional instances suggested in the literature. Conclusions are presented in section 5.

2. Problem Description

The container loading problem involves the selection of a subset of boxes, each box with a given utility, such that they fit in a single container and maximize the total utility subject to a set of constraints. In this paper the utility of each box is its volume, and constraints of concern are orientation and stability. The boxes have $k = 1,...,6$ orientations and are grouped in m types, each type t characterized by three spatial dimensions $d_{1tk}, d_{2tk}, d_{3tk}$, a volume v_t and a number q_t of boxes, $t = 1, ..., m$. The set of boxes is said to be homogeneous if it has a single type, while it is weakly heterogeneous or strongly heterogeneous if the number of types is small or large relative to the total number of items, respectively. Without loss of generality the dimensions of the container and of the boxes are positive integers. Constraints related to the orientation of the boxes are represented by a set of binary parameters u_{tk}, such that $u_{tk} = 1$ if the orientation a box of type t and an orientation k is allowed and $u_{tk} = 0$, otherwise. The stability constraint specifies that all loaded boxes are fully supported by the container floor or one or more the boxes.

In order to locate boxes in the container, consider a coordinate system within it such that when the container is viewed from the front, its origin is the bottom left corner of the container back, and the three coordinates refer to positions along the length, width and height, in that order. An empty space i in the container corresponds to a parallelepiped of dimensions $X_i \times Y_i \times Z_i$ and volume ve_i, whose coordinates (x_i, y_i, z_i) are associated to its lower left back vertex. When the container is empty, there exists a single empty space with the container dimensions and coordinates (0, 0, 0).

Figure 2 shows that for each box placed in the container, three new empty spaces are created: the depthwise, widthwise, and heightwise spaces. The intersection space of the depthwise and widthwise empty spaces is not considered as an empty space since it is contained in these spaces. An empty space is discarded if its minimum dimension is less than the minimum dimension of any unloaded box.

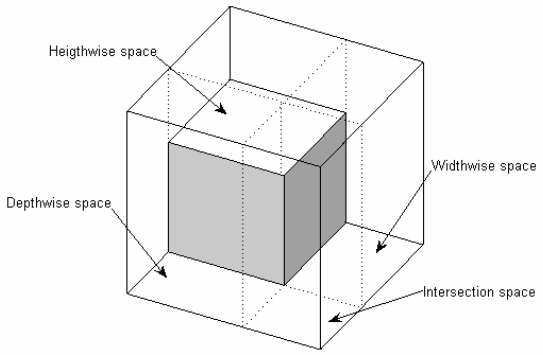
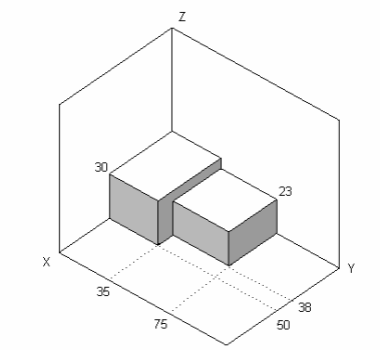


Figure 2. Spaces generated by the loading of one box

The proposed heuristic makes use of an adaptation of the spatial representation suggested by Ngoi *et al.* (1994), in which a two-dimensional matrix represents a view from the top of the container and contains cells that correspond to the height of potential loading surfaces. The search for available loading spaces amounts to scanning this matrix for contiguous surfaces at same height. The adaptation is similar to that proposed by Bischoff (2006). The spatial representation technique provides excellent flexibility in generating the packing sequence and in identifying all potential placement locations, unlike the wall approach building used in several heuristics. In this paper, the identification of empty spaces in the container is performed in a single direction, from backward to forward, and in both directions, from backward to forward and from forward to backward.

Figure 3 illustrates an example of such a representation for a container with dimensions $90\ \text{cm} \times 120\ \text{cm} \times 100\ \text{cm}$ and two boxes of dimensions $50\ \text{cm} \times 35\ \text{cm} \times 30\ \text{cm}$ (box 1) and $38\ \text{cm} \times 40\ \text{cm} \times 23\ \text{cm}$ (box 2), with coordinates $(0,0,0)$ and $(0,35,0)$, respectively. The first row and first column of the matrix contain the values of the projections of all vertical box/container edges onto the two horizontal axes, with the exception of cell (1, 1) that represents the height of the container. The figures of 35, 75 and 120 in row 1 correspond to the width of box 1, the sum of the widths of the two boxes, and the container width (projections of the edges onto the *y* axis). Analogously, the figures of 38, 50 and 90 represent the length box 2, the length of box 1 and the container length (projections of the edges onto the *x* axis). The remaining cells correspond to the heights of the surfaces: cells (2, 2) and (3, 2) correspond to the height of box 1 and cell (2, 3) represents the height of box 2.



Matrix of surfaces

100	35	75	120
38	30	23	0
50	30	0	0
90	0	0	0

Figure 3. Representation of loading surfaces

3. A Multi-start Randomized Constructive Heuristic

In this paper we propose a multi-start constructive heuristic for the container loading problem. Multi-start heuristics have been used in combinatorial optimization since the seminal paper by Lin & Kernighan (1973) for the symmetric traveling salesman and they represent a strategy to achieve search diversification in the solution space. One of the most well known multi-start methods is the greedy random adaptive search procedure (GRASP), which was introduced by Feo & Resende (1995). GRASP is a meta-heuristic that has been applied with success to solve a variety of combinatorial problems (Festa & Resende, 2004), and enhancement techniques to the basic GRASP are discussed in Resende & Ribeiro (2003). Each GRASP iteration consists of two phases: construction and local search. The construction phase builds a feasible solution by probabilistically selecting the next element to be incorporated in a partial solution from a restricted candidate list (RCL) composed of the best elements, as measured by a greedy function. Local search is then applied to the constructed solution until a local optimum is found. This process is repeated for a number of iterations, and the best local optimum is selected.

The proposed heuristic is based only on the construction phase due to the difficulty of devising a restricted neighborhood with promising moves that operate directly on the constructed solution. To the best of our knowledge, Faroe *et al.* (2003) is the only work that performs a local search directly on the loading solutions for a three-dimensional loading problem, which in this case is the three-dimensional bin packing problem. The neighborhood of a solution consists of all solutions that can be obtained by translating any single box along one of the coordinate axes or to the same position in another bin. In this way, solutions with overlapping of boxes are generated and hence the objective is to minimize the total volume of the pairwise overlap between boxes. An obvious drawback of this strategy is that the orientation of boxes of neighbor solutions does not change. The remaining papers that make use of local search for the container loading problem use a codification of the solution, and a

neighbor is defined by a move applied to the coded solution (Bortfeldt & Gehring, 1998, Bortfeldt & Gehring, 2001, Bortfeldt *et al*., 2003, Moura & Oliveira, 2005). Martí (2003) describes the best known multi-start heuristics and also remarks that for some problems it is more effective to construct solutions than apply a local search procedure.

The proposed heuristic has some similar features with the two-step heuristic suggested by Eley (2002), which can be summarized as follows. In the first step, a greedy heuristic builds a solution by initially sorting the boxes by decreasing volume. All possible empty spaces for stowing the next box are examined, and this box is placed in the empty space where the sum of the volume of spare spaces that cannot filled by the remaining boxes is minimal. The second step of the heuristic consists of a tree search in which the root node represents an empty container and each further node that is not a leaf node constitutes a partially filled container. If all orientations are permitted for each box type at a given node, then each partial solution is branched into at most $6 \cdot m$ new partial solutions. A best search strategy is applied, which expands a number (a parameter) of nodes that obtained the highest ranking from the evaluation function, which is a lower bound derived by filling the remaining space of the corresponding partial solution by applying the greedy heuristic.

3.1 Heuristic Description

Cuboids are homogeneous blocks composed of boxes of the same type and orientation. A cuboid is represented by a triple $c_{tki} = (c^x_{tki}, c^y_{tki}, c^z_{tki})$, such that its elements denote the number of boxes of type *t* with orientation *k* in the coordinates *x, y, z*, that can be loaded in the empty space *i*. The number of boxes that form a cuboid is given by $|c_{tki}| = c^x_{tki} \cdot c^y_{tki} \cdot c^z_{tki}$. Figure 4 shows a cuboid $c_{tki} = (2, 2, 3)$ with $|c_{tki}| = 12$ boxes of type *t* and orientation *k*, with two boxes along its length and width and three boxes along its height. A single box is a cuboid with representation (1, 1, 1), and the volume of the cuboid is the product of the volume of the box type and the number of boxes that make up the cuboid, i.e., $v_t \cdot |c_{tki}|$.

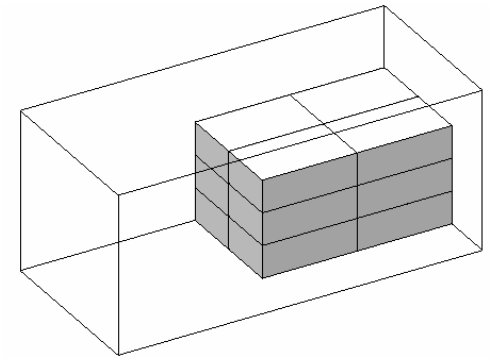


Figure 4. A cuboid (2, 2, 3) composed by boxes of the same type and orientation

Let $\overline{c}_{tki} = (\overline{c}^{x}_{tki}, \overline{c}^{y}_{tki}, \overline{c}^{z}_{tki})$ be the largest cuboid formed by boxes of type t and orientation k which can be assigned to an empty space i with dimensions $X_i \times Y_i \times Z_i$ and volume ve_i. The number of boxes which determines each dimension of the cuboid $\overline{c}_{tki}$ is calculated as follows:

$$\overline{c}^{z}_{tki} = \min\left(\left\lfloor \frac{Z_i}{d_{3tk}} \right\rfloor, \hat{q}_t\right) \quad (1)$$

$$\overline{c}^{y}_{tki} = \min\left(\left\lfloor \frac{Y_i}{d_{2tk}} \right\rfloor, \left\lfloor \frac{\hat{q}_t}{\overline{c}^{z}_{tki}} \right\rfloor\right) \quad (2)$$

$$\overline{c}^{x}_{tki} = \min\left(\left\lfloor \frac{X_i}{d_{1tk}} \right\rfloor, \left\lfloor \frac{\hat{q}_t}{\overline{c}^{z}_{tki}\,\overline{c}^{y}_{tki}} \right\rfloor\right), \quad (3)$$

where $\hat{q}_t$ denotes the number of boxes of type *t* that have not been loaded in the container.

Any cuboid c_{tki} such that $1 \le c^{x}_{tki} \le \overline{c}^{x}_{tki}$, $1 \le c^{y}_{tki} \le \overline{c}^{y}_{tki}$, and $1 \le c^{z}_{tki} \le \overline{c}^{z}_{tki}$ can compose the loading pattern. Therefore, the use of cuboids with variable size generates a constructive tree, in which each node generates at most $\sum_{t=1}^{m}\sum_{k=1}^{6}\left|\overline{c}_{tki}\right|$ child nodes, which is a substantial increase relative to the maximum of $6 \cdot m$ new partial solutions in each node of the tree suggested by Eley (2002). In the example of Figure 4, the cuboid (2, 2, 3) generates 12 nodes. As a result, the search is conducted in a larger solution space, thus providing a more flexible approach to tackle weakly and strongly heterogeneous cargoes. At each node a bias function is used to prioritize the building of "good" cuboids.

The addition of the length with lateral support to the computation of the cuboid prevents a large fragmentation of the front space of the container and also improves the horizontal stability.

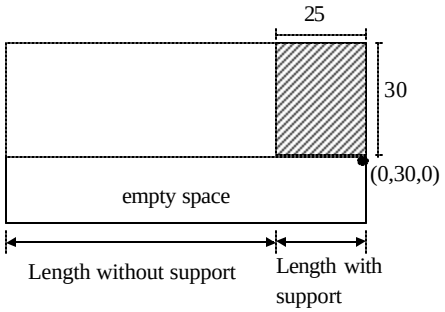


Figure 5. Top view of empty spaces for a container with one box

Figure 5 shows the top view of empty spaces with and without lateral support for a container with one box that has a base of $25\text{ cm} \times 30\text{ cm}$. The empty space with coordinates (0, 30, 0) has the same length as the container, from which the length of 25 cm has lateral support. Let s_i^x denote the length of an empty space *i* with lateral support. Then expression (3) is modified as follows:

$$c_{tki}^{x} = \min\left(\min\left(\left\lfloor \frac{X_i}{d_{1tk}} \right\rfloor, \left\lfloor \frac{s_i^x}{d_{1tk}} \right\rfloor \right), \left\lfloor \frac{\hat{q}_t}{c_{tki}^z c_{tki}^y} \right\rfloor \right) \qquad (4)$$

Note that the second term in expression (4) yields the maximum number of boxes that can be loaded with lateral support.

Another strategy employed to reduce the fragmentation of the front space is to discard empty spaces that are considered *too far* from empty spaces that lie further back in the container (Verweij, 1996). Formally, let x_i and x_0 be the coordinates along the axis x of an empty space i and an empty space 0 further back in the container, respectively, and let $d_{1k_0t_0}$ be the largest length of the unloaded boxes. Then any empty space that satisfies $x_i - x_0 > d_{1k_0t_0}$ is considered *too far*, as illustrated in Figure 6.

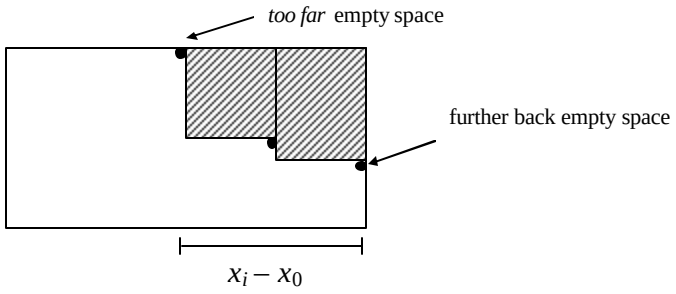


Figure 6. Example of a *too far* empty space

For each combination of box type, orientation and empty space, a candidate cuboid $\overline{c}_{tki}$ is evaluated by sets of the six criteria that are arranged in decreasing order of importance.

C1. Largest normalized space utilization of the empty space:

$$ad\left(X_i - \overline{c}^x_{tki}\, d_{1kt}\right) + ad\left(Y_i - \overline{c}^y_{tki}\, d_{2kt}\right) + ad\left(Z_i - \overline{c}^z_{tki}\, d_{3kt}\right)$$

where

$$ad\left(D - \overline{c}\, d\right) = \begin{cases} 1 & \text{if } D - \overline{c}\, d = 0 \\ 1 \Big/ \left(D - \overline{c}\, d\right) & \text{otherwise,} \end{cases}$$

D represents one of the dimensions X_i, Y_i, Z_i of the empty space, and $\overline{c}\, d$ denotes the respective dimension of the packed boxes $\overline{c}^x_{tki}\, d_{1kt}, \overline{c}^y_{tki}\, d_{2kt}, \overline{c}^z_{tki}\, d_{3kt}$.

C2. Smallest difference between the height of the empty space and the height of the cuboid: $Z_i - d_{3tk}\overline{c}^z_{kit}$.

C3. Largest volume utilization of the empty space: $\dfrac{v_t \left|\overline{c}_{tki}\right|}{ve_i}$.

C4. Largest base area: $d_{1tk}\overline{c}^x_{tki}\, d_{2tk}\overline{c}^y_{tki}$.

C5. Smallest lengthwise protrusion: $d_{itk}\overline{c}^x_{tki} + x_i$.

C6. Empty space with smallest widthwise coordinate: y_i.

Table 1. Sets of evaluation criteria

Set	Criteria
E_1	C1, C2, C5, C6
E_2	C3, C4, C5, C6
E_3	C1, C2, C4, C6
E_4	C3, C2, C4, C6

Table 1 shows four evaluation sets, each one containing four criteria, which are used to assess a candidate cuboid. The first criterion in each set is the most important and the remainders are tie-breakers. For each instance the most suitable evaluation set is determined probabilistically according to the mechanism proposed by Prais & Ribeiro (2000) for selecting the parameter that restricts the candidate list in the construction phase of GRASP. In the first K iterations of the heuristic, we collect information about the container volume utilization associated to the choice of each evaluation set $E_i, i=1,2,3,4$, and the probabilities are made equal to p_i = 1/4. Let V^* denote the best current volume utilization and let $\overline{V}_i$ be the average volume of all solutions found by using the evaluation set $E_i, i=1,2,3,4$. The selection probabilities are periodically revaluated every *K* iterations, by taking $p_i = l_i / \sum_{j=1}^{4} l_j$ with $l_i = (\overline{V}_i / V^*)^f$ for *i* = 1, 2, 3, 4. The parameter f may assume a value greater than one in order to accentuate the difference among l_i, *i* = 1, 2, 3, 4.

The t best cuboids, as evaluated by the above procedure are ranked from the best to the worst in a restricted candidate list (RLC). A cuboid in RLC is then selected with probability that is based on its rank $r(s)$ and a bias function that favors the selection toward some particular candidates, as suggested by Bresina (1996). Let $r(s)$ denote the rank of an element s in RLC and let bias($r(s)$) represent the bias function. The probability $p(s)$ of selecting element s is

$$p(s) = \frac{bias(r(s))}{\sum_{s' \in RLC} bias(r(s'))} \qquad (5)$$

The number of boxes of the selected cuboid is then reduced probabilistically along the three dimensions by means of a bias function that favors values close to zero. Let *qb* be the

number of boxes along one dimension and let $rd \in [0, qb-1]$ be the reduction number. Then the probability of selecting the reduction r along this dimension is given by

$$p(rd) = \frac{bias(rd)}{\sum_{rd'=0}^{qb-1} bias(rd')}. \quad (6)$$

Figure 7 presents the pseudo-code of the multi-start heuristic algorithm for the container loading problem. The variable V^* and the variable *iter*, which counts the number of iterations, are initialized in lines 1 and 2, respectively. The counter is updated in line 4. The procedure of choosing a set of evaluation criteria in line 5 is as presented above. Line 6 invokes the procedure that constructs a solution S. The best current solution (incumbent solution) is updated in line 7, and the incumbent solution is returned in line 9.

	Procedure multi-start algorithm for container loading
1	$V^* \leftarrow 0$
2	*iter* $\leftarrow 0$
3	**Repeat**
4	*iter* $\leftarrow$ *iter* + 1
5	*Criteria* $\leftarrow$ choose a set of evaluation criteria
6	$S \leftarrow$ Random_Construction (*Criteria*)
7	**If** $V^* < V^S$ **then** $S_{incumbent} \neg S$ **End_if**
8	**Until** *stopping condition is met*
9	**Return**($S_{incumbent}$)

Figure 7. Multi-start heuristic algorithm

Figure 8 shows the pseudo code of the random constructive heuristic. The structure that stores a solution S and the list of empty spaces are initialized in lines 1 and 2. The construction of a solution proceeds while there exist unloaded boxes and empty spaces, as indicated in line 3. Lines 4 to 16 correspond to the evaluation of the combination of empty spaces and box types with feasible orientations in order to build the RCL. Cuboids are calculated in lines 9, and in line 11, the elements of RCL are ordered according to the evaluation criteria. In line 17 an element of RCL is selected, in line 18 a reduction is applied to this element, which is then added to the partial solution in line 19. The lists of empty spaces unloaded boxes are updated in line 20, while the complete solution in returned in line 22.

Procedure Random_Construction (*Criteria*)

1 $S \leftarrow \{\}$

2 $list\_empty\_spaces \leftarrow$ empty spaces of S

3 **While** $\sum_{t=1}^{m} \hat{q}_t > 0$ **AND** $|list\_empty\_spaces| > 0$ **do**

4 **For** $i \neg 1$ **to** $|list\_empty\_spaces|$ **do**

5 **For** $t \leftarrow 1$ **to** m **do**

6 **If** $\hat{q}_t > 0$ **AND** a box of type t fits in the empty space i **then**

7 **For** $k \leftarrow 1$ **to** 6 **do**

8 **If** the orientation k of Box of type t is feasible **then**

9 Compute the cuboid $\overline{c}_{tk}$ according to expressions (1), (2) e (4)

10 **End_if**

11 Update RLC by using *Criteria*

12 **End_for**

14 **End_if**

15 **End_for**

16 **End_for**

17 Select a cuboid from RLC according to the bias function (5)

18 Apply reductions to the selected cuboid by using the bias function (6)

19 Include the selected cuboid in the solution S

20 Update $list\_empty\_spaces$ and the number of unloaded boxes

21 **End_while**

22 **Return**(S)

Figure 8. Algorithm for the random constructive heuristic

4. Computational experiments

The multi-start algorithm was coded in C++ by using the version 3.3.3 of the GCC compiler with optimizer option O3 and computational tests for sets of instances of the literature were carried out, unless otherwise stated, on a PC Intel Pentium IV 2.8 GHz with the operating system Linux Fedora Core 2.

If the identification of empty spaces in the container is performed from backward to forward, the heuristic version is denoted AAR1 and in case it is carried out from backward to

forward and forward to backward simultaneously, it is denoted AAR2. Unless otherwise stated, the maximum number of constructed solutions is 240.

From the bias functions suggested by Bresina (1996), we selected the same function in expressions (5) and (6) with the form $bias(x) = x^{-n}$ since the best candidate elements in RLC have close evaluation values and best reduction values should be close to zero. Let V^p denote the value of a partial solution. The exponent *n* used in expressions (5) and (6) is then defined as

$$n = \begin{cases} n_1 & \text{if } V^p < r \cdot V^* \\ n_2 & \text{otherwise} \end{cases}$$

where, r = 0,8, $n_1 = 2$ e $n_2 = 3$ for instances with less than eight types of boxes and $n_1 = 3$ e $n_2 = 4$, otherwise. The remaining parameters were set to τ = 10, K = 100 and f = 10.

The reported results for versions AAR1 e AAR2 correspond to the average throughout ten runs, with different seeds for the pseudo-random number generator. The performance of the multi-start heuristic is compared with that of seventeen heuristics:

- MA – heuristic AND/OR-graph (Morabito & Arenales,1994)
- BJR – constructive heuristic (Bischoff *et al.*,1995)
- BR – constructive heuristic (Bischoff & Ratcliff, 1995)
- BG_1 – genetic algorithm (Gehring & Bortfeldt, 1997)
- BG_2 – tabu search (Bortfeldt & Gehring, 1998)
- DB – constructive heuristic (Davies & Bichoff,1999)
- BG_3 – hybrid genetic algorithm (Bortfeldt e Gehring, 2001)
- GB – parallel genetic algorithm (Gehring e Bortfeldt, 2002)
- E – constructive heuristic with tree search (Eley, 2002)
- BGM_1 – sequential tabu search (Bortfeldt et al., 2003)
- BGM_2 – parallel tabu search (Bortfeldt et al., 2003)
- CM – five constructive heuristics (Cecílio & Morabito, 2004)
- CD – constructive heuristic (Chien & Deng, 2004)
- MBG – hybrid tabu search/simulated annealing (Mack *et al.*, 2004)
- LZ – squeaky wheel optimization (Lim & Zhang, 2005)
- MO – GRASP (Moura & Oliveira, 2005)
- B – multi-start heuristic (Bischoff, 2006)

The code of Cecílio & Morabito (2004) was used in Tables 2, 3, 4 and 5 to obtain results for the heuristic CM. Table 2 shows the results for instances generated according to the scheme suggested by Cecílio & Morabito (2004), who also proposed five heuristics that correspond to refinements of the approach by George & Robinson (1980). The best result obtained by the application of such heuristics to each instance is reported. The numbers in parenthesis in the first column represent the number of box types. In terms of volume utilization, the version AAR2 outperforms AAR1, which in turn outperforms CM. However, AAR2 spends more computational time, followed by AAR1 and CM. The small computational time required by CM can be explained by the identification of a smaller number of empty spaces due to the use of vertical walls, and a greedy heuristic that selects one of the box types. Recall that algorithms AAR1 and AAR2 do not impose constraints on the identification of empty spaces and considers all possible combinations of empty spaces and box types for the selection of a cuboid that builds the load arrangement. If the number of solutions evaluated by AAR1 and AAR2 is reduced to 60, then the mean computational time is lowered to 0.13 seconds and 0.31 seconds, respectively. With this reduction, the mean volume utilization becomes 89.56% for AAR1 and 90.20% for AAR2, still maintaining the dominance ordering for all instances as mentioned above

Table 2. Results for instances of Cecílio & Morabito (2004)

Instances	CM		AAR1		AAR2	
	Vol. (%)	Time (s)	Vol. (%)	Time (s)	Vol. (%)	Time (s)
C1 (10)	89.25	0.16	93.81	0.21	**94.30**	0.45
C2 (10)	81.05	0.07	87.76	0.13	**88.07**	0.25
C3 (10)	90.78	0.18	94.31	0.14	**95.06**	0.36
C4 (10)	83.93	0.08	89.51	0.10	**89.90**	0.19
C5 (50)	86.56	0.43	91.00	2.02	**91.25**	4.66
C6 (50)	72.87	0.18	85.32	0.91	**85.49**	1.63
C7 (50)	87.19	0.36	92.39	1.04	**92.98**	2.69
C8 (50)	80.35	0.20	91.17	0.56	**91.54**	1.11
Mean	84.00	0.21	90.66	0.64	**91.07**	1.42

Table 3 presents the volume utilization percentage for twelve real instances relative to seven companies as described by Cecílio & Morabito (2004). The instances are grouped by company, for example, instances B1 and B2 refer to company B. For eight instances, all heuristics find optimal solutions, i.e., solutions in which all boxes are loaded. The heuristics

AAR1 and AAR2 find an optimal solution for instance G-3, while the heuristic CM finds an optimal solution for instance G-1. The heuristic AAR1 obtains the best solutions for instances B-1 and G-2, for which optimal solutions are not available. Computational times to solve all instances are similar, being of order of hundredths of seconds for CM and tenths of seconds for AAR1 and AAR2.

Table 3. Results for real instances of Cecílio **&** Morabito (2004)

Instances	CM Vol. (%)	AAR1 Vol. (%)	AAR2 Vol. (%)
A (47)	*** 86.22**	*** 86.22**	*** 86.22**
B-1 (26)	85.56	**89.25**	88.67
B-2 (22)	*** 83.67**	*** 83.67**	*** 83.67**
C (04)	*** 79.99**	*** 79.99**	*** 79.99**
D (10)	*** 75.02**	*** 75.02**	*** 75.02**
E (05)	*** 80.73**	*** 80.73**	*** 80.73**
F-1 (05)	*** 94.23**	*** 94.23**	*** 94.23**
F-2 (05)	*** 92.59**	*** 92.59**	*** 92.59**
F-3 (05)	*** 93.54**	*** 93.54**	*** 93.54**
G-1 (05) #	*** 99.29**	96.51	97.90
G-2 (04)	98.38	**98.98**	98.96
G-3 (09)	89.02	*** 89.80**	*** 89.80**

* Optimal solution

We have not been able to reproduce this result with the code of Cecílio & Morabito (2004)

A PC Athlon 800 MHz, 512 RAM was used for the execution of computational tests of Tables 4 and 5. Table 4 shows the results obtained by the heuristics MA, CM, AAR1 and AAR2 for the 80 instances generated by Morabito & Arenales (1994). Such instances refer to the unconstrained container involving 5, 10, 20 and 30 types (in parentheses in the first column) of boxes with dimensions ranging from 5% to 85% of the container dimensions. The heuristic MA suggested by Morabito & Arenales (1994), which was adapted to allow rotations (Cecílio & Morabito, 2004), outperforms the remaining heuristics, though at a larger computational effort.

Since several boxes of the 80 instances have dimensions comparable to the container dimensions, it is reasonable to adjust the parameters of the bias functions (5) and (6) in order to

obtain a higher degree of randomness, so that $n_1 = 1$ and $n_2 = 4$. The modified heuristic is denoted AAR1*. The stopping criterion now is computational time, which is similar to that reported by Morabito & Arenales (1994). Table 5 shows that the heuristics MA and AAR1 are very competitive and that the version AAR1* yields best solutions in six sets and a better mean volume utilization.

Table 4. Results for instances of Morabito & Arenales (1994)

Instances	MA		CM		AAR1		AAR2	
	Vol. (%)	Time (s) ¥	Vol. (%)	Time (s)	Vol. (%)	Time (s)	Vol. (%)	Time (s)
s1 (05)	**88.02**	10.63	81.50	0.01	86.29	0.16	85.90	0.24
s2 (05)	**93.40**	12.81	83.86	0.01	90.78	0.13	90.70	0.18
s3 (05)	**83.54**	3.87	82.52	0.01	83.20	0.16	83.09	0.19
s4 (10)	**98.71**	40.78	94.73	0.03	97.29	0.16	97.29	0.20
s5 (05)	**95.21**	24.49	88.92	0.04	94.20	0.09	94.39	0.13
s6 (10)	**97.17**	66.34	94.51	0.01	96.38	0.16	96.46	0.19
s7 (20)	**98.12**	127.05	95.61	0.05	97.14	0.29	97.28	0.31
s8 (30)	**98.44**	147.33	96.70	0.02	98.16	0.36	98.16	0.43
Mean	94.08	54.16	89.79	0.02	92.93	0.19	92.91	0.23

¥ Pentium III, 800 MHz, 256 MB RAM

Table 5. New results for instances of Morabito & Arenales (1994)

Instances	MA		AAR1		AAR1*	
	Vol. (%)	Time (s)	Vol. (%)	Time (s)	Vol. (%)	Time (s)
s1 (05)	88.02	10.63	88.24	10.00	**88.44**	10.00
s2 (05)	93.40	12.81	93.32	12.00	**93.59**	12.00
s3 (05)	83.54	3.87	86.05	4.00	**86.86**	4.00
s4 (10)	**98.71**	40.78	98.34	40.00	98.50	40.00
s5 (05)	95.21	24.49	**95.50**	24.00	95.48	24.00
s6 (10)	97.17	66.34	97.36	66.00	**97.41**	66.00
s7 (20)	98.12	127.05	98.34	125.00	**98.41**	125.00
s8 (30)	98.44	147.33	98.86	145.00	**98.89**	145.00
Mean	94.08	54.16	94.50	53.25	**94.70**	53.25

Table 6 shows comparative results with the heuristic proposed by Chien & Deng (2004) for eleven real instances of shipping companies in Taiwan. The heuristic version AAR2 obtains best results for all instances with a computational time inferior to 0.3 seconds, and in three of the ten runs it yields a solution for instance 10 that fills 100% of the container volume. The total container filling is also obtained in one of the ten runs of the heuristic version AAR1 for instances 9 and 10.

Table 6. Results for instances of Chien & Deng (2004)

Heuristic	Instances										
	1	2	3	4	5	6	7	8	9	10	11
CD	83.08	95.22	83.43	87.68	80.49	81.16	91.30	90.55	95.55	93.08	92.02
AAR1	83.94	97.26	93.29	98.66	**83.04**	**85.25**	99.56	**92.40**	98.09	99.57	97.57
AAR2	**83.94**	**97.32**	**94.73**	**98.64**	**83.04**	**85.25**	**99.74**	**92.40**	**99.16**	**99.82**	**97.81**

Table 7 lists heuristics and metaheuristics from the literature that utilize 700 instances generated by Bischoff e Ratcliff (1995) in order to validate their performance. These instances are divided into seven test cases BR1-BR7, each case with number of distinct boxes types (in parentheses). Orientation and stability constraints are imposed and the the box stability limit is set to a value 2. This implies that for a box of type t with orientation k, $u_{tk} = 0$ if $d_{3tk} / \left(\min_{j=1,2} d_{jtk} \right) \geq 2$, i.e., orientation k is not allowed if its height is greater than the double of one of the base dimensions. The results in this table are presented in increasing order of the mean volume occupation over the seven instances. The heuristic version AAR2 obtains best results for all test cases followed by the version AAR1. The mean computational times spent by AAR2 and AAR1 are 0.29 seconds and 0.14 seconds, respectively. The distinct computers used in the execution of the heuristics of Table 7 make it difficult to compare the computational time required by them in order to solve the instances. Nevertheless, it is worth stressing that the mean time spent by the metaheuristic MO on a 2.4 GHz Pentium IV is 33.5 seconds.

Table 7. Results for instances of Bischoff & Ratcliff (1995)

Heuristic	Instances							Mean
	BR1(03)	BR2(05)	BR3(08)	BR4(10)	BR5(12)	BR6(15)	BR7(20)	
BJR	81.76	81.70	82.98	82.60	8276	81.50	80.51	81.97
BR	83.79	84.44	83.94	83.71	83.80	82.44	82.01	83.45
DB	84.10	84.50	85.00	84.70	84.60	83.70	82.70	84.19
BG_1	85.80	87.26	88.10	88.04	87.86	87.85	87.68	87.51
CM	89.05	87.40	87.21	86.75	87.09	86.05	84.82	88.34
E	88.05	88.44	89.23	89.24	88.99	88.91	88.36	88.75
MO	89.07	90.43	90.86	90.42	89.57	89.71	88.05	89.07
LZ	87.4	88.7	89.3	89.7	89.7	89.7	89.4	89.13
AAR1	90.86	90.88	90.94	90.67	90.40	90.14	89.46	90.48
AAR2	**91.73**	**91.60**	**91.47**	**91.06**	**90.90**	**90.46**	**89.54**	**90.96**

Bischoff & Ratcliff (1995) propose a heuristic that is designed to produce patterns which combine high space utilization with a high degree of stability, and suggest two measures related the stability. Measure 1 is the average number of boxes that support boxes that do not lie on the floor (the higher the better), while measure 2 is the average percentage of boxes not surrounded on at least three sides (the lower the better). Table 8 presents comparative results with six heuristics and metaheuristics that reported results for such measures. With regards to measure 1, BJR outperforms the other methods, while AAR1 is superior relative to measure 2, due to the cuboid approach. As expected, the performance of AAR2 is worse than AAR1, since in the first version the loading is carried out from backward to forward and forward to backward simultaneously, which results in empty spaces at the junction of the backward and forward fronts.

Table 8. Load stability for instances of Bischoff & Ratcliff (1995)

Heuristic	Instances													
	BR1		BR2		BR3		BR4		BR5		BR6		BR7	
	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
BJR	2.02	8.50	2.22	11.21	2.20	15.93	2.10	17.51	2.09	21.60	2.04	22.13	1.92	27.07
BR	1.13	10.36	1.10	14.60	1.08	19.67	1.07	23.53	1.06	26.03	1.06	31.04	1.04	35.99
BG_1	-	11.00	-	16.00	-	18.50	-	21.50	-	22.50	-	25.00	-	28.50
CM	1.14	7.57	1.12	10.75	1.10	13.72	1.10	14.99	1.10	16.50	1.10	19.58	1.10	21.76
E	-	9.80	-	13.50	-	18.00	-	20.50	-	21.50	-	22.90	-	26.00
MO	1.07	11.53	1.10	12.67	1.09	17.75	1.10	20.03	1.10	22.75	1.10	26.50	1.11	28.86
AAR1	1.15	6.00	1.15	9.22	1.12	10.35	1.11	12.48	1.11	13.70	1.10	15.70	1.08	18.24
AAR2	1.18	10.77	1.18	13.72	1.15	16.17	1.13	18.09	1.13	19.51	1.12	20.91	1.10	23.91

Table 9 shows results in increasing order of the mean volume occupation obtained by additional metaheuristics from the literature for the 700 instances BR1-BR7. The authors of the metaheuristics BGM_1, BGM_2 and MBG report only the mean volume occupation for fully supported boxes. For this experiment we stipulated computational time as the stopping criterion for AAR1 and AAR2. The heuristic version AAR2 obtains best results for all test cases followed by the version AAR1, with the exception of test case BR7. Note that the proposed approach is more effective for weakly heterogeneous box sets. The time to solve all instances of each group is shown in the last line of the table. The mean computational time to solve the 700 instances is 52 seconds, and based on data available at http://www.spec.org/cpu2000/results/cfp2000.html we concluded that this time corresponds to approximately 75 seconds in an Intel Pentium 2GHz. This computer was used by Mack *et al.* (2004) who report a mean computational time of 205 seconds to solve the 700 instances BR1-BR7.

Table 9. Additional results for instances of Bischoff & Ratcliff (1995)

Heuristic	Instances							Mean
	BR1(03)	BR2(05)	BR3(08)	BR4(10)	BR5(12)	BR6(15)	BR7(20)	
BG_3	87.81	89.40	90.48	90.63	90.73	90.72	90.65	90.06
GB	88.10	89.56	90.77	91.03	91.23	91.28	91.04	90.43
BG_2	92.41	92.33	91.97	91.26	90.40	89.57	88.18	90.87
B	90.57	90.84	91.43	91.21	91.25	91.04	90.81	91.02
BGM_1	-	-	-	-	-	-	-	91.60
BGM_2	-	-	-	-	-	-	-	92.20
MBG	-	-	-	-	-	-	-	92.41
AAR1	92.58	92.93	92.96	92.70	92.48	92.11	**91.53**	92.53
AAR2	**93.38**	**93.25**	**93.11**	**92.77**	**92.51**	**92.16**	91.34	**92.64**
Time (s)	6.00	12.00	36.00	60.00	70.00	80.00	100.00	52.00

We have also tested the performance of the proposed heuristic on a real life example reported by George & Robinson (1980), which consists of 784 boxes distributed in eight types to be loaded in a container with available dimensions in millimiters $5793 \times 2236 \times 2261$. For the stopping criterion of 240 evaluated solutions, AAR1 and AAR2 produce solutions in which all boxes are packed. We have extended this example by increasing one additional box of each type. In this case, AAR1 and AAR2 pack all boxes in 5 and 10 executions, respectively. When we consider two additional boxes of each type, AAR1 and AAR2 pack all boxes in 3 executions, respectively. Finally, AAR2 packs 24 additional boxes (three additional boxes of each type) in one execution with a container volume utilization of 93.03%.

5. Conclusions

In this paper we proposed a multi-start random constructive heuristic for loading boxes in a single container, with the objective of maximizing its volume utilization subject to orientation and stability constraints. Several conclusions can be drawn from the design and experiments for the proposed heuristic. Initially, the cuboid arrangement is an effective approach, even when dealing with a rather strongly heterogeneous set of boxes. In addition, the use of the adaptation of the spatial representation is very important to identify empty spaces. Finally, we have learned that a constructive heuristic with a controlled degree of randomization coupled with a suitable bias function is competitive and simpler when compared with metaheuristics proposed in the literature for this problem. The proposed heuristic is fairly

robust, has few parameters and it is able to produce high quality solutions in short computational time. Further research involves the use of the proposed approach to deal with other practical constraints, such weight distribution and limited load bearing strength.

Acknowledgements

This research was partially funded by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). The authors are also grateful to Fabiana Oliveira Cecilio and Reinaldo Morabito for providing us with the code of their algorithms.

References

Bischoff, E.E. & Marriott, M.D. (1990). A Comparative Evaluation of Heuristics for Container Loading. *European Journal of Operational Research,* **44**, 267-276.

Bischoff, E.E. & Ratcliff, M.S.W. (1995). Issues in the development of approaches to container loading", *Omega*, **4**, 377-390.

Bischoff, E.E. (2006). Three-dimensional packing of items with limited load bearing strength, *European Journal of Operational Research*, **168** (3), 952-966.

Bischoff, E.E.; Janetz, F. & Ratcliff, M.S.W. (1995). Loading Pallets with Non-Identical Items**.** *European Journal of Operational Research***, 84**, 681-692.

Bortfeldt, A. & Gehring, H. (1998). Ein Tabu Search-Verfahren für Containerbeladeprobleme mit schwach heterogenem Kistenvorrat. *OR Spektrum*, **20**(4), 237-250.

Bortfeldt, A. & Gehring, H. (2001). A Hybrid Genetic Algorithm for the Container Loading Problem. *European Journal of Operational Research*, **131**, 143-161.

Bortfeldt, A.; Gehring, H. & Mack D. (2003). A parallel tabu search algorithm for solving the container loading problem, *Parallel Computing*, **29**(5), 641-662.

Bresina, J. (1996). Heuristic-Biased Stochastic Sampling, In Proceedings of the 13th national Conference on Artificial Intelligence, 271-278.

Cecilio, F.O. & Morabito, R. (2004). Refinamentos na heurística de George e Robinson para o problema de carregamento de caixas dentro de contêineres, *Transportes*, **12**(1), 32-45.

Chen, C.S.; Lee, S.M. & Shen, Q.S. (1995). An Analytical Model for the Container Loading Problem, *European Journal of Operational Research*, **80**, 68-76.

Chien C.F. & Deng J.F. (2004). A container packing support system for determining and visualizing container packing patterns, *Decision Support Systems*, **37**(1), 23-34.

Davies, A.P. & Bischoff, E.E. (1999). Weight Distribution Considerations in Container Loading. *European Journal of Operational Research*, **114**, 509-527.

Dyckhoff, H. (1990). A typology of cutting and packing problems, *European Journal of Operational Research*, **44**, 145-159.

Eley, M. (2002). Solving container loading problems by block arrangement, *European Journal of Operational Research*, **141**(2), 393-409

Faroe O.; Pisinger D. & Zachariasen M. (2003). Guided Local Search for the Three-Dimensional Bin-Packing Problem, *INFORMS Journal on Computing*, **15**(3), 267-283.

Feo, T.A. & Resende, M.G.C. (1995). Greedy randomized adaptive search procedures, *Journal of Global Optimization*, **6**, 109-133.

Festa, P. & Resende, M.G.C. (2004). " An annotated bibliography of GRASP ", *European Journal of Operational Research*, submitted.

Gehring, H. & Bortfeldt, A. (1997). A Genetic Algorithm for Solving the Container Loading Problem. *International Transactions in Operations Research*, **4**(5-6), 401-418.

Gehring, H.; Menschner, K. & Meyer, M. (1990). A Computer-Based Heuristic for Packing Pooled Shipment Containers. *European Journal of Operational Research*, **44**, 277-288.

George, J.A., Robinson, D.F., 1980. A Heuristic for Packing Boxes into a Container, *Computers and Operations Research*, **7**, 147-156.

Haessler, R.W. & Talbot, F.B. (1990). Load Planning for Shipments of Low Density Products. *European Journal of Operational Research*, **44**, 289-299.

Hassamontr, J. (2003). On decomposing 3D Packing Problem in Wooden Furniture Industry, In: *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 497-502.

He, D.Y. & Cha, J.Z. (2002). Research on Solution to Complex Container Loading Problem Based on Genetic Algorithm, In: *Proceeding of First International Conference on Machine Learning and Cybernetics*, **1**, 78-82, Beijing, November.

Hifi, M. (2004). Exact algorithms for unconstrained three-dimensional cutting problems: a comparative study, *Computers & Operations Research*, **31**(5), 657-674.

Lai, K.K.; Xue, J. & Xu B. (1998).Container packing in a multi-customer delivering operation, *Computers & Industrial Engineering*, **35**(1-2), 323-326.

Lim, A. & Zhang, X. (2005). The container loading problem, *ACM Symposium on Applied Computing*, 913-917.

Lim, A.; Rodrigues, B. & Wang Y. (2003). A multi-faced buildup algorithm for three-dimensional packing problems, *Omega*, **31**(6), 471-481.

Lin, S. & Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Travelling-Salesman Problem, *Operations Research*, **21**, 0498-0516.

Mack, D.; Bortfeldt, A. & Gehring, H. (2004). A parallel hybrid local search algorithm for the container loading problem. *International Transaction in Operational Research*, **11**, 511-534.

Martello, S.; Pisinger, D. & Vigo, D. (2000). The Three Dimensional Bin Packing Problem, *Operations Research*, **48**, 256-267.

Martí, R. (2003). Multi-Start Methods, In: Handbook of MetaHeuristics [edited by F. Glover and G. Kochenberger], Kluwer, 355-368.

Morabito, R. & Arenales, M. (1994). An And/Or-graph Approach to the Container Loading Problem. *International Transactions in Operations Research*, **1**(1), 59-73.

Moura, A. & Oliveira, J.F. (2005). A GRASP Approach to the Container-Loading Problem, *IEEE Intelligent Systems*, **4**(20), 50-57.

Ngoi, B.K.A.; Tay, M.L. & Chua, E.S. (1994). Applying spatial representation techniques to the container packing problem, *International Journal of Production Research*, **32**, 111-123.

Pisinger, D. (2002). Heuristics for the container loading problem, *European Journal of Operational Research*, **141**, 143-153.

Prais, M. & Ribeiro, C.C. (2000). Reactive GRASP: "An application to a matrix decomposition problem in TDMA traffic assignment", *INFORMS Journal on Computing*, **12**, 164-176.

Resende, M.G.C. & Ribeiro, C.C. (2003). Greedy randomized adaptive search procedures", In: Handbook of MetaHeuristics [edited by F. Glover and G. Kochenberger], Kluwer, 219-249.

Verweij, A.M. (1996). Multiple destination bin packing. UU-CS (Ext. r. no. 1996-39). Utrecht, the Netherlands: Utrecht University: Information and Computing Sciences.