UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2569

Estrutura de Dados Utilizada na Abordagem em Grafo E/OU para a Resolução de Problemas de Corte

Andréa Carla Gonçalves Vianna
Marcos Nereu Arenales
Reinaldo Morabito

Nº 126

RELATÓRIOS TÉCNICOS

São Carlos – SP
Nov./2000

Estrutura de dados utilizada na abordagem em Grafo E/OU para a Resolução de Problemas de Corte

Andréa Carla Gonçalves Vianna
UNESP - DCo - Bauru
vianna@fc.unesp.br

Marcos Nereu Arenales
USP - ICMC - São Carlos
arenales@icmc.sc.usp.br

Reinaldo Morabito
UFSCar - DEP - São Carlos
morabito@power.ufscar.br

Agosto de 2000

SUMÁRIO

Página

Lista de Figuras iii
Lista de Tabelas v
Lista de Gráficos vi
Resumo vii
Abstract viii
1. Introdução 1
2. Problema de Corte 3
2.1. Classificação dos Problemas 3
2.2. O Problema de Corte Bidimensional 7
3. Resolução do Problema através de Grafo E/OU 10
3.1. Representação em Grafo E/OU 10
3.2. Busca no Grafo E/OU 14
3.2.1. Geração do conjunto de discretização 14
3.2.2. Limitante Inferior 19
3.2.3. Limitante Superior 20
3.2.4. Estratégia de Busca 21
3.3. Heurísticas 23
4. Implementação Computacional 26
5. Resultados Computacionais 33
5.1. Conjunto de discretização 33
5.2. Problema Irrestrito 41
6. Conclusões 45
7. Referências Bibliográficas 47

LISTA DE FIGURAS

Página

2.1. Problema de corte unidimensional 5
2.2. Padrão de corte unidimensional 5
2.3. Problema de corte bidimensional 6
2.4. Padrão de corte bidimensional 6
2.5. Problema de corte tridimensional 7
2.6. Padrão de corte tridimensional 7
2.7. Corte guilhotinado horizontal e vertical 8
2.8. Corte não-guilhotinado 8
2.9. Padrão de corte guilhotinado e não-guilhotinado 9
2.10. Padrão de corte guilhotinado 2-estágios, 3-estágios e 4-estágios 9
3.1. Representação de grafo 11
3.2. Representação de hipergrafo 11
3.3. Representação de grafo E/OU 12
3.4. Padrão de corte 12
3.5. Grafo E/OU representando o padrão da Figura 3.4 13
3.6. Grafo E/OU com diversos padrões de corte 14
3.7. Padrão Normal ($x=\ell_1+2.\ell_2$) 15
3.8. Simetria 16
3.9. Exclusão ($x=\ell_1+\ell_2$ pode ser excluído do conjunto X) 17
3.10. Ordenação 19
3.11. Padrão de corte homogêneo 20
3.12. Padrão homogêneo com apenas cortes verticais 20
4.1. Lista seqüencial 28
4.2. Lista encadeada 28
4.3. Árvore binária ordenada 29
4.4. Dados armazenados em cada nó do grafo em problemas guilhotinados 29
4.5. Estrutura do nó armazenado 30
4.6. Dados armazenados em cada nó do grafo em problemas não-guilhotinados 30
4.7. Estrutura do nó gerado pelo corte não-guilhotinado .. 31

4.8. Dados armazenados em cada nó do grafo em problemas guilhotinados (Morabito, 1992) 31
4.9. Estrutura gerada por Morabito (1992) 31

LISTA DE TABELAS

Página

5.1. Discretização nos exemplos pequenos 34
5.2. Discretização nos exemplos grandes 35
5.3. Discretização nos exemplos pequenos (com simetria) . 37
5.4. Discretização nos exemplos grandes (com simetria) .. 38
5.5. Discretização nos exemplos pequenos com $|X| < 100$ e $|Y| < 100$ (Heurística H5) 39
5.6. Discretização nos exemplos grandes com $|X| < 100$ e $|Y| < 100$ (Heurística H5) 40
5.7. Exemplos pequenos 42
5.8. Exemplos grandes 42
5.9. Exemplos pequenos 43
5.10. Exemplos grandes 43
5.11. Exemplos pequenos 44
5.12. Exemplos grandes 44

LISTA DE GRÁFICOS

Página

5.1. Discretização nos exemplos pequenos 37

5.2. Discretização nos exemplos grandes 37

5.3. Discretização nos exemplos pequenos com $|X| < 100$ e $|Y| < 100$ (Heurística H3) 41

5.3. Discretização nos exemplos grandes com $|X| < 100$ e $|Y| < 100$ (Heurística H3) 41

RESUMO

Este trabalho tem por objetivo mostrar a estrutura de dados e algumas técnicas utilizadas para a resolução do problema de corte bidimensional através da abordagem em Grafo E/OU.

Inicialmente é feita uma apresentação do problema de corte e da estratégia de busca híbrida, onde se combina a busca em profundidade primeiro com limite e a busca *hill-climbing,* utilizando heurísticas baseadas nos limitantes superiores e inferiores. Esta abordagem foi proposta inicialmente por Morabito (1989).

Com isto pretende-se dar um suporte computacional para quem pretende implementar o problema de corte através da abordagem em Grafo E/OU, já que a maioria dos pesquisadores que o fazem, não estão ligados diretamente a área da computação e, não se tem disponível na literatura tais comentários.

Palavras-chave: Problemas de corte e empacotamento, grafo E/OU, otimização, heurísticas.

ABSTRACT

Keywords: Cutting and packing problems, AND/OR-graph search, otimization, heuristics.

1. Introdução

O Problema de Corte, genericamente, consiste em cortar unidades maiores (objetos) em unidades menores (itens), otimizando uma determinada função (por exemplo, minimização da perda). Este tipo de problema aparece em diversos processos industriais de corte, tais como no corte de bobinas de papel e alumínio, barras de aço, chapas metálicas e de madeira, placas de circuito impresso, caixas de papelão, rolos de tecidos, entre outros.

Análogo ao Problema de Corte, o Problema de Empacotamento consiste em empacotar unidades menores em unidades maiores, otimizando determinada função (por exemplo, minimização do espaço ocioso) e satisfazendo um conjunto de restrições. Exemplos do Problema de Empacotamento ocorrem quando se carregam produtos embalados sobre paletes ou dentro de contêineres.

Estas duas classes de problemas de otimização estão intimamente relacionadas e são tratados na literatura como Problemas de Corte e Empacotamento.

O número de aplicações para os Problemas de Corte e Empacotamento é muito grande. Na literatura existem diversas abordagens para resolvê-los, entre elas Gilmore e Gomory (1961, 1963, 1965), Christofides e Whitlock (1977), Wang (1983), Oliveira e Ferreira (1990), Morabito e Arenales (1994), Arenales e Morabito (1995), Morabito e Arenales (1996)

e Hifi (1997).

Uma abordagem de solução baseada numa busca em grafo E/OU para problemas que envolvam objetos de mesma forma, barras (caso unidimensional), retângulos (caso bidimensional) e caixas (caso tridimensional) foi apresentada por Morabito (1992). Nesta abordagem foi utilizada uma técnica de busca híbrida (semi-informada), onde se combinaram a busca em profundidade com limite e a busca *Hill-Climbing*. Utilizaram-se, também, heurísticas baseadas nos limitantes superiores e inferiores.

Em 1993, Arenales apresentou uma extensão genérica da abordagem, sem, entretanto, realizar estudos de casos.

Este trabalho tem por objetivo o estudo de diferentes estruturas de dados e técnicas computacionais para a resolução do Problema de Corte Bidimensional, principalmente parar a geração do conjunto de discretização.

O trabalho está dividido em 7 seções. Na segunda seção é apresentado o problema de corte bidimensional; na seção seguinte é feita a representação usando a abordagem em Grafo E/OU, assim como a geração do conjunto de discretização e o uso de limitantes na estratégia de busca no Grafo E/OU e também, a utilização de heurísticas; na quarta seção são apresentadas duas técnicas de implementação computacional para a geração do conjunto de discretização, assim como a estrutura de dados utilizada na sua representação e na representação do grafo; os resultados computacionais obtidos são apresentados na seção seguinte, seguido da conclusão e das referências bibliográficas.

2. Problemas de Corte

Nesta seção é apresentada uma classificação dos problemas de corte, segundo Dyckhoff (1990), e um estudo mais detalhado do problema de corte bidimensional.

2.1. Classificação dos Problemas

O Problema de Corte consiste, basicamente, em cortar unidades maiores em unidades menores, otimizando um determinado objetivo. Entretanto, os problemas podem aparecer numa diversidade muito grande de casos.

Em 1990, Dyckhoff propôs um sistema de classificação a partir das quatro principais características dos problemas:

- a dimensão do problema
  - (1) unidimensional
  - (2) bidimensional
  - (3) tridimensional
  - (n) n-dimensional, para n>3
- a forma de alocação das unidades
  - (V) seleção de unidades grandes
    (um conjunto de unidades grandes deve ser escolhido para incluir todas as unidades pequenas)
  - (B) seleção de unidades pequenas
    (um conjunto de unidades pequenas deve ser escolhido

para ocupar todas as unidades grandes)

- sortimento de unidades grandes
  - (0) uma unidade
    (uma única unidade grande)
  - (I) unidades de tamanhos iguais
    (todas as unidades grandes são iguais)
  - (D) unidades de tamanhos diferentes
    (as unidades grandes têm tamanhos diferentes)
- sortimento de unidades pequenas
  - (F) poucas unidades de tamanhos diferentes
    (poucas unidades pequenas, geralmente de tamanhos diferentes)
  - (M) muitas unidades de muitos tamanhos diferentes
    (muitas unidades pequenas e a maioria delas em tamanhos diferentes)
  - (R) muitas unidades de poucos tamanhos diferentes
    (muitas unidades pequenas com poucos tamanhos diferentes)
  - (C) unidades de tamanhos iguais
    (todas as unidades pequenas são iguais)

Estas quatro características permitem obter 96 tipos diferentes para problemas de corte e empacotamento e a tipologia é definida pela quadra $\alpha / \beta / \gamma / \sigma$. Cada um dos símbolos corresponde a uma das características citadas.

Um problema de corte de estoque definido pela quadra 2/V/I/R significa que o problema é bidimensional (2), todas as unidades pequenas devem ser alocadas numa seleção de unidades grandes (V), todas de tamanhos iguais (I) e com muitas unidades pequenas variando pouco o tamanho (R).

Considerando somente estas quatro características, os problemas apresentam ainda muitas diferenças quanto às suas restrições, por exemplo, problemas restritos e irrestritos, guilhotinado e não-guilhotinado etc.

A seguir, são definidos problemas de corte,

classificando-os pelas suas dimensões.

O problema de corte unidimensional apresenta apenas uma das dimensões relevante para o processo de cortagem. Considere um objeto, por exemplo, uma barra como na Figura 2.1, que deve ser cortado ao longo de seu comprimento em itens de comprimento especificado. Cada item tem um valor associado, chamado de valor de utilidade. Então, o problema consiste em maximizar o valor de utilidade total.

O modo como os itens estão arranjados ao longo do objeto é chamado de padrão de corte. Para o problema da Figura 2.1, um possível padrão de corte é ilustrado na Figura 2.2.

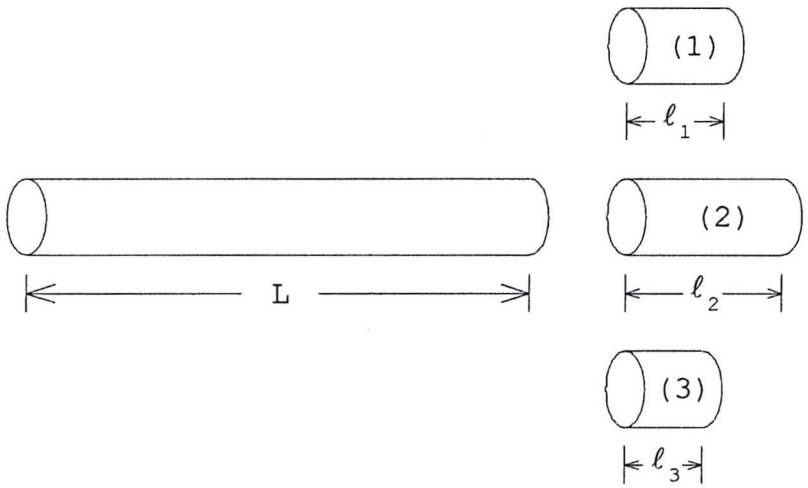

Figura 2.1 - Problema de corte unidimensional

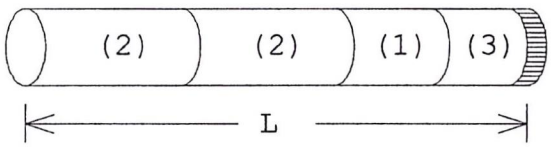

Figura 2.2 - Padrão de corte unidimensional

No problema de corte bidimensional são relevantes duas dimensões, sendo a geometria neste problema determinante, pois a forma e as medidas do objeto e dos itens determinam os padrões de corte.

As Figuras 2.3 e 2.4 representam, respectivamen-

te, um problema de corte bidimensional com objeto e itens retangulares e um padrão de corte do problema.

Na próxima seção o problema de corte bidimensional é discutido com mais detalhes, pois este é o objeto de estudo do trabalho.

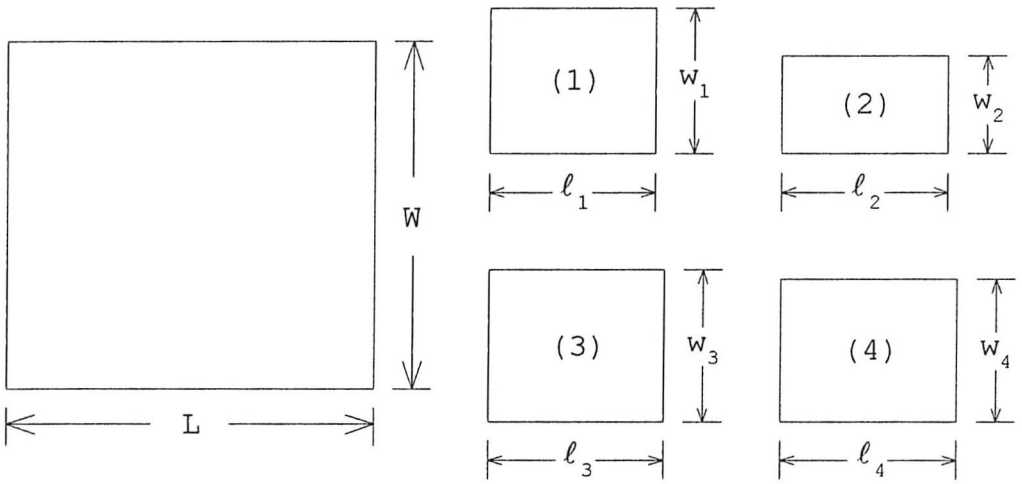

Figura 2.3 - Problema de corte bidimensional

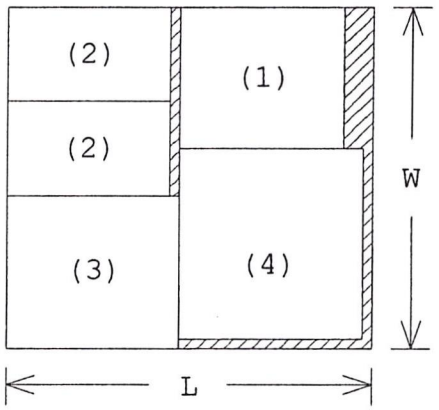

Figura 2.4 - Padrão de corte bidimensional

No problema de corte tridimensional, três dimensões são relevantes; este problema surge, principalmente, na alocação de caixas dentro de caixas maiores. O problema de corte tridimensional e um possível padrão de corte são apresentados nas Figuras 2.5 e 2.6, respectivamente.

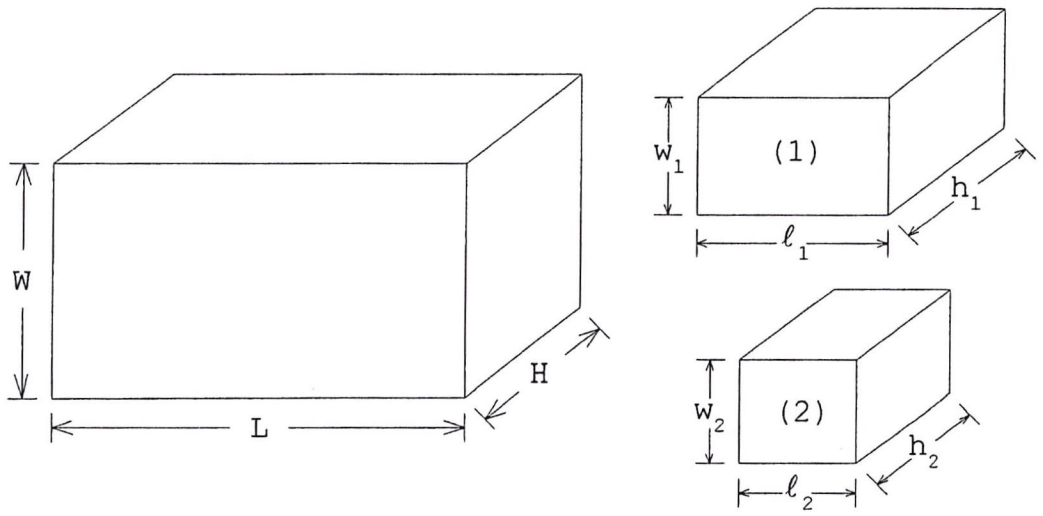

Figura 2.5 - Problema de corte tridimensional

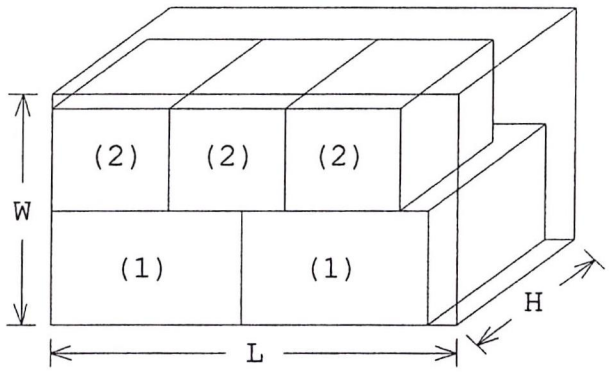

Figura 2.6 - Padrão de corte tridimensional

2.2. O Problema de Corte Bidimensional

Considere uma placa retangular (objeto) de dimensões (L,W), onde L é o comprimento e W é a largura, um conjunto de m peças retangulares (itens) de dimensão (ℓ_i, w_i), onde ℓ_i é comprimento e w_i a largura da peça i, e o valor de utilidade v_i, i=1,...,m. Um problema de corte bidimensional consiste em cortar a placa retangular em peças menores, de forma a otimizar um determinado objetivo, ou seja,

$$\text{maximizar} \sum_{i=1}^{m} v_i.a_i$$

sujeito a: $(a_1, a_2, \ldots, a_m,)$ padrão de corte viável

$a_i \geq 0$, inteiro, i=1,...,m,

onde a_i corresponde ao número de peças i alocadas na placa (L,W).

O corte é chamado guilhotinado quando, aplicado em um retângulo, produz dois novos retângulos, ou seja, o retângulo ou parte dele é cortado somente de modo horizontal ou vertical, conforme a Figura 2.7. Um padrão de corte é do tipo guilhotinado se for obtido por cortes guilhotinados sucessivos.

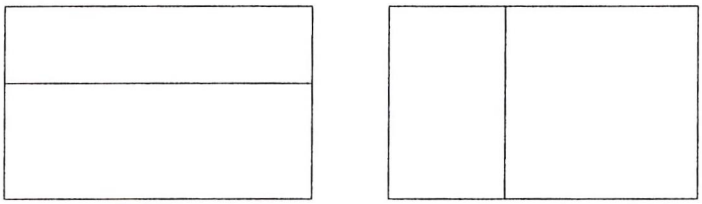

Figura 2.7 - Corte guilhotinado horizontal e vertical

Um corte é do tipo não-guilhotinado se ele produz novos retângulos arranjados de forma que não formam um padrão de corte guilhotinado. Arenales e Morabito (1995) definiram o corte da Figura 2.8 como um corte não-guilhotinado de 1ª ordem. Um padrão de corte não-guilhotinado é de 1ª ordem se for obtido por cortes sucessivos guilhotinados e/ou de 1ª ordem não-guilhotinados.

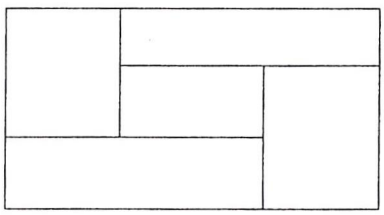

Figura 2.8 - Corte não-guilhotinado

A Figura 2.9 mostra um padrão de corte guilhotinado e um padrão de corte não-guilhotinado.

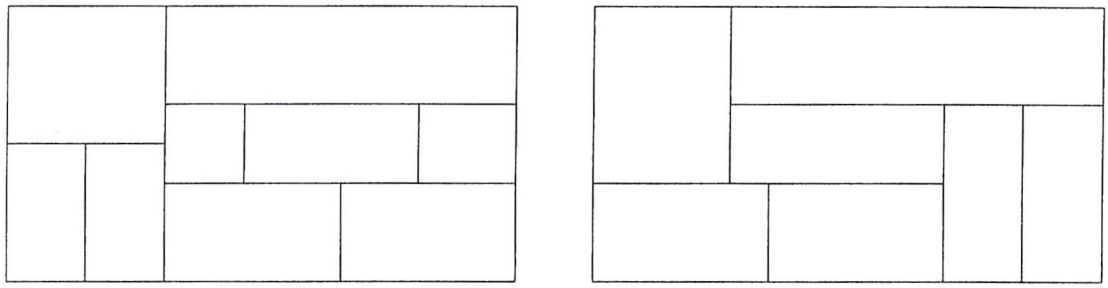

Figura 2.9 - Padrão de corte guilhotinado e não-guilhotinado

Um padrão de corte guilhotinado pode ser tipo estagiado (k-estágios) ou não-estagiado. No primeiro estágio, todos os cortes são feitos de forma paralela a um dos lados da placa, por exemplo, cortes guilhotinados verticais; no segundo estágio, os cortes são ortogonais aos do estágio anterior e assim por diante. Se há um limite k para o número máximo de estágios, o problema é classificado como um problema de corte guilhotinado k-estagiado; caso contrário, o problema é não-estagiado.

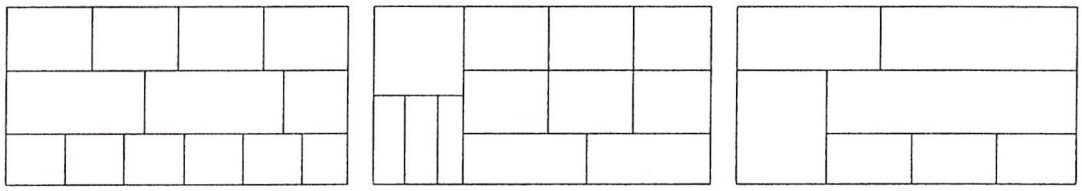

Figura 2.10 - Padrão de corte guilhotinado 2-estágios, 3-estágios e 4-estágios

Na seção seguinte é apresentada a resolução do problema de corte bidimensional através da abordagem em Grafo E/OU.

3. Resolução do Problema através de Grafo E/OU

A abordagem em Grafo E/OU para a resolução de Problemas de Corte foi inicialmente proposta por Morabito (1989), para problemas de corte guilhotinado bidimensional irrestrito e não-estagiado. Em 1992, Morabito estendeu esta abordagem para problemas de corte guilhotinado irrestrito e restrito, considerando as dimensões unidimensional, bidimensional e tridimensional. Ainda se referindo a problemas guilhotinados, Morabito e Arenales (1996) apresentaram a abordagem em grafo E/OU para a resolução de problemas estagiados e restritos.

Esta abordagem também foi proposta para problemas de corte não-guilhotinado. Arenales (1993) e Arenales e Morabito (1995) generalizaram-na para problemas de diferentes dimensões e formas geométricas.

3.1. Representação em Grafo E/OU

Um grafo $G = (V, \varepsilon)$ consiste de um conjunto finito não-vazio de nós (ou vértices) $V=\{1,2,\ldots,r\}$ e um conjunto de arcos (ou arestas) $\varepsilon = \{e_1, e_2, \ldots, e_s\}$, cujos elementos são subconjuntos de V de tamanho 2, isto é, $e_u=(i,j)$, onde $i, j \in V$. Observe que um arco define a relação entre dois nós (Figura 3.1).

Uma maneira de generalizar um grafo é permitir

arcos no conjunto ε de diferentes tamanhos, definindo uma relação entre um subconjunto de nós, por exemplo, um arco $e_u=(i,j,k)$, $i, j, k \in V$. Deste modo, o grafo é chamado de hipergrafo. A Figura 3.2 representa um hipergrafo, onde $V = \{1,2,3,4\}$ e $\varepsilon = \{(1,2), (2,3,4)\}$.

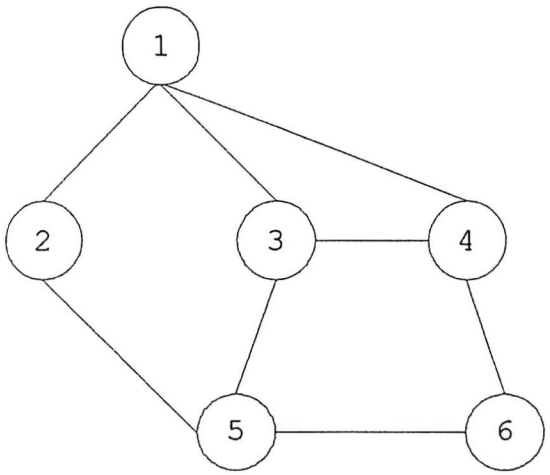

Figura 3.1 - Representação de grafo

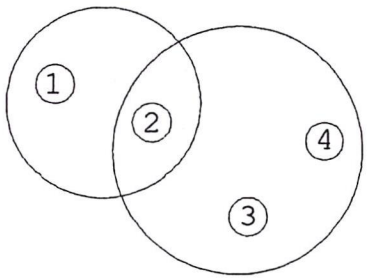

Figura 3.2 - Representação de hipergrafo

Outra maneira de generalizar um grafo é definir arcos como pares $e_u=(i,V_u)$, onde $i \in V$ e $V_u \subset V$, representando a relação entre um nó e um subconjunto de nós, por exemplo, um arco $e_u=(i,\{j,k\})$, onde $i \in V$ e $\{j,k\} \subset V$. Se V_u tem cardinalidade maior que 1, então e_u é chamado um arco E. Os vértices de V_u são sucessores de i. A Figura 3.3, com $V = \{1,2,3,4,5\}$ e $\varepsilon = \{(1, \{2,3\}), (3, \{4,5\})\}$, ilustra este tipo de grafo, conhecido por grafo E/OU (um grafo E/OU é um hipergrafo particular).

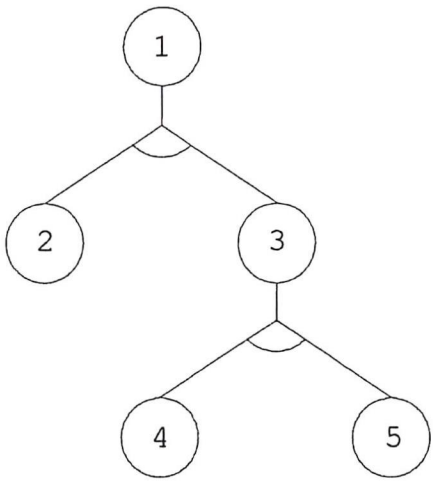

Figura 3.3 - Representação de grafo E/OU

Um grafo E/OU pode ser definido para representar todos os possíveis padrões do problema de corte. Nesta representação, um padrão de corte é representado por um caminho completo no grafo E/OU, onde os nós representam retângulos (no caso do problema de corte bidimensional guilhotinado) e os arcos representam cortes. As Figuras 3.4 e 3.5 ilustram, respectivamente, um padrão de corte e sua representação no grafo E/OU, onde R_i (i=4,5,6,8,10,11) são os retângulos finais e C_k (k=1,2,3,4,5) os cortes aplicados durante o processo de cortagem.

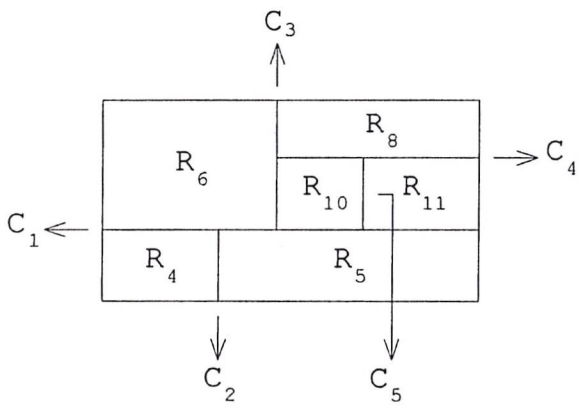

Figura 3.4 - Padrão de corte

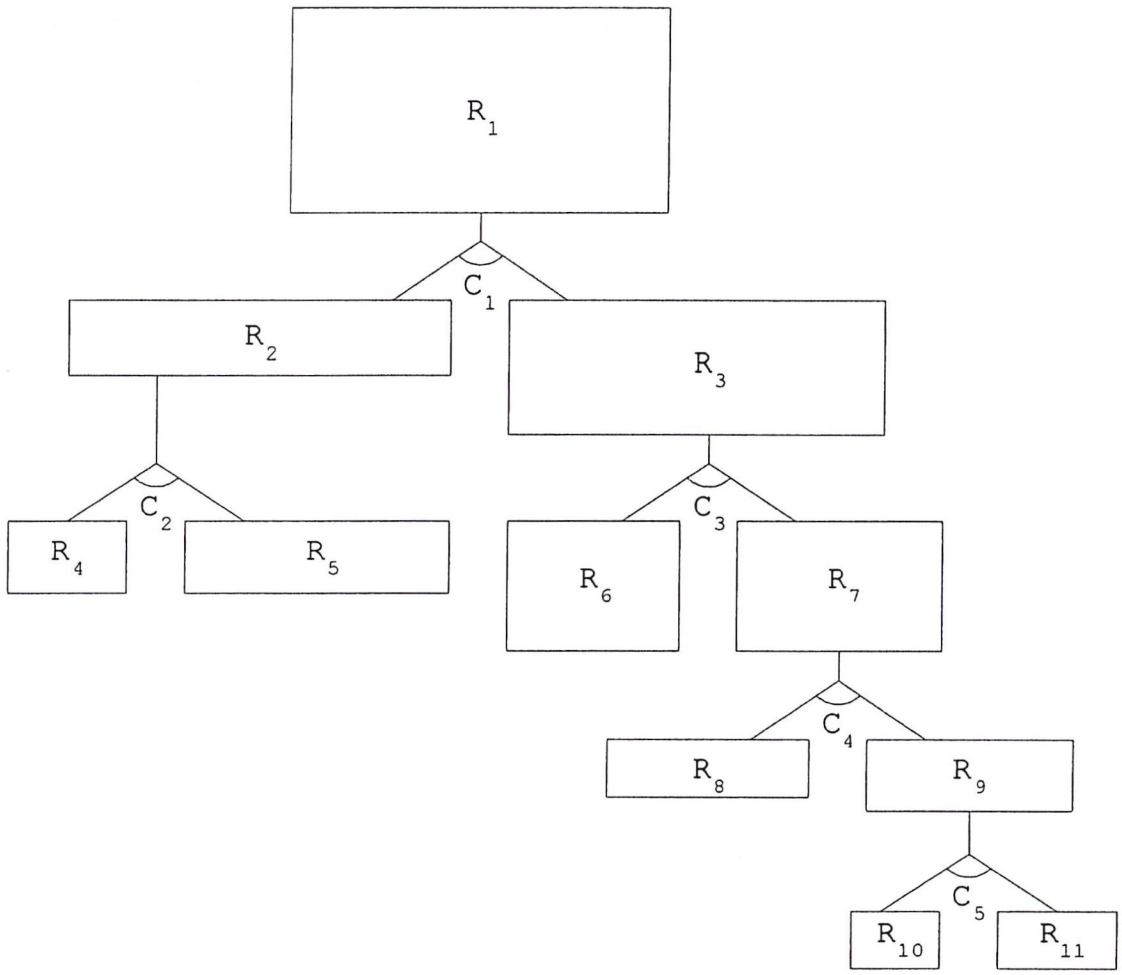

Figura 3.5 - Grafo E/OU representando o padrão da Figura 3.4

Os padrões são gerados examinando todas as possibilidades de corte e uma delas é reproduzir o próprio retângulo (0-corte). O nó inicial é representado pela placa (L,W) e os nós finais são aqueles originados de um 0-corte (sem perda de generalidade, associam-se aos retângulos finais um ou mais itens idênticos), que encerram o processo de cortagem (Figura 3.6). Os cortes (verticais ou horizontais) podem ser restritos, sem perda de generalidade, a um conjunto finito formado pelas combinações lineares não-negativas dos tamanhos dos itens (veja seção 3.2.1), de modo que o grafo que representa todos os possíveis padrões de corte é finito. Arenales (1993) mostrou que esta afirmação é válida para qualquer problema de corte, independentemente das dimensões e das formas geométricas dos objetos e itens.

É fácil perceber que há somente um padrão correspondendo a cada caminho completo no grafo E/OU, mas há pelo menos um caminho completo correspondendo a cada padrão de corte.

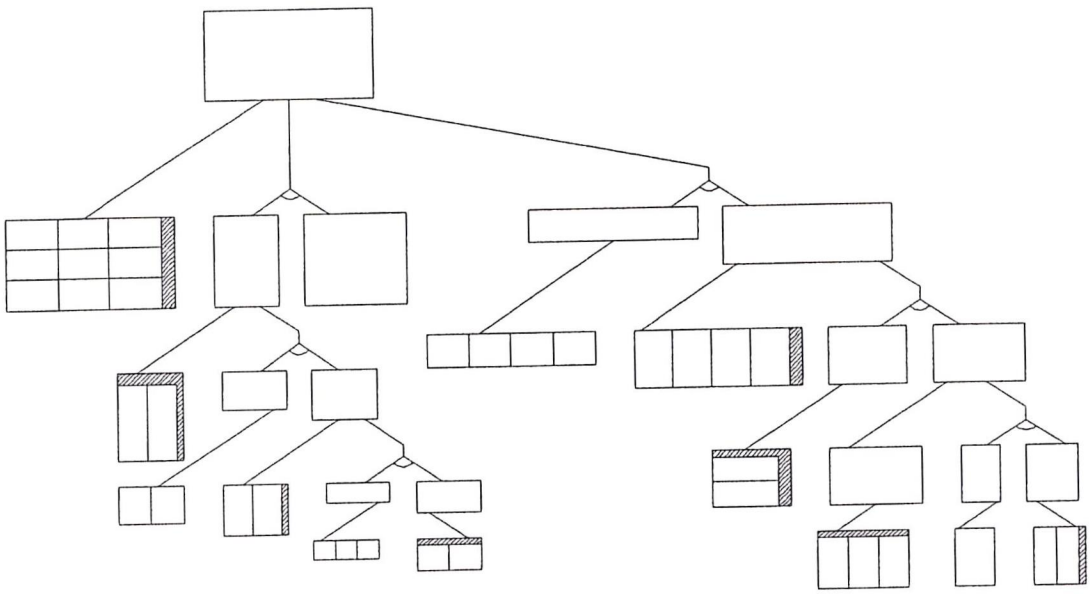

Figura 3.6 - Grafo E/OU com diversos padrões de corte

3.2. Busca no Grafo E/OU

3.2.1. Geração do conjunto de discretização

Durante o processo de busca é possível reduzir o número de nós explicitamente gerados, utilizando as regras apresentadas a seguir.

- **Padrões Normais**

Herz (1972) e mais tarde Christofides e Whitlock (1977) mostraram, sem perda de generalidade, que os cortes guilhotinados (Arenales e Morabito, 1995, estenderam para cortes não-guilhotinados) podem ser combinações lineares não-negativas das dimensões das peças, ou seja, é possível reduzir os cortes verticais ao longo do comprimento L aos elementos do

conjunto X:

$$X = \left\{ x \mid x = \sum_{i=1}^{m} \alpha_i . \ell_i,\ 1 \leq x \leq L - \ell_o,\ \alpha_i \geq 0 \text{ e inteiro} \right\}$$

onde $\ell_0 = \min\{\ell_i, i = 1, \ldots, m\}$.

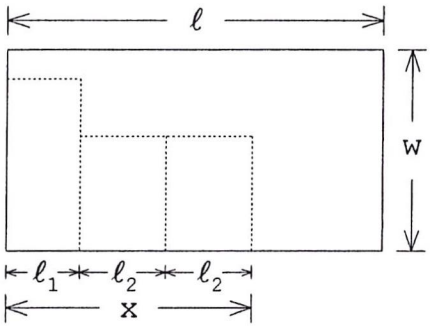

Figura 3.7 - Padrão Normal ($x=\ell_1+2.\ell_2$)

Analogamente, o conjunto Y é definido para os cortes horizontais ao longo da largura W.

$$Y = \left\{ y \mid y = \sum_{i=1}^{m} \beta_i . w_i,\ 1 \leq y \leq W - w_o,\ \beta_i \geq 0 \text{ e inteiro} \right\}$$

onde $w_0 = \min\{w_i, i = 1, \ldots, m\}$.

Os conjuntos X e Y são chamados de conjuntos de discretização.

- **Simetria**

Christofides e Whitlock também mostraram, sem perda de generalidade, que o conjunto X pode ser reduzido considerando o efeito simetria. Considere um nó (ℓ, w) e um corte vertical $x_1 \in X$ e $x_1 \leq L - \ell_o$, gerando os nós (x_1, w) e $(\ell - x_1, w)$. Considere, ainda, $x_2 = \ell - x_1$, logo $x_1 = \ell - x_2$. Se $x_2 \in X$, então os nós

(x_2,w) e $(\ell-x_2,w)$ também são gerados, produzindo uma duplicação dos nós (x_1,w) e (x_2,w) (Figura 3.8). O conjunto X pode ser reduzido a:

$$X = \left\{ x \mid x = \sum_{i=1}^{m} \alpha_i . \ell_i,\ 1 \leq x \leq \left\lfloor \frac{L}{2} \right\rfloor,\ \alpha_i \geq 0 \text{ e inteiro} \right\}.$$

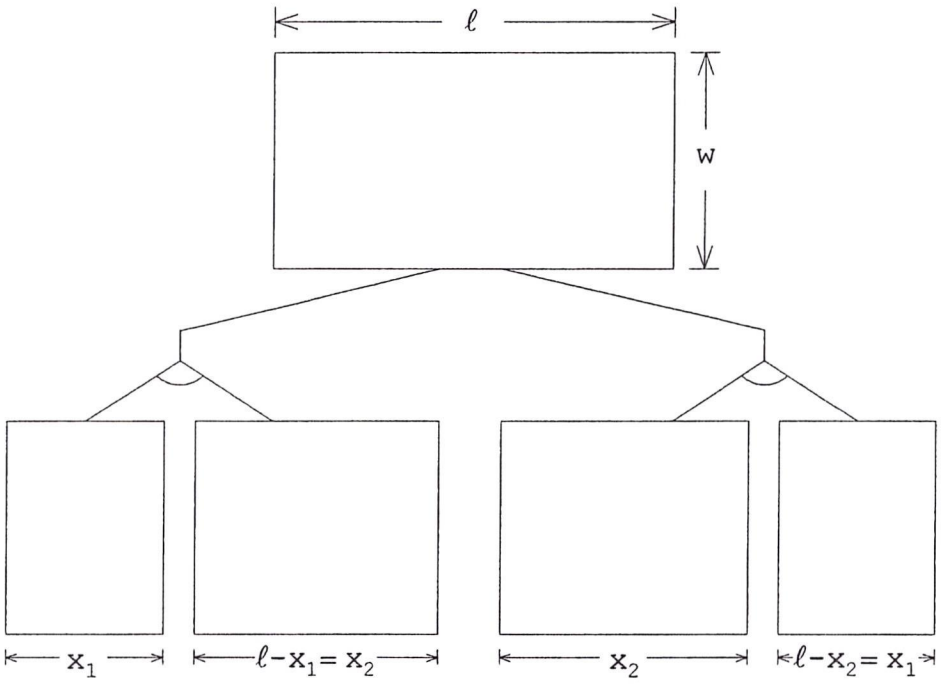

Figura 3.8 - Simetria

O conceito de simetria também é aplicado na geração do conjunto Y dos cortes horizontais.

- **Exclusão**

O conceito de exclusão se refere ao fato de descartar cortes inviáveis durante o processo de busca. Considere o retângulo (ℓ,w) e o corte $x=\ell_1+\ell_2$, tal que $w_1<w$ e $w_2>w$ (Figura 3.9). Observe que este corte deve ser descartado pois, apesar de $x=\ell_1+\ell_2$ e $w_1<w$, tem-se $w_2>w$.

$\lfloor x \rfloor$ denota o maior inteiro menor ou igual a x.

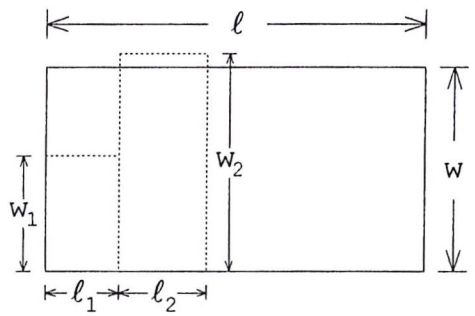

Figura 3.9 - Exclusão ($x=\ell_1+\ell_2$ pode ser excluído do conjunto X)

Assim, o conjunto X pode novamente ser reduzido a:

$$X = \left\{ x \mid x = \sum_{i=1}^{m} \alpha_i . \ell_i, (\text{se } w_i > w \Rightarrow \alpha_i = 0), 1 \leq x \leq \left\lfloor \frac{L}{2} \right\rfloor, \alpha_i \geq 0 \text{ e inteiro} \right\}$$

O conceito de exclusão é aplicado também na geração dos cortes horizontais do conjunto Y.

Christofides e Whitlock (1977) propuseram uma fórmula recursiva para a geração do conjunto de discretização X e Y. Inicialmente, os conjuntos de discretização são determinados para o nó inicial (L,W) e, depois disto, eles são facilmente determinados para qualquer nó N.

Considere $\ell_1 \leq \ell_2 \leq \ldots \leq \ell_m$ e $F_i(x)$ o mínimo da maior largura das peças $1,2,\ldots,i$ ($i \leq m$) que pode ser combinada para formar x. Então,

$$F_i(x) = \begin{cases} \min \left\{ F_{i-1}(x) ; \max \left\{ w_i, \min_{\rho} \left\{ (F_{i-1}(x - \rho.\ell_i), 1 \leq \rho \leq \left\lfloor \frac{x}{\ell_i} \right\rfloor \text{ e inteiro} \right\} \right\} \right\} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \ell_i \leq x \leq \left\lfloor \frac{L}{2} \right\rfloor \\ F_{i-1}(x), \; 0 < x < \ell_i, \end{cases}$$

sendo $F_0(x)=\infty$, para $x=1,2,\ldots,\left\lfloor \frac{L}{2} \right\rfloor$, e $F_i(0)=0$.

Se $F_i(x)<\infty$, então $x = \sum_{j=1}^{i} \alpha_j.\ell_j$, para $\alpha_j \geq 0$ e inteiro, $j=1,\ldots,i$; logo $x \in X$.

Já que $F_i(x)$ independe do nó, o conjunto de discretização X pode ser gerado antes do início do processo de busca. Depois, o conjunto de discretização do nó N, X(N), pode ser gerado facilmente. Suponha que o nó N represente o retângulo (ℓ,w); então $x \in X(N) \Leftrightarrow F_m(x) \leq w$.

De maneira análoga, aplica-se ao conjunto Y.

- **Ordenação de cortes**

Considere o retângulo (ℓ,w) de um determinado nó N e $x_1 \in X$ o corte vertical aplicado ao retângulo, originando dois novos retângulos (x_1,w) e $(\ell-x_1,w)$. A seguir, considere o retângulo gerado $(\ell-x_1,w)$ e um novo corte vertical $x_2 \in X$, originando os retângulos (x_2,w) e $(\ell-x_1-x_2,w)$. Estes três retângulos, (x_1,w), (x_2,w) e $(\ell-x_1-x_2,w)$, também podem ser gerados aplicando inicialmente o corte $x_2 \in X$, gerando (x_2,w) e $(\ell-x_2,w)$ e, a seguir, aplicando o corte $x_1 \in X$ em $(\ell-x_2,w)$, gerando (x_1,w) e $(\ell-x_2-x_1,w)$ (Figura 3.10).

Sem perda de generalidade, esta duplicação pode ser evitada utilizando uma ordem arbitrária nos sucessivos cortes verticais. Este critério é válido também para os cortes horizontais (Christofides e Whitlock, 1977).

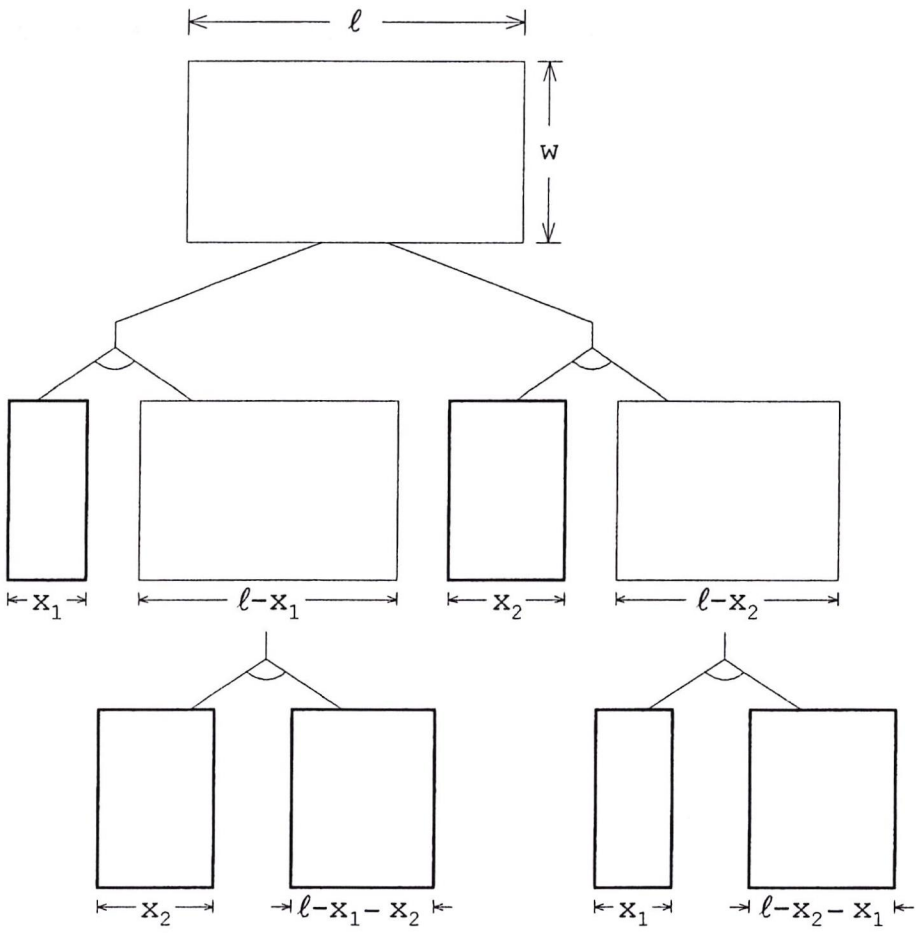

Figura 3.10 - Ordenação

Durante o processo de busca, enumerar explicitamente todos os caminhos do grafo é, na maioria das vezes, inviável. As soluções podem ser enumeradas implicitamente, ou seja, é possível descartar a expansão de um nó sem perder a solução ótima, usando seus limitantes.

3.2.2. Limitante Inferior

É usual, em problemas de otimização (maximização), a determinação de limitantes inferiores através da obtenção de soluções viáveis. Nos problemas de corte, soluções viáveis são facilmente determinadas.

Uma solução trivial para o subproblema do nó (x,y) é preenchê-lo somente com peças iguais, conforme os padrões das Figura 3.11 e 3.12. Um padrão de corte que contém

peças todas iguais é denominado padrão de corte homogêneo.

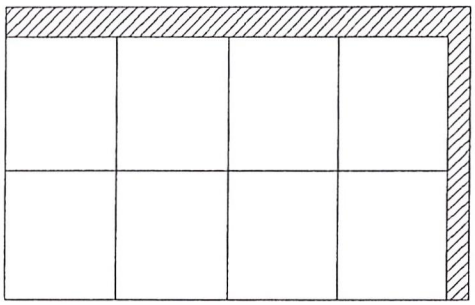

Figura 3.11 - Padrão de corte homogêneo

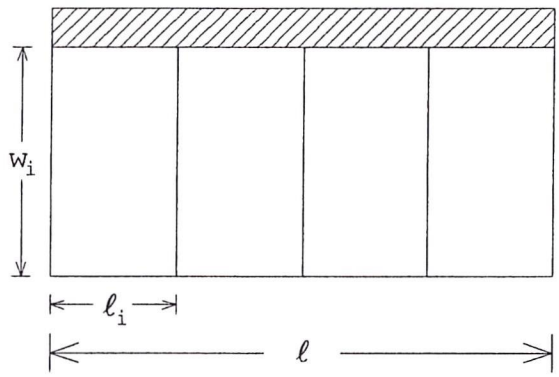

Figura 3.12 - Padrão homogêneo com apenas cortes verticais

Escolhendo-se o melhor dentre estes padrões, tem-se o limitante inferior dado por:

$$LI(x, y) = \max_{1 \le i \le m} \left\{ v_i \cdot \left\lfloor \frac{x}{\ell_i} \right\rfloor \cdot \left\lfloor \frac{y}{w_i} \right\rfloor \right\}$$

3.2.3. Limitante Superior

Como é usual em problemas de otimização (maximização), um limitante superior é obtido pela relaxação do problema.

Um limitante superior para o nó (x,y) é definido relaxando o problema de corte, considerando-se apenas que a área dos itens alocados não exceda a área do retângulo, ou seja,

$$LS(x, y) = \text{maximizar} \sum_{i \in A(x,y)} v_i . a_i$$

$$\text{sujeito a : } \sum_{i \in A(x,y)} (\ell_i . w_i) . a_i \leq (x.y)$$

$$a_i \geq 0, i \in A(x, y)$$

onde $A(x, y) = \{i \mid \ell_i \leq x \text{ e } w_i \leq y\}$ com a seguinte solução trivial

$$LS(x, y) = \begin{cases} \max\limits_{i \in A(x,y)} \left\{ v_i . \left(\frac{x}{\ell_i} \right) . \left(\frac{y}{w_i} \right) \right\} \\ 0, \text{ se } A(x, y) = \phi \end{cases}$$

3.2.4. Estratégia de busca

A estratégia de busca utilizada é conhecida na literatura e combina busca *backtracking* (BT) e *hill-climbing* (HC) (Morabito, 1989).

A estratégia de busca *backtracking* é uma variação da estratégia de busca em profundidade (*depth-first*), que consiste em ramificar primeiro os nós gerados mais recentemente. A profundidade de um grafo é definida tal que o nó raiz tem profundidade zero e a profundidade de qualquer outro nó é obtida somando 1 na profundidade do nó que o gerou. No caso da estratégia utilizada, um limite de profundidade máximo deve ser estabelecido, de forma que quando um nó atingir este limite, não possui sucessores.

Morabito combinou a estratégia *backtracking* com uma busca baseada em otimizações locais (*hill-climbing*). Nesta estratégia híbrida, depois de gerados todos os caminhos de um nó até a profundidade máxima permitida, escolhe-se o melhor deles, descartando-se os demais, e expandem-se os nós de maior profundidade aplicando, novamente, *backtracking* até a profundidade permitida.

Considere o nó N, seus nós sucessores N_1 e N_2 e

v(N) o valor do nó N dado pelo seu limitante inferior (valor de uma solução viável, por exemplo, a solução homogênea). Durante a expansão deste nó, utilizam-se os limitantes inferiores e superiores para evitar a geração de nós desnecessários, sem perder caminhos não promissores. Assim,

- se $v(N)=LS(N)$, então v(N) é solução ótima.

- se $v(N)<LI(N_1)+LI(N_2)$, então o valor do nó N é atualizado por $v(N)=LI(N_1)+LI(N_2)$.

- se $v(N)\geq LS(N_1)+LS(N_2)$, ou seja, se o valor obtido até o momento no nó N é melhor que a soma dos limitantes superiores de seus sucessores, então não é necessário expandir o nó N.

Observe que o valor de um caminho é dado pela soma dos valores dos nós gerados no processo de busca e o melhor caminho é determinado por aquele que apresenta o melhor valor v(N).

Note que, com o limitante na profundidade do grafo e a estratégia *hill-climbing*, perde-se a otimalidade do problema de corte.

A seguir, é apresentado o algoritmo para esta estratégia de busca híbrida.

- **Algoritmo BT-HC**

Passo 0

Considere a placa (L,W), as peças (ℓ_i, w_i), i=1,...,m, e MP, a profundidade máxima permitida na estratégia de busca. Considere o nó raiz como o nó que contém as informações da placa inicial.

Passo 1 - *Backtracking*

Aplique a estratégia de busca *backtracking* com profundidade MP a partir do nó raiz, armazenando o caminho que apresenta o melhor valor para este nó.

Passo 2 - *Hill-Climbing*

Para cada nó final do caminho gerado pela estratégia *backtracking*, verifique se é possível expandi-lo. Se for, retorne ao Passo 1, considerando-o como nó raiz.

O algoritmo, apesar de percorrer todo o conjunto de discretização X e Y, ou seja, gerar muito cortes, não requer muita memória computacional, uma vez que ele armazena somente o melhor caminho em cada nó.

3.3. Heurísticas

Para reduzir o espaço de busca, recorre-se a algumas heurísticas que podem ser aplicadas no algoritmo, de forma a diminuir o espaço de busca (Morabito, 1992; Arenales e Morabito, 1995; Morabito e Arenales, 1996). Considere um nó qualquer N e seus sucessores N_1 e N_2.

- H1 (Arenales, 1996)
  Se $(1 + \lambda_1).v(N) \geq LS(N_1) + LS(N_2)$ com $\lambda_1 > 0$, os nós N_1 e N_2 são descartados como não-promissores.

- H2 (Arenales, 1996)
  Se $\lambda_2 . LI(N) \geq LI(N_1) + LI(N_2)$ com $0 < \lambda_2 < 1$, os nós N_1 e N_2 são descartados.

- H3 (Beasley, 1985)
  Esta heurística introduz uma limitação no número de elementos dos conjuntos de discretização X e Y.

Considere N o conjunto com o índice das peças utilizadas para a geração do conjunto X, sendo, inicialmente, $N = \{1,2,\ldots,m$, $|X|$ o número de elementos deste conjunto, onde

$$X = \left\{ x \mid x = \sum_{i \in N} \alpha_i . \ell_i,\ 1 \leq x \leq L - \min\{\ell_j, j \in N\}\ \alpha_i \geq 0 \text{ e inteiro} \right\}$$

A heurística consiste em diminuir o tamanho do conjunto X da maneira que segue, sendo XMax o tamanho máximo permitido:

Passo 0: $N = \{1,2,\ldots,m\}$

$|X| = +\infty$

Passo 1: Enquanto $|X| > XMax$ faça

Geração do conjunto X

Se $|X| > Xmax$ então

$\ell_j = \min\{\ell_i, i \in N\}$

$N = N-\{j\}$

Fim se

Fim enquanto

O procedimento é análogo para o conjunto de discretização Y.

- H4

Esta sugestão se refere à seleção dos pontos nos conjuntos de discretização X e Y, ou seja, o modo como eles serão examinados:

- percorrê-lo em ordem crescente, examinando assim, na estratégia *backtracking*, sempre os menores retângulos primeiro;
- percorrê-lo percorrê-lo em ordem decrescente, preenchendo primeiro os retângulos maiores;
- percorrê-lo de maneira aleatória, ou seja, não é possível prever como serão feitos os cortes.

A seguir, são apresentados os resultados compu-

tacionais obtidos com estas heurísticas para problemas de corte irrestrito.

4. Implementação Computacional

O algoritmo descrito foi implementado na linguagem Pascal, utilizando o compilador Borland Pascal versão 7.0, em um microcomputador Pentium, 167 Mhz com 32 Mbytes de RAM.

Inicialmente, foi feita a implementação da geração do conjunto de discretização utilizando as fórmulas de Christofides e Whitlock, descritas na seção 3.2.1.

Como as fórmulas são recursivas, escreveu-se uma rotina de geração de cortes também usando a técnica de programação de recursividade.

A recursividade é uma técnica em que uma rotina, no processo de execução de suas tarefas, chama a si própria. Por este motivo, é necessária uma área de memória para salvar o seu contexto antes dessa chamada recursiva (pilha de registros), a fim de que este possa ser restaurado ao final dessa chamada. Sua principal vantagem é a redução do código-fonte da rotina; entretanto, o uso de recursividade apresenta algumas desvantagens, entre elas o baixo desempenho na execução, devido ao tempo gasto no gerenciamento da pilha de registros e o espaço por ela ocupado, e a dificuldade de depuração dos programas (Villas *et al.*, 1993).

Foram feitos diversos testes com a rotina recursiva, observando-se que o tempo computacional aumenta exponencialmente em função da dimensão do problema (veja seção 5.1). Os dados foram gerados novamente, em um microcomputador Pen-

tium II, com 128 Mbytes de RAM; entretanto o tempo computacional obtido é grande quando comparado com o tempo obtido de uma versão não recursiva.

Portanto, apesar da clareza e elegância do algoritmo recursivo na implementação das fórmulas de Christofides e Whitlock, o desempenho ficou muito longe do esperado.

A rotina foi, então, implementada de maneira iterativa; o código ficou bem mais extenso e não tão fácil de se compreender, mas o desempenho do algoritmo compensa estas perdas.

Exemplos de comparações entre as duas técnicas de programação, recursiva e iterativa, para a geração do conjunto de discretização são vistas no próximo capítulo.

A estrutura de dados utilizada para o armazenamento dos conjuntos de discretização (X e Y), lembrando que eles são definidos inicialmente e, durante o processo de busca, são facilmente determinados para qualquer nó (x,y), foi uma lista seqüencial usando alocação estática. Neste tipo de alocação, a estrutura de dados utilizada deve estar completamente definida e é alocado um espaço finito e pré-determinado de memória, não permitindo alteração durante o processo de execução do programa.

No caso do conjunto de discretização, foi definido um vetor (em Pascal, *array*) com no máximo 100 posições. Portanto, se o conjunto tiver apenas 20 elementos, as demais posições ficarão vazias, porém ocupando espaço de memória. Caso seja necessário, não é possível ter mais de 100 cortes, já que o tamanho máximo do vetor deve ser definido durante a fase de programação. A vantagem desta estrutura de dados é o acesso direto a cada posição do vetor. Um exemplo deste tipo de estrutura armazenando um vetor de números inteiros é ilustrado pela Figura 4.1.

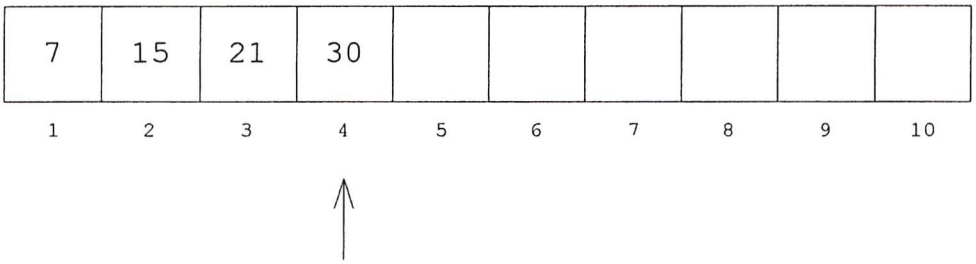

Figura 4.1 - Lista seqüencial

Também implementou-se a geração do conjunto utilizando uma lista encadeada usando alocação dinâmica (Figura 4.2). Este tipo de alocação permite que a estrutura de dados não tenha um tamanho pré-definido, ficando limitado à memória do computador.

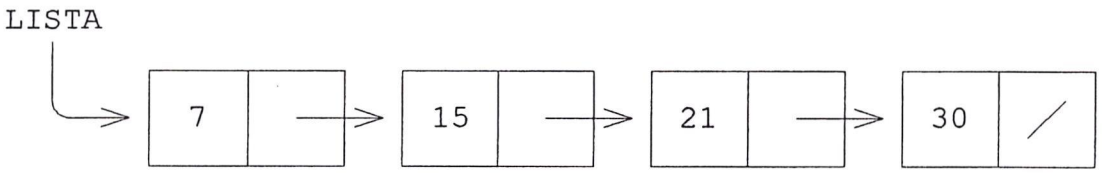

Figura 4.2 - Lista encadeada

Note que são alocados apenas os elementos que são definidos; entretanto, para acessar o último elemento da lista, é necessário percorrer todos os anteriores. Este tipo de estrutura também oferece uma flexibilidade para o caso de muitas inserções e remoções de elementos na lista.

Um outro tipo de estrutura de dados, que pode ser utilizado para a representação do conjunto de discretização, é uma árvore binária ordenada (Figura 4.3). Nesta estrutura, assim como na estrutura com listas encadeadas, não é necessário definir seu tamanho previamente.

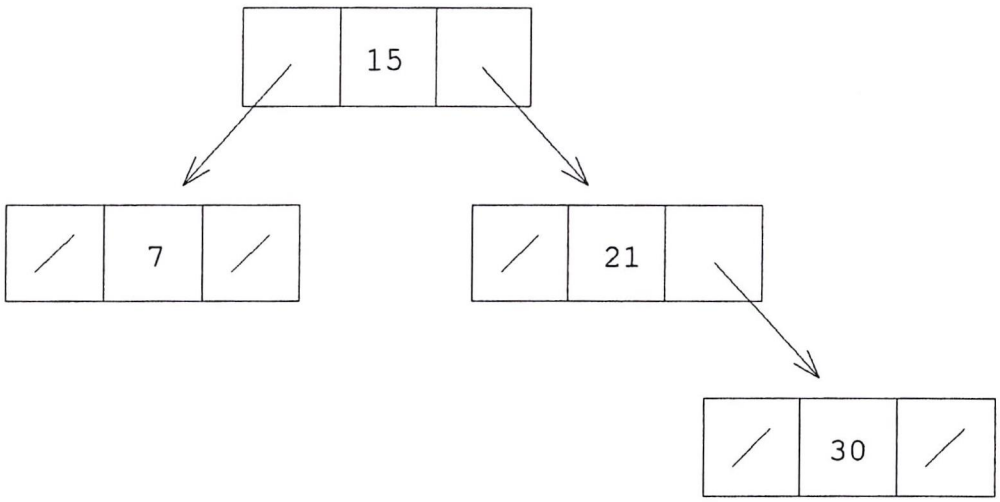

Figura 4.3 - Árvore binária ordenada

Este tipo de estrutura necessita de um tempo computacional maior que a lista encadeada para sua construção, entretanto, como ela está ordenada, o tempo computacional para a procura de um elemento é menor.

No caso do conjunto de discretização, após a implementação usando dois tipos de estrutura de dados, lista seqüêncial e lista encadeada, optou-se pela primeira, já que a heurística utilizada (vista na seção 3.3) limita o tamanho do conjunto e o acesso aos elementos da lista seqüencial é mais rápido, uma vez que, depois dos conjuntos de discretização gerados, basta percorrê-los para gerar os conjuntos de um determinado nó.

Na implementação do algoritmo de busca, utilizou-se alocação dinâmica para a representação do grafo, tendo sido armazenados os seguintes dados para cada nó:

Informações do nó N
Endereço do nó N_1
Endereço do nó N_2

Figura 4.4 - Dados armazenados em cada nó do grafo em problemas guilhotinados

As informações do nó N se referem à dimensão do objeto representado pelo nó, ao valor v(N) obtido até o nó e aos limitantes inferior e superior, entre outros dados. Este conjunto de informações varia de acordo com o problema e as restrições impostas por ele.

Este tipo de estrutura permite que se tenha um grafo gerado conforme a Figura 4.5. Lembre que o algoritmo armazena um único caminho do grafo.

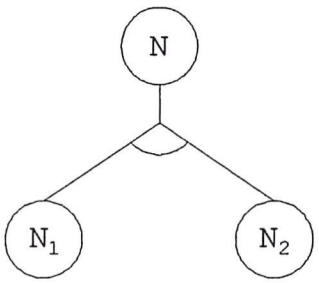

Figura 4.5 - Estrutura do nó armazenado

Com este tipo de estrutura, se o problema permitir gerar mais que dois nós (por exemplo, o problema de corte não-guilhotinado da Figura 2.8), basta que se inclua nos dados do nó o endereço dos demais nós, ou seja,

Informações do nó N
Lista encadeada com o endereço dos nós gerados

Figura 4.6 - Dados armazenados em cada nó do grafo em problemas não-guilhotinados

Note que o tamanho da lista encadeada depende do número de sucessores e, no caso do corte não-guilhotinado (Figura 2.8), cada nó do grafo gera no máximo 5 novos nós (Figura 4.7).

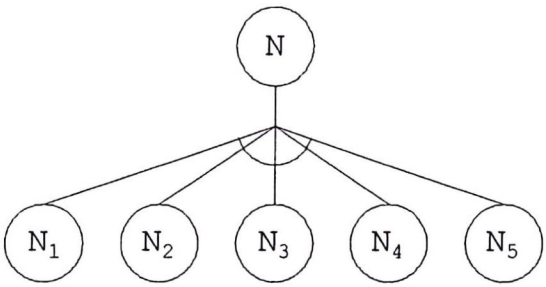

Figura 4.7 - Estrutura do nó gerado pelo corte não-guilhotinado

A implementação do problema bidimensional irrestrito foi comparada com a implementação feita por Morabito (1992). Nesta implementação, a estrutura de dados foi definida de forma que cada nó do grafo armazene as informações dos dois nós (N_1 e N_2) gerados após o corte, ou seja,

Informações do nó anterior
Informações sobre o nó N_1
Informações sobre o nó N_2

Figura 4.8 - Dados armazenados em cada nó do grafo em problemas guilhotinados (Morabito, 1992)

que pode ser esquematizada pela Figura 4.9.

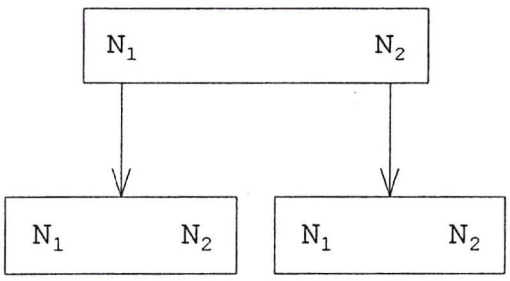

Figura 4.9 - Estrutura gerada por Morabito (1992)

Para o problema de corte bidimensional guilhotinado, estes dois tipos de estruturas são equivalentes (Figuras 4.5 e 4.9). Entretanto, para um problema que permita outros tipos de corte, por exemplo, o corte não-guilhotinado da Figu-

ra 2.8, esta segunda estrutura necessita de mais memória.

Os resultados computacionais obtidos com esta implementação são apresentados no próximo capítulo.

5. Resultados Computacionais

Neste capítulo são apresentados os resultados computacionais da geração do conjunto de discretização (seção 3.2.1) e do algoritmo BT-HC descrito na seção 3.2.4 para o problema de corte bidimensional irrestrito, alterando o modo de seleção de pontos do conjunto de discretização.

5.1. Conjunto de discretização

Para análise dos resultados obtidos na geração do conjunto de discretização foram gerados exemplos aleatoriamente. Estes exemplos foram divididos em duas classes, exemplos pequenos e grandes, de acordo com Morabito (1992).

- Exemplos pequenos: nesta classe de problema, os conjuntos de discretização resultam com poucos elementos. Os valores ℓ_i e w_i, i=1,...,m, foram gerados aleatoriamente e, em seguida, arredondados, dentro dos intervalos [0.25L,0.75L] e [0.25W,0.75W], respectivamente.

- Exemplos grandes: os conjuntos de discretização resultam com muitos elementos, tendo sido os valores ℓ_i e w_i, i=1,...,m, gerados aleatoriamente e, em seguida, arredondados, dentro dos intervalos [0.10L,0.50L] e [0.10W,0.50W], respectivamente.

A Tabela 5.1 mostra os resultados obtidos na geração dos conjuntos de discretização dos problemas pequenos. A fórmula utilizada para o cálculo foi apresentada na seção 3.2.1, entretanto, para a geração das Tabelas 5.1 e 5.2, não se considerou a regra de simetria. A tabela foi dividida em 9 colunas, onde a primeira indica o número de exemplos contidos numa classe de problemas e, a segunda, terceira e quarta se referem ao número de peças (m) e dimensões da placa (L,W), respectivamente. Nas duas colunas seguintes, tem-se a média aritmética do número de elementos dos conjuntos X e Y juntamente com o desvio padrão. As últimas três colunas referem-se ao tempo computacional obtido, em segundos, utilizando na geração dos conjuntos a recursividade com alocação dinâmica (1), a recursividade com alocação estática (2) e o algoritmo iterativo com alocação estática (3). Note que, são apresentados dois tempos computacionais distintos para cada classe de problemas, o tempo obtido em um microcomputador Pentium, 167 Mhz, com 32 Mbytes de RAM (linha superior), e o tempo obtido com um microprocessador Pentium II, 200 Mhz, com 128 Mbytes de RAM (linha inferior).

Nas tabelas, os tempos computacionais indicados por (*) foram superiores a uma hora.

Tabela 5.1 - Discretização nos exemplos pequenos

Qte	m	L	W	|X| (DP)	|Y| (DP)	T(s) (1)	T(s) (2)	T(s) (3)
100	5	100	100	15.40(5.28)	13.50(5.52)	0.05	0.06	0.00
						0.02	0.03	0.00
100	10	100	100	34.70(8.34)	36.50(7.74)	0.29	0.16	0.00
						0.10	0.05	0.00
100	20	100	100	56.80(2.93)	56.20(3.52)	1.44	0.78	0.00
						0.35	0.17	0.01
100	30	100	100	65.90(2.34)	65.20(2.23)	6.06	2.92	0.01
						0.86	0.41	0.00
100	5	1000	1000	17.70(6.83)	15.30(9.02)	1.16	0.56	0.00
						0.37	0.17	0.01

100	10	1000	1000	48.10(13.50)	49.50(19.58)	3.23	1.53	0.01
						1.00	0.46	0.01
100	20	1000	1000	139.80(31.88)	143.30(22.67)	13.81	7.44	0.01
						3.36	1.52	0.01
100	30	1000	1000	250.00(36.00)	280.00(28.00)	63.89	29.85	0.02
						7.96	3.89	0.02
100	50	1000	1000	359.20(32.23)	405.20(25.69)	194.12	95.49	0.06
						26.74	13.38	0.03
50	5	3000	3000	12.80(6.21)	9.40(4.22)	2.64	1.24	0.01
						0.90	0.44	0.01
50	10	3000	3000	49.00(22.91)	54.20(29.58)	10.12	4.96	0.02
						2.91	1.31	0.00
50	20	3000	3000	218.00(71.61)	135.20(34.37)	37.23	18.45	0.07
						9.62	4.36	0.01
50	30	3000	3000	365.20(67.49)	263.00(37.14)	139.87	71.44	0.09
						21.73	10.68	0.02
50	50	3000	3000	660.60(37.98)	700.00(98.68)	(*)	(*)	0.13
						68.60	35.03	0.05
50	50	5000	5000	842.80(109.21)	1005.20(165.39)	(*)	(*)	0.23
						132.35	66.13	0.09

Tabela 5.2 - Discretização nos exemplos grandes

Qte	M	L	W	|X| (DP)	|Y| (DP)	T(s) (1)	T(s) (2)	T(s) (3)
100	5	100	100	49.80(10.65)	52.40(17.96)	1.54	0.71	0.00
						0.43	0.20	0.00
100	10	100	100	73.70(4.17)	76.00(6.23)	10.45	4.74	0.00
						2.27	1.03	0.00
100	20	100	100	83.70(3.61)	84.20(2.23)	51.94	23.55	0.00
						13.70	6.17	0.00
100	30	100	100	88.20(1.17)	88.50(1.28)	190.09	99.34	0.01
						71.55	32.27	0.00
100	5	1000	1000	70.70(42.67)	97.40(70.70)	11.57	5.74	0.01
						4.09	1.86	0.01
100	10	1000	1000	323.40(130.66)	276.30(60.32)	45.03	22.09	0.00
						15.67	7.05	0.01
100	20	1000	1000	541.30(65.84)	565.10(91.14)	403.91	196.94	0.03
						171.32	76.92	0.01
100	30	1000	1000	656.50(17.50)	666.00(23.00)	(*)	(*)	0.06
						384.67	171.71	0.00

100	50	1000	1000	735.80(29.92)	756.40(11.72)	(*) 915.12	(*) 303.01	0.09 0.04
50	5	3000	3000	124.40(74.95)	101.80(69.71)	49.63 16.40	23.23 7.42	0.01 0.01
50	10	3000	3000	364.00(34.97)	432.60(213.94)	(*) 31.64	(*) 14.23	0.05 0.03
50	20	3000	3000	1329.00(133.43)	1337.00(224.56)	(*) 412.44	(*) 185.21	0.10 0.04
50	30	3000	3000	1804.80(147.21)	1757.00(115.54)	(*) 1019.23	(*) 433.09	0.15 0.06
50	50	3000	3000	2076.80(55.36)	1898.40(72.17)	(*) (*)	(*) (*)	0.25 0.09
50	50	5000	5000	2898.00(61.24)	2880.80(186.17)	(*) (*)	(*) (*)	0.60 0.15

Observe que o tempo computacional para a geração dos conjuntos de discretização cresce exponencialmente com o tamanho do problema. A execução dos problemas em processadores mais velozes (tempos computacionais apresentados na linha inferior de cada classe de problema) não torna viável a utilização da técnica de recursividade, pois apesar do tempo computacional diminuir bastante na execução destes problemas, ele ainda é muito grande em relação ao tempo obtido com o algoritmo recursivo, além de na prática ser inviável, pois os computadores utilizados têm uma memória menor que 128 Mbytes. Logo, pode-se concluir que a aplicação da recursividade na geração dos conjuntos não é um bom exemplo de sua utilização, sendo de grande valia para outras aplicações.

Nos gráficos 5.1 e 5.2, a seguir, é possível observar que o tempo cresce exponencialmente conforme o tamanho do problema. Note também que o uso de um microprocessador mais veloz e uma quantidade maior de memória RAM diminui significativamente o tempo computacional, porém ainda é muito maior comparado ao tempo obtido com o processo iterativo.

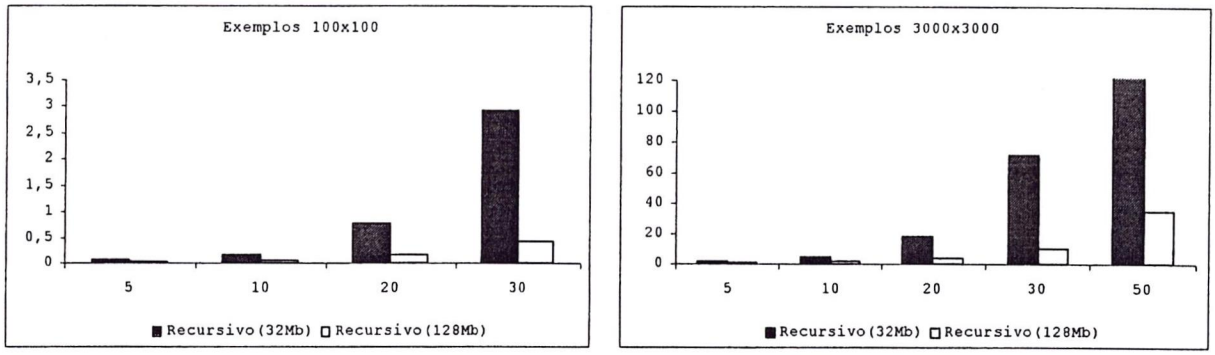

Gráfico 5.1 - Discretização nos exemplos pequenos

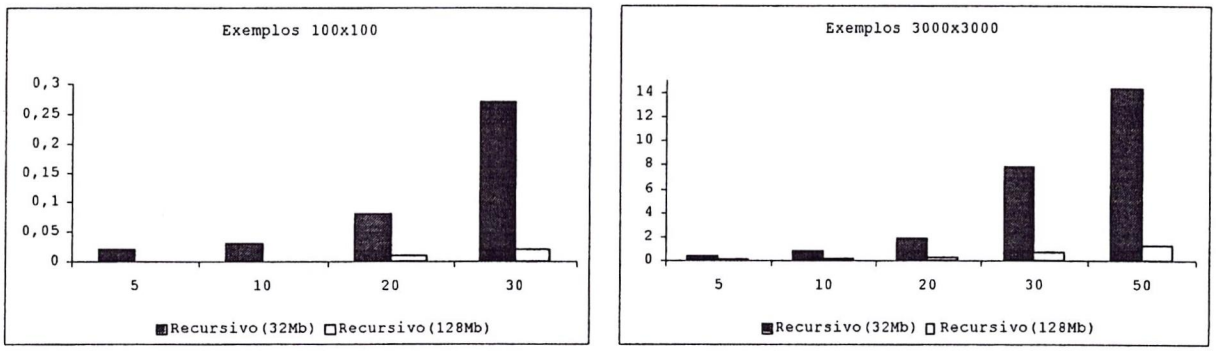

Gráfico 5.2 - Discretização nos exemplos grandes

A seguir, nas Tabelas 5.3 e 5.4, os conjuntos foram novamente gerados, considerando a regra de simetria, apresentada na seção 3.2.1. O tamanho dos conjuntos diminui visivelmente, assim como o tempo computacional, resultando na diminuição do espaço de busca do problema.

Tabela 5.3 - Discretização nos exemplos pequenos(com simetria)

Qte	M	L	W	|X| (DP)	|Y| (DP)	T(s) (1)	T(s) (2)	T(s) (3)
100	5	100	100	2.80(0.75)	2.50(1.12)	0.03 0.01	0.02 0.00	0.00 0.00
100	10	100	100	5.40(1.69)	6.20(1.17)	0.05 0.01	0.03 0.00	0.00 0.00
100	20	100	100	11.00(1.26)	10.20(1.66)	0.17 0.03	0.08 0.01	0.00 0.00
100	30	100	100	16.60(1.62)	16.10(1.76)	0.55 0.04	0.27 0.02	0.00 0.00
100	5	1000	1000	2.90(1.37)	2.20(1.40)	0.24 0.05	0.13 0.03	0.00 0.00

100	10	1000	1000	5.00(1.26)	5.70(1.35)	0.51 0.11	0.25 0.05	0.01 0.00
100	20	1000	1000	10.40(2.46)	9.70(2.19)	1.37 0.25	0.64 0.13	0.00 0.00
100	30	1000	1000	16.00(1.43)	17.50(1.55)	2.88 0.41	2.72 0.24	0.03 0.00
100	50	1000	1000	23.00(2.83)	26.00(3.16)	11.05 0.82	4.76 0.46	0.01 0.00
50	5	3000	3000	2.00(1.41)	1.60(1.36)	0.68 0.16	0.35 0.09	0.00 0.00
50	10	3000	3000	4.80(2.23)	5.40(2.42)	1.19 0.35	0.80 0.15	0.01 0.00
50	20	3000	3000	10.80(1.17)	9.40(2.24)	3.98 0.72	1.87 0.34	0.00 0.00
50	30	3000	3000	15.60(2.42)	13.80(1.72)	16.12 1.20	7.82 0.67	0.00 0.00
50	50	3000	3000	25.00(1.90)	24.60(3.01)	29.63 2.40	14.31 1.32	0.02 0.02
50	50	5000	5000	25.80(2.79)	26.20(3.87)	51.60 4.28	24.48 2.34	0.03 0.02

Tabela 5.4 - Discretização nos exemplos grandes (com simetria)

Qte	m	L	W	|X| (DP)	|Y| (DP)	T(s) (1)	T(s) (2)	T(s) (3)
100	5	100	100	12.20(3.66)	14.10(7.49)	0.04 0.01	0.03 0.01	0.00 0.00
100	10	100	100	24.00(3.87)	26.10(6.17)	0.19 0.04	0.09 0.02	0.00 0.00
100	20	100	100	33.70(3.61)	34.20(2.23)	0.86 0.14	0.36 0.06	0.01 0.00
100	30	100	100	38.20(1.17)	38.50(1.28)	2.09 0.33	1.04 0.16	0.00 0.00
100	5	1000	1000	10.90(5.84)	15.10(9.37)	0.35 0.15	0.29 0.07	0.00 0.00
100	10	1000	1000	37.90(20.41)	29.00(7.46)	1.48 0.39	0.68 0.18	0.01 0.00
100	20	1000	1000	84.10(31.19)	104.10(46.35)	6.23 1.32	3.06 0.60	0.01 0.00
100	30	1000	1000	157.00(17.00)	166.00(23.00)	19.01 3.07	8.96 1.46	0.03 0.02

100	50	1000	1000	235.80(29.92)	256.40(11.72)	79.46	38.94	0.02
						10.95	5.32	0.01
50	5	3000	3000	16.80(8.30)	15.00(8.49)	2.19	1.01	0.01
						0.54	0.24	0.01
50	10	3000	3000	28.40(1.36)	33.60(14.01)	3.63	1.70	0.01
						0.99	0.46	0.00
50	20	3000	3000	121.60(32.78)	133.00(53.56)	20.04	8.73	0.02
						4.19	1.90	0.01
50	30	3000	3000	340.80(113.39)	301.20(77.90)	63.50	30.78	0.04
						13.58	6.33	0.01
50	50	3000	3000	577.00(55.13)	405.00(66.84)	158.14	123.41	0.07
						33.18	16.14	0.02
50	50	5000	5000	558.00(130.33)	517.00(148.21)	1045.16	516.74	0.10
						40.49	20.16	0.03

Entretanto, nos exemplos, quando L>100 (comprimento) ou W>100 (largura), o tamanho do conjunto de discretização é muito grande, o que, às vezes, inviabiliza o processo de busca. Note nestas tabelas que com o uso da simetria, o tempo computacional utilizando a técnica de recursividade diminuiu bastante, sendo nos problemas pequenos, muitas vezes próximos dos tempos obtidos com o processo iterativo.

Utilizando a heurística H3, apresentada na seção 3.3, que limita o tamanho dos conjuntos de discretização, obtêm-se os resultados apresentados nas Tabelas 5.5 e 5.6, com $|X| < 100$ e $|Y| < 100$.

Tabela 5.5 - Discretização nos exemplos pequenos com $|X| < 100$ e $|Y| < 100$ (Heurística H3)

Qte	M	L	W	$\|X\|$ (DP)	$\|Y\|$ (DP)	T(s) (1)	T(s) (2)	T(s) (3)
100	20	1000	1000	89.10(4.23)	87.40(5.78)	13.02	12.74	0.03
						3.37	3.52	0.01
100	30	1000	1000	88.50(1.50)	97.00(1.00)	158.87	82.30	0.06
						7.96	14.60	0.01
100	50	1000	1000	93.40(4.20)	94.60(5.18)	304.12	419.75	0.04
						26.73	62.33	0.02

50	20	3000	3000	89.60(6.53)	86.80(10.54)	48.28 9.63	34.65 9.37	0.08 0.02
50	30	3000	3000	92.30(3.74)	90.20(6.14)	147.06 21.75	206.08 39.03	0.11 0.02
50	50	3000	3000	91.80(3.94)	93.50(3.61)	(*) 68.65	(*) 189.46	0.17 0.06
50	50	5000	5000	92.80(4.83)	92.10(5.63)	2040.01 132.35	4265.03 296.12	3.06 0.09

Tabela 5.6 - Discretização nos exemplos grandes com $|X| < 100$ e $|Y| < 100$ (Heurística H3)

Qte	m	L	W	$\|X\|$ (DP)	$\|Y\|$ (DP)	T(s) (1)	T(s) (2)	T(s) (3)
100	10	1000	1000	83.10(12.59)	71.30(12.02)	87.06 15.64	16.70 3.76	0.02 0.01
100	20	1000	1000	85.20(7.97)	88.30(8.23)	1808.47 171.03	127.99 14.54	0.03 0.01
100	30	1000	1000	89.00(1.00)	94.50(4.50)	(*) 502.22	484.77 39.17	0.02 0.01
100	50	1000	1000	91.00(1.63)	94.33(0.94)	(*) 613.23	(*) 97.12	0.09 0.01
50	5	3000	3000	50.00(15.34)	58.60(23.09)	77.45 16.40	19.42 5.05	0.02 0.02
50	10	3000	3000	80.00(13.48)	84.80(12.70)	109.60 31.62	36.43 9.21	0.05 0.01
50	20	3000	3000	88.67(11.09)	89.00(8.83)	2603.17 273.64	319.26 40.37	0.09 0.03
50	30	3000	3000	93.67(4.50)	93.00(5.35)	(*) 1092.23	10391.1 234.37	0.08 0.03
50	50	3000	3000	92.50(1.50)	96.00(2.00)	(*) 1590.12	(*) 407.22	0.08 0.03
50	50	5000	5000	94.80(5.21)	92.00(4.67)	(*) 2011.34	(*) 576.33	0.13 0.04

O tempo computacional com o uso da heurística de Beasley cresce um pouco com o uso da técnica iterativa, mas também com a técnica recursiva, já que o processo repete-se várias vezes até que o tamanho do conjunto de discretização seja menor que o desejado. Com isto, quando compara-se os tem-

pos computacionais de ambas as técnicas, descarta-se mesmo a recursiva (Gráficos 5.3 e 5.4).

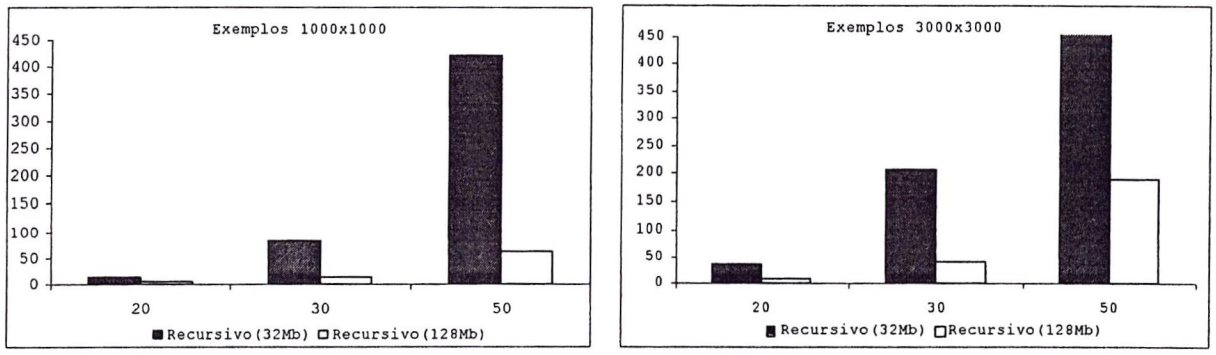

Gráfico 5.3 - Discretização nos exemplos pequenos com $|X| < 100$ e $|Y| < 100$ (Heurística H3)

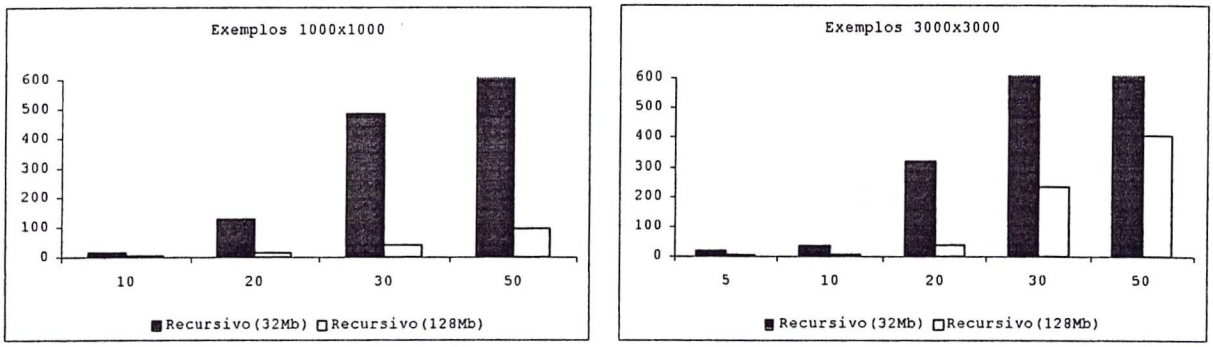

Gráfico 5.4 - Discretização nos exemplos grandes com $|X| < 100$ e $|Y| < 100$ (Heurística H3)

5.2. Problema Irrestrito

Nesta seção são apresentados os resultados obtidos com o problema de corte irrestrito.

Foram feitos testes com diferentes tipos de conjuntos de discretização. Aqui, são apresentados os dados obtidos com o problema irrestrito não-estagiado, percorrendo os conjuntos de discretização de maneira ordenada crescente e decrescente e, também aleatória. Na geração dos conjuntos utilizou-se a regra de simetria e profundidade máxima de 4 na estratégia de busca.

- **Cortes ordenados crescente**

A seguir, são apresentados os resultados obtidos percorrendo os conjuntos de cortes de maneira ordenada crescentemente, os quais foram limitados em 100 elementos.

Tabela 5.7 - Exemplos pequenos

Qte	M	L	W	Solução	Nº-nós	T(s)
50	5	100	100	0.8776	8.00	0.01
50	10	100	100	0.9362	20.30	0.01
50	20	100	100	0.9792	52.52	0.05
50	5	1000	1000	0.8583	7.14	0.00
50	10	1000	1000	0.9383	28.00	0.02
50	20	1000	1000	0.9530	70.72	0.06
50	30	1000	1000	0.9690	209.16	0.21
50	5	3000	3000	0.9240	8.10	0.00
50	10	3000	3000	0.9382	18.18	0.02
50	20	3000	3000	0.9565	70.54	0.08
50	30	3000	3000	0.9651	194.54	0.28

Tabela 5.8 - Exemplos grandes

Qte	M	L	W	Solução	Nº-nós	T(s)
50	5	100	100	0.9661	35.00	0.03
50	10	100	100	0.9893	350.60	0.40
50	20	100	100	0.9964	83.00	0.37
50	5	1000	1000	0.9758	15.80	0.02
50	10	1000	1000	0.9914	254.60	0.58
50	20	1000	1000	0.9926	309.87	1.16
50	30	1000	1000	0.9908	457.32	3.30
50	5	3000	3000	0.9832	34.20	0.06
50	10	3000	3000	0.9958	361.80	0.51
50	20	3000	3000	0.9932	595.40	3.51
50	30	3000	3000	0.9445	895.12	12.03

- **Cortes ordenados decrescente**

A seguir, são apresentados os resultados obtidos

percorrendo os conjuntos de cortes de maneira ordenada decrescentemente, os quais foram limitados em 100 elementos.

Tabela 5.9 - Exemplos pequenos

Qte	m	L	W	Solução	Nº-nós	T(s)
50	5	100	100	0.8775	9.01	0.02
50	10	100	100	0.9360	18.23	0.02
50	20	100	100	0.9783	44.66	0.04
50	5	1000	1000	0.8498	6.55	0.01
50	10	1000	1000	0.9380	22.70	0.03
50	20	1000	1000	0.9529	69.96	0.07
50	30	1000	1000	0.9687	165.03	0.20
50	5	3000	3000	0.9234	5.98	0.01
50	10	3000	3000	0.9381	15.11	0.03
50	20	3000	3000	0.9559	59.01	0.01
50	30	3000	3000	0.9649	136.01	0.24

Tabela 5.10 - Exemplos grandes

Qte	M	L	W	Solução	Nº-nós	T(s)
50	5	100	100	0.9660	36.20	0.03
50	10	100	100	0.9890	366.10	0.47
50	20	100	100	0.9964	130.12	0.46
50	5	1000	1000	0.9758	15.96	0.02
50	10	1000	1000	0.9912	301.01	0.61
50	20	1000	1000	0.9926	371.22	1.22
50	30	1000	1000	0.9908	484.83	3.20
50	5	3000	3000	0.9832	37.27	0.05
50	10	3000	3000	0.9961	390.12	0.55
50	20	3000	3000	0.9932	621.87	3.13
50	30	3000	3000	0.9438	900.01	16.10

- **Cortes aleatórios**

Neste caso, os conjuntos de discretização são percorridos de maneira aleatória (heurística H5 da seção 3.3). As Tabelas 5.11 e 5.12 apresentam os mesmos exemplos utilizados na geração com cortes ordenados.

Tabela 5.11 - Exemplos pequenos

Qte	m	L	W	Solução	Nº-nós	T(s)
50	5	100	100	0.8776	8.40	0.01
50	10	100	100	0.9360	20.88	0.01
50	20	100	100	0.9789	57.64	0.09
50	5	1000	1000	0.8580	7.44	0.01
50	10	1000	1000	0.9383	30.10	0.03
50	20	1000	1000	0.9530	78.14	0.13
50	30	1000	1000	0.9688	427.01	0.44
50	5	3000	3000	0.9238	8.68	0.00
50	10	3000	3000	0.9371	19.80	0.03
50	20	3000	3000	0.9562	177.54	0.12
50	30	3000	3000	0.9650	506.98	1.01

Tabela 5.12 - Exemplos grandes

Qte	m	L	W	Solução	Nº-nós	T(s)
50	5	100	100	0.9660	43.89	0.07
50	10	100	100	0.9893	525.76	0.88
50	20	100	100	0.9963	173.40	0.75
50	5	1000	1000	0.9758	17.80	0.01
50	10	1000	1000	0.9914	450.60	1.19
50	20	1000	1000	0.9926	245.17	1.02
50	30	1000	1000	0.9908	187.69	4.56
50	5	3000	3000	0.9832	77.80	0.10
50	10	3000	3000	0.9960	593.03	1.41
50	20	3000	3000	0.9929	347.49	4.23
50	30	3000	3000	0.9439	885.38	6.81

Em geral, os resultados obtidos percorrendo os conjuntos de discretização de maneira ordenada, seja de maneira crescente ou decrescente, são melhores que os obtidos usando aleatoriedade; apenas em alguns casos isolados o percurso de maneira aleatória foi melhor.

6. Conclusões

O objetivo principal deste relatório técnico é mostrar algumas técnicas e estrutura de dados para a implementação computacional do problema de corte.

A princípio, imaginava-se que implementando o conjunto de discretização de maneira recursiva seria mais proveitoso; entretanto, constatou-se que os tempos computacionais obtidos são muito ruins comparados com os tempos obtidos com o processo iterativo. Isto, a princípio, foi atribuído a velocidade do processador e a quantidade de memória RAM disponível nos primeiros testes. Os testes foram novamente refeitos com um processador mais veloz e, uma quantidade de memória também maior, que na prática não encontramos sempre disponível com o usuário do produto final do problema de corte, entretanto, mesmo assim, observa-se que o tempo computacional é grande quando comparado com o tempo obtido de maneira iterativa. Portanto, pode-se afirmar que a utilização da recursividade para a implementação do conjunto de discretização não é um bom exemplo de uso desta técnica.

Quanto as estruturas de dados utilizadas na abordagem em Grafo E/OU para a resolução do problema de corte, independente de sua dimensão, pois como foi visto, isto altera apenas um ou mais campos da sua estrutura, é possível desenvolvê-la de modo que permita incluir novas características ao problema, seja elas por limitação de equipamentos de corte ou

propriedades do material, sem que seja necessário fazer muitas mudanças na própria estrutura de dados.

7. Referências Bibliográficas

ARENALES, M. N. *Uma Teoria para o Problema de Corte.* São Carlos: USP, 1993. Tese (Livre-docência) - Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, 1993.

ARENALES, M. N., MORÁBITO, R. An AND/OR-graph approach to the solution of two-dimensional non-guillotine cutting problems. *European Journal of Operational Research.* v.84, p.599-617, 1995.

BEASLEY, J. Algorithms for unconstrained two dimensional guillotine cutting. *Journal of the Operational Research Society.* v.36, p.297-306, 1985.

CHRISTOFIDES, N., WHITLOCK, C. An algorithm for two-dimensional cutting problems. *Operations Research.* v.25, p.30-44, 1977.

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

GILMORE, P., GOMORY, R. A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research.* v.9, p.849-859,

1961.

GILMORE, P., GOMORY, R. A Linear Programming Approach to the Cutting Stock Problem - Part II. *Operations Research*. v.11, p.863-888, 1963.

GILMORE, P., GOMORY, R. MultiStage cutting stock problems of two and more dimensions. *Operations Research*. v.14, p.1045-1074, 1965.

HERZ, J. Recursive Computational Procedure for Two Dimensional Stock Cutting. *IBM Journal of Research and Development*. v.16, p.462-469, 1972.

HIFI, M. The DH/KD algorithm: a hybrid approach for unconstrained two-dimensional cutting problems. *European Journal of Operational Research*. v.97, p.41-52, 1997.

HINXMAN, A. The Trim-Loss and Assortment Problems: A Survey. *European Journal of Operational Research*. v.5, p.8-18, 1980.

MORÁBITO, R. *Corte de Estoque Bidimensional*. São Carlos: USP, 1989. Dissertação (Mestrado) - Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, 1989.

MORÁBITO, R. *Uma Abordagem em Grafo E/OU para o Problema do Empacotamento: Aplicação ao Carregamento de Paletes e Contêineres*. São Carlos: USP, 1992. Tese (Doutorado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, 1992.

MORÁBITO, R., ARENALES, M. N. An AND/OR-graph approach to the container loading problem. *International Transactions in Operational Research*. v.1, p.59-73, 1994.

MORÁBITO, R., ARENALES, M. N. Staged and constrained two-dimensional guillotine cutting problems: An AND/OR-graph approach. *European Journal of Operational Research*. v.94, p.548-560, 1996.

OLIVEIRA, J., FERREIRA, J. An improved version of Wang's algorithm for two-dimensional cutting problems. *European Journal of Operational Research*. v.44, p.256-266, 1990.

VELOSO, P, SANTOS, C., AZEVEDO, P., FURTADO, A *Estruturas de Dados*. Rio de Janeiro: Campus, 1984.

VILLAS, M. V., FERREIRA, A. G. M., LEROY, P. G., MIRANDA, C., BOCKMAN, C. L. *Estruturas de dados: conceitos e técnicas de implementação*. Rio de Janeiro: Campus, 1993. 298p.

WANG, P. Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*. v.31, p.573-587, 1983.