O PROBLEMA DA MOCHILA COMPARTIMENTADA
APLICADO NO CORTE DE BOBINAS DE AÇO

Robinson Samuel Vieira Hoto

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Nelson Maculan Filho, D. Hábil.

Prof. Marcos Nereu Arenales, D.Sc.

Prof. Claudio Thomas Bornstein, Dr. Rer. Nat.

Prof. Nei Yoshihiro Soma, Ph.D.

Prof. Luiz Satoru Ochi, D.Sc.

Profª Maria Helena Cautiero Horta Jardim, D.Sc.

RIO DE JANEIRO, RJ, BRASIL
DEZEMBRO DE 2001

HOTO, ROBINSON SAMUEL VIEIRA

O Problema da Mochila Compartimentada Aplicado no Corte de Bobinas de Aço [Rio de Janeiro] 2001

IX, 188 p., 29,7 cm (COPPE / UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2001)

Tese – Universidade do Rio de Janeiro, COPPE

1. Problema de Corte e Empacotamento

I. COPE / UFRJ II. Título (série)

Dedico este trabalho a meu saudoso amigo Marcos Cassiolato.

Meus agradecimentos serão breves, mas sinceros. Inicialmente quero destacar a seriedade profissional e sobretudo o lado humano e alegre do Professor Nelson Maculan, com quem tive o privilégio de muito aprender e firmar amizade. O Professor Marcos Arenales, com quem já havia tido o prazer de trabalhar, demonstrou mais uma vez ter paciência e dedicação, tivemos produtivas conversas.

Não posso deixar de citar a colaboração do Professor Nei Soma que muito gentilmente me recebeu e forneceu informações valorosas para a finalização deste trabalho.

A cooperação do Professor Arenales permitiu a troca de idéias com Fabiano Marques, com quem firmei amizade, e que muito colaborou quando precisei.

O apoio dos colegas da COPPE e da UEL, além das pessoas presentes nos momentos atribulados, foi muito importante do ponto de vista humano, agradeço a todos.

O incentivo das pessoas mais íntimas, sobretudo da família.

Por fim, agradeço aos membros da banca examinadora por dedicarem seu tempo na leitura e avaliação deste texto.

Este trabalho teve o apoio financeiro da Universidade Estadual de Londrina (UEL) e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

Resumo da Tese apresentada a COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

O PROBLEMA DA MOCHILA COMPARTIMENTADA APLICADO NO CORTE DE BOBINAS DE AÇO

Robinson Samuel Vieira Hoto

Dezembro/2001

Orientadores: Nelson Maculan Filho
Marcos Nereu Arenales

Programa: Engenharia de Sistemas e Computação

Neste trabalho abordamos um Problema de Corte e Empacotamento (Problema de Corte de Bobinas de Aço), onde os padrões de corte devem ser estruturados em compartimentos. Utilizamos a Técnica de Geração de Colunas de Gilmore-Gomory para resolver o problema, de modo que, para gerar os padrões compartimentados definimos uma nova modalidade de mochila que denominamos Mochila Compartimentada. Descrevemos um método de resolução para esta mochila e propomos um procedimento heurístico. Tendo em vista a aplicação prática, desenvolvemos o aplicativo RollCut que auxilia na programação de cortes de bobinas de aço.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

THE COMPARTMENTED KNAPSACK PROBLEM APPLIED IN THE CUT OF BOBBINS OF STEEL

Robinson Samuel Vieira Hoto

December/2001

Advisors: Nelson Maculan Filho
Marcos Nereu Arenales

Department: Engineering of Systems and Computation

This work approaches a Cutting and Packing Problem (Cut of Steel Bobbins Problem), where the cut patterns must be structuralized in compartments. We use the Technique of Columns Generation of Gilmore-Gomory to resolve the problem and we define a new modality of knapsack that, Compartmented Knapsack, used to generate the compartmented patterns. We describe a method of resolution for this knapsack and consider a heuristic. In view of the practical application, we develop the applicatory RollCut that assists in the programming of cuts of steel bobbins.

Índice

Introdução 1

Capítulo 1 Teoria Básica dos Problemas de Corte e Empacotamento 7

Problemas Básicos 7

Restrições Adicionais 8

Padrões Compartimentados 9

Seleção de Objetos 10

Seleção de Itens 11

O Problema de Corte de Estoque 11

Uma estratégia de solução do Corte de Estoque 13

Solução Inteira e Arredondamento 14

O Método *Branch-and-price* 16

Corte de Estoque com vários objetos em quantidades ilimitadas 17

Corte de Estoque com vários objetos em quantidades limitadas 19

Textos importantes sobre Corte e Empacotamento 20

O caso Bidimensional 23

Bin-packing (heurísticas NF, NFD, FF, FFD, BF e BFD) 26

Capítulo 2 PCBA - Problema de Corte de Bobinas de Aço 31

Bobinas, Fitas e Tubos 31

Bobinas Intermediárias 32

Laminação a Frio 35

Revisão da Literatura sobre o PCBA 36

Heurística de Ferreira-Neves-Castro 37

Heurística de Hoto-Arenales 38

A abordagem de Carvalho-Rodrigues 40

A abordagem 1.5-dimensional de Hoto 43

Uma nova modelagem para resolver o PCBA 45

Agrupamentos das Fitas 46

Heurística de Arredondamento para o PCBA 47

Capítulo 3 Problemas Clássicos da Mochila 49

Apresentação do Problema da Mochila 49

Mochila 0-1 50

Mochila Restrita e Irrestrita 52

Soma de Subconjuntos, Designação e Múltipla Escolha 54

Múltiplas Mochilas e Mochilas com Coeficientes Variáveis 56

Mochila Encapsulada 57

O Problema do Troco e o *Bin-packing* na linguagem de mochilas 59

Capítulo 4 PMC - Problema da Mochila Compartimentada 62

O que é um compartimento? 63

Compartimentos Construtivos 64

Compartimentos Dominantes 64

Padrões de Corte Compartimentados 65

Formalização Matemática do PMC (Agrupamentos e Compartimentos) 66

O gerador de padrões de Carvalho-Rodrigues é um caso particular do PMC 68

Decomposição e Resolução do PMC 69

Seleção de Compartimentos Construtivos 70

Primeira etapa da Decomposição do PMC 70

Segunda etapa da Decomposição do PMC 71

O Algoritmo COMPEX 72

Uma Heurística de Compartimentação 73

O Algoritmo COMPMT 74

Resultados Computacionais pelo COMPEX e COMPMT 76

Compartimentação Binária (PMC 0-1) 79

Um caso particular do PMC 0-1 79

Soluções Sensíveis no PMC 0-1 80

Justaposição e Pseudointersecção de Agrupamentos 81

Um Método *Branch-and-bound* para o PMC 0-1 82, 86

Limitante Superior para o PMC 0-1 84

Propriedades dos Limitantes Superiores do PMC 0-1 85

Um exemplo do PMC 0-1 87

Compartimentação Restrita (PMC restrito) 88

O Algoritmo COMPREST (PMC restrito) 90

Heurística de Decomposição de Marques-Arenales (PMC restrito) 92

Heurística do Melhor Compartimento de Marques-Arenales (PMC restrito) 93

Desempenho do COMPEX, COMPMT e Heurística de Decomposição 95

Capítulo 5 O PMC aplicado no PCBA 97
O Modelo Matemático do PCBA 97
Estrutura Matricial do Modelo do PCBA 99
Custo Relativo de uma coluna no PCBA 103
Solução Básica Inicial do PCBA 105
Problema Artificial do PCBA (Fase 1 do Simplex) 107
A Matriz Inversa da corrente Base 108
Arredondamento da Solução 108
O Problema Residual 109
O Gerador de Padrões Restritos 110
Fracionamento das Bobinas do Estoque 110
Resultados Computacionais 112

Capítulo 6 Conclusões e Investigações Futuras 117

Apêndice A Tipologia dos Problemas de Corte e Empacotamento 121
Estrutura dos Problemas 122
Dimensionalidade 122
Tipos de Seleção de Objetos e Itens 124
Tipos de Medida 124
Aparência 124
Sortimento 125
Disponibilidade 125
Restrições sob os Padrões 125
Objetivos 126
Status e Variabilidade dos Dados 126
Métodos de Solução 126

Apêndice B Método do Simplex, Decomposição de Dantzig-Wolfe, Geração de Colunas de Gilmore-Gomory 121
Método do Simplex 121
Decomposição de Dantzig-Wolfe 138
Geração de Colunas de Gilmore-Gomory 141

Apêndice C Classes Algorítmicas **146**
Problemas de Decisão 146
Problemas de Localização 147
Problemas de Otimização 147
Algoritmos Não Determinísticos 148
A questão de Cook 148
Problema da SAT 149
A resposta de Cook 149
Problemas NP-árduos e NP-completos 149

Apêndice D Interface do Aplicativo RollCut **151**
A Interface do Aplicativo 151
Fluxo de Dados entre os Procedimentos principais do RollCut 156

Apêndice E Pseudocódigos de alguns Problemas da Mochila **156**
Algoritmo Guloso de Martello-Toth para Mochila 0-1 156
Algoritmo Guloso de Martello-Toth para Mochila Restrita 156
Algoritmo de Gilmore-Gomory para Mochila Irrestrita 157
Algoritmo de Martello-Toth para Mochila Irrestrita 158
Algoritmo de Yanasse-Soma para Mochila Irrestrita (igualdade) 161
Algoritmo de Yanasse-Soma para Mochila Irrestrita (desigualdade) 162
Algoritmo de Yanasse-Soma para Mochila Restrita (desigualdade) 164

Bibliografia **166**

Capítulo 1

Teoria Básica dos Problemas de Corte e Empacotamento

Neste capítulo apresentaremos os problemas mais básicos de corte e empacotamento, portanto, daremos ênfase ao caso unidimensional, veja apêndice A sobre classificação segundo a dimensão. Ao mesmo tempo, comentaremos alguns trabalhos importantes relacionados ao assunto.

1 – Problemas de uma Única Mochila

Suponha que um **objeto** (barra, bobina, etc) deva ser cortado ao longo de sua largura em **itens** (pedaços) de larguras especificadas. Cada um destes itens possui um valor associado que denominamos "valor de utilidade". Surge um primeiro problema de otimização combinatória: maximizar o valor total de utilidade associado ao corte de itens num objeto.

Este problema de corte, embora seja bem simples, é fundamental na resolução de problemas de corte mais elaborados, e também pode sofrer a adição de condições que o torna mais difícil de ser tratado, como ocorre no estudo do Problema de Corte de Bobinas de Aço.

Observe que o problema é unidimensional (apenas uma dimensão é relevante no processo de corte) e pode ser modelado como um problema de otimização linear inteiro, mais precisamente como um Problema da Mochila que estudaremos com maiores detalhes no capítulo 4.

$$\text{maximizar} \quad \sum_{i=1}^{n} u_i a_i \qquad (1.1)$$

sujeito a:

$$\sum_{i=1}^{n} \ell_i a_i \leq L \qquad (1.2)$$

$$a_i \geq 0 \text{ e inteiro}, i = 1, \ldots, n \qquad (1.3)$$

No modelo anterior n é o número de itens, u_i é a utilidade do item i, ℓ_i é a largura do item i, L é a largura de uma barra e a_i é o número de itens tipo i que aparecem numa barra.

Condições adicionais podem surgir, por exemplo, a quantidade de itens deve ser limitada, digamos por b_i, $i = 1, \ldots, n$. Neste caso o modelo anterior toma o aspecto do Problema da Mochila Restrito.

$$\text{maximizar} \quad \sum_{i=1}^{n} u_i a_i \qquad (2.1)$$

sujeito a:

$$\sum_{i=1}^{n} \ell_i a_i \leq L \qquad (2.2)$$

$$0 \leq a_i \leq b_i \text{ e inteiro}, \; i = 1, \ldots, n \qquad (2.3)$$

Podemos ainda ter a situação em que apenas um único exemplar de cada item pode ser cortado, isto é, o Problema da Mochila 0-1.

$$\text{maximizar} \quad \sum_{i=1}^{n} u_i a_i \qquad (3.1)$$

sujeito a:

$$\sum_{i=1}^{n} \ell_i a_i \leq L \qquad (3.2)$$

$$a_i = 0 \text{ ou } 1, \; i = 1, \ldots, n \qquad (3.3)$$

Uma outra situação ocorre quando o número de cortes numa barra deve ser limitado por um valor F. Este tipo de restrição aparece freqüentemente no corte de bobinas de papel ou de aço e o modelo têm o seguinte aspecto:

$$\text{maximizar} \quad \sum_{i=1}^{n} u_i a_i \qquad (4.1)$$

sujeito a:

$$\sum_{i=1}^{n} \ell_i a_i \leq L \qquad (4.2)$$

$$\sum_{i=1}^{n} a_i \leq F \qquad (4.3)$$

$$a_i \geq 0 \text{ e inteiro}, \; i = 1, \ldots, n \qquad (4.4)$$

Outras condições podem ainda surgir, como construir numa mochila compartimentos de tamanhos variáveis, porém limitados inferiormente e superiormente, que devem acomodar itens compatíveis entre si. Por exemplo, digamos que itens de índices pares não podem ser combinados com itens de índices ímpares. Na figura 1 temos uma ilustração desta situação no corte de uma bobina de aço, onde $L_{min} \leq L_1 \leq L_{max}$, $L_{min} \leq L_2 \leq L_{max}$, $L_{min} \leq L_3 \leq L_{max}$ e $L_1 + L_2 + L_3 \leq L_{max}$.

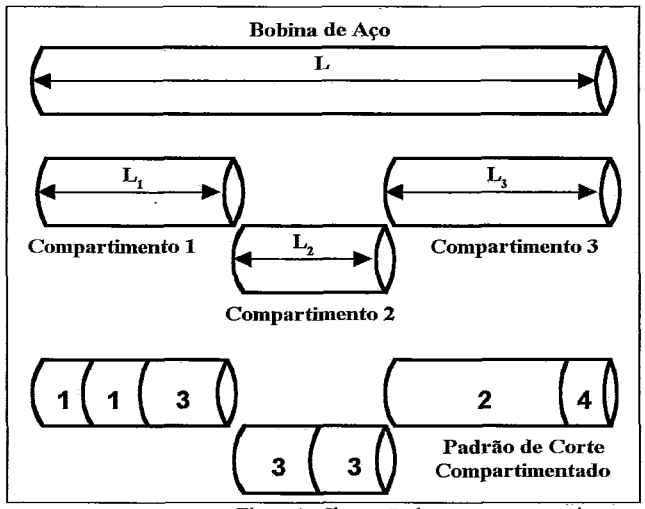

Figura 1 – Ilustração de corte em compartimentos.

Um "Padrão Compartimentado" como o da figura 1, pode ser definido por um problema que denominamos "Problema da Mochila Compartimentada". Esta é uma nova classe de problemas da mochila que será mais bem definida no capítulo 5.

2 - Problemas de Múltiplas Mochilas

Suponha que no problema de corte anterior, várias barras estejam disponíveis para serem cortadas, visando à produção dos vários itens. Dois problemas podem ser identificados: no primeiro as barras são suficientes para a produção de todos os itens, e no segundo as barras são insuficientes.

Para resolver o primeiro problema precisamos determinar quais barras devem ser cortadas, ou seja, necessitamos "selecionar objetos", já no segundo todas as barras serão utilizadas, assim devemos "selecionar itens" a serem produzidos.

2.1 - Seleção de Objetos (todos os itens são produzidos)

Neste problema iremos atribuir custos aos objetos, de modo que, o objetivo será o de minimizar o custo total dos objetos envolvidos na produção dos itens.

Dados do problema:

n: número total de itens;

m: número total de objetos (barras);

ℓ_i: largura do item i, $i = 1, \ldots, n$;

L_j: largura do objeto (barra) j, $j = 1, \ldots, m$;

c_j: custo do objeto (barra) j, $j = 1, \ldots, m$;

Variáveis do problema:

$$y_j = \begin{cases} 1, \text{se a barra j for cortada;} \\ 0, \text{caso contrário.} \end{cases}$$

$$x_{ij} = \begin{cases} 1, \text{se o item i for cortado da barra j;} \\ 0, \text{caso contrário.} \end{cases}$$

A Seleção de Barras é feita segundo o modelo:

$$\text{minimizar} \quad \sum_{j=1}^{m} c_j y_j \qquad (5.1)$$

sujeito a:

$$\sum_{i=1}^{n} \ell_i x_{ij} \leq L_j y_j \ , \quad j = 1, \ \ldots, \ m \qquad (5.2)$$

$$\sum_{j=1}^{m} x_{ij} = 1, \ i = 1, \ldots, n \qquad (5.3)$$

$$x_{ij} = 0 \text{ ou } 1, \ y_j = 0 \text{ ou } 1, \ i = 1, \ldots, n, \ j = 1, \ldots, m \qquad (5.4)$$

Se por ventura existir uma limitação d_i, $i = 1, \ldots, n$, sob os itens, as restrições (5.3) e (5.4) devem ser alteradas para:

$$\sum_{j=1}^{m} x_{ij} = d_i, \ i = 1, \ldots, n \qquad (5.5)$$

$$x_{ij} \geq 0 \text{ e inteiro}, \ y_j = 0 \text{ ou } 1, \ i = 1, \ldots, n, \ j = 1, \ldots, m \qquad (5.6)$$

2.2 - Seleção de Itens (todos os objetos serão cortados)

Em analogia aos Problemas da Mochila atribuiremos valores de utilidade u_i para os itens $i = 1, \ldots, n$. Observe que nem todos os itens serão produzidos, uma vez que, as barras não são suficientes. Assim, o objetivo será o de maximizar o valor total de utilidade.

A Seleção de Itens é feita segundo o modelo:

$$\text{maximizar} \quad \sum_{j=1}^{m}\sum_{i=1}^{n} u_i x_{ij} \qquad (6.1)$$

sujeito a:

$$\sum_{i=1}^{n} \ell_i x_{ij} \leq L_j, \; j = 1, \ldots, m \qquad (6.2)$$

$$\sum_{j=1}^{m} x_{ij} \leq 1, \; i = 1, \ldots, n \qquad (6.3)$$

$$x_{ij} = 0 \text{ ou } 1, \; i = 1, \ldots, n, \; j = 1, \ldots, m \qquad (6.4)$$

Como no caso anterior, havendo uma limitação d_i, $i = 1, \ldots, n$, sob os itens, as restrições (6.3) e (6.4) devem ser alteradas para:

$$\sum_{j=1}^{m} x_{ij} \leq d_i, \; i = 1, \ldots, n \qquad (6.5)$$

$$x_{ij} \geq 0 \text{ e inteiro}, i = 1, \ldots, n, \; j = 1, \ldots, m \qquad (6.6)$$

Vale destacar que se o limite sob cada item for arbitrariamente grande o problema se reduz a m Problemas da Mochila.

3 - O Problema de Corte de Estoque

Conhecido na literatura como *Cutting-Stock Problem*, e semelhante ao caso da Seleção de Objetos, onde todos os itens serão produzidos, o Corte de Estoque entretanto, é diferenciado pela seguinte característica: existem poucos tipos de itens a serem produzidos, porém, a quantidade a ser produzida é alta. Em outras palavras, a variabilidade é baixa, mas, a demanda é alta.

Os objetos em estoque podem estar disponíveis em grande quantidade ou num número limitado, além disso, pode haver mais de um tipo de objeto em estoque. Inicialmente apresentaremos os dados associados aos itens, pois, estes serão comuns a todos os casos que apresentaremos.

Dados dos itens:

n: número total de itens;

ℓ_i: largura do item i, $i = 1, \ldots, n$;

d_i: demanda do item i, $i = 1, \ldots, n$;

Podemos pensar em modelar o Corte de Estoque em duas etapas:

1) defina todas as possíveis maneiras de cortar os objetos em estoque, ou seja, defina todos os padrões de corte;
2) decida quantas vezes cada padrão de corte será utilizado para atender as demandas.

Note que na primeira etapa temos um problema essencialmente combinatório, já na segunda o problema é puramente contábil. A cada padrão de corte j iremos associar um vetor n-dimensional $a_j = (a_{1j}, \ldots, a_{nj})^T$, cujas componentes contabilizam o número de itens a serem cortados segundo o padrão. Para que fique mais claro considere o seguinte exemplo, onde a largura de um objeto será $L = 120$:

Exemplo:

3: número total de itens;

$\ell_1 = 30$, $\ell_2 = 42$ e $\ell_3 = 45$: larguras dos itens;

$d_1 = 1000$, $d_2 = 1250$ e $d_3 = 2000$: demandas dos itens;

Na figura 2 são mostrados quatro padrões, sendo que os três primeiros são denominados "Padrões Homogêneos", que são aqueles que produzem apenas um tipo de item.

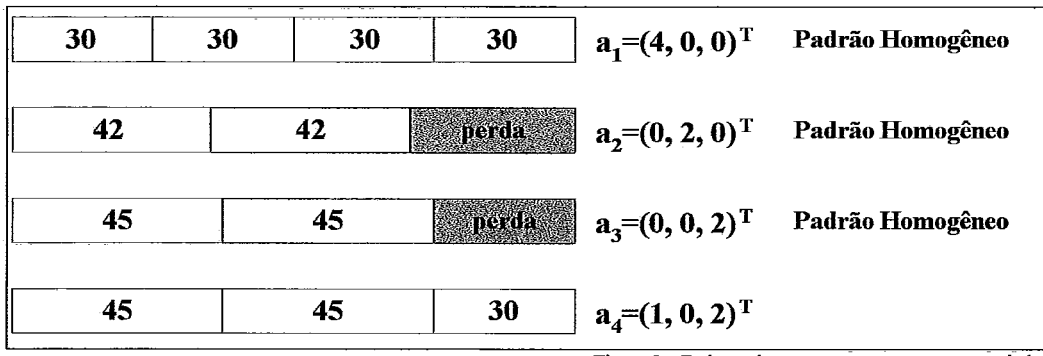

Figura 2 – Padrões de corte com seus vetores associados.

Digamos que x_1, x_2, x_3, ..., x_p sejam o número de vezes que os padrões 1, 2, 3, ..., p serão respectivamente utilizados para cumprir as demandas $d_1 = 1000$, $d_2 = 1250$ e $d_3 = 2000$:

$$a_1x_1 + \cdots + a_px_p = d \quad \Rightarrow \quad \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} x_1 + \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} x_2 + \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} x_3 + \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} x_4 + \cdots + \begin{bmatrix} a_{1p} \\ a_{2p} \\ a_{3p} \end{bmatrix} x_p = \begin{bmatrix} 1000 \\ 1250 \\ 2000 \end{bmatrix}$$

Veja que a segunda etapa corresponde a resolver um sistema de n equações lineares algébricas com p variáveis, onde p é o número de padrões de corte gerados na etapa anterior. O modelo básico pode então ser escrito como:

1.a etapa Encontre todas as p colunas que definem os padrões de corte, segundo as restrições:

$$\sum_{i=1}^{n} \ell_i a_{ij} \leq L, \; j = 1, \ldots, p \qquad (7.1)$$

$$a_{ij} \geq 0, \text{inteiro}, \; i = 1, \ldots, n, \; j = 1, \ldots, p \qquad (7.2)$$

2.a etapa Resolva o seguinte Problema de Otimização Linear Inteiro:

$$\text{minimizar} \quad \sum_{j=1}^{p} c_j x_j \qquad (8.1)$$

sujeito a:

$$a_1x_1 + \cdots + a_px_p = d \qquad (8.2)$$

$$x_j \geq 0, \text{inteiro}, \; j = 1, \ldots, p \qquad (8.3)$$

No problema (8.1 – 8.3), o parâmetro c_j da função objetivo pode representar um custo operacional do padrão de corte j, mas, ele também pode ser considerado unitário e, neste caso, o objetivo será o de minimizar o número de objetos necessários para o cumprimento das demandas dos itens.

Ressaltamos que p pode atingir a ordem de vários milhões ou bilhões, de modo que, utilizar o Método do Simplex para resolver o problema não é computacionalmente viável. GILMORE e GOMORY (1961, 1963) descrevem a Técnica de Geração de Colunas que é capaz de resolver o problema, apêndice B.

Um outro aspecto importante é o fato das variáveis x_j serem inteiras, o que torna o problema muito difícil. O problema de se obter soluções inteiras para um problema de programação linear é foco de estudo de vários pesquisadores como veremos na próxima seção.

3.1 – Solução Inteira e Arredondamento

O momento é oportuno para falarmos da inerente dificuldade de se obter uma solução inteira de um problema de programação linear, em particular estamos interessados nos problemas de corte e empacotamento. O que normalmente se faz é obter uma solução inteira viável por meio de técnicas de arredondamento.

GILMORE e GOMORY (1961, 1963) apresentam resultados práticos de que vale a pena terminar a execução da geração de colunas se a função objetivo não for melhorada em mais de um décimo do seu valor anterior, e também sugerem arredondar as variáveis para cima ou para baixo no caso das demandas serem suficientemente altas. Infelizmente este procedimento de arredondamento não é satisfatório quando as demandas são relativamente baixas, pois, neste caso elas podem ser comprometidas em excesso ou em falta.

GOULIMIS (1990) descreve uma enumeração de padrões viáveis, resolvendo o problema inteiro por meio de um *branch-and-bound* que utiliza planos de corte. Ele menciona o trabalho de MARCOTTE (1985, 1986) em que ela prova que certos exemplos de problemas de corte possuem a propriedade de arredondamento para cima (*IRUP - Integer Round-Up Property*), que foi definida por BAUM e TROTTER (1982). Para compreendermos a propriedade IRUP, seja um exemplo arbitrário e considere:

$$z_{PI} = \min\left\{ \sum_{j=1}^{p} x_j \;\middle|\; \sum_{j=1}^{p} a_{ij} x_j \geq b_i,\ x_j \geq 0 \text{ e inteiro},\ i = 1, \ldots, n \right\}$$

$$z_{PL} = \min\left\{ \sum_{j=1}^{p} x_j \;\middle|\; \sum_{j=1}^{p} a_{ij} x_j \geq b_i,\ x_j \geq 0 \text{ e racional},\ i = 1, \ldots, n \right\}$$

A propriedade se verifica quando $z_{PI} = \lceil z_{PL} \rceil$, onde $\lceil z_{PL} \rceil$ é a função menor inteiro maior ou igual a z_{PL} (função teto). MARCOTTE (1986) apresentou um exemplo artificial com coeficientes da ordem 10^7 para a qual a propriedade não se verifica, na verdade para seu exemplo $z_{PI} - \lceil z_{PL} \rceil = 1$.

SCHEITHAUER e TERNO (1991) comentam que FIELDHOUSE (1990) apresentou um exemplo com coeficientes muito menores que os de MARCOTTE (1986) em que $z_{PI} = 33$ e $z_{PL} = 31{,}96666\ldots$. Os próprios autores apresentam um exemplo com *gap* igual a 1, 0378... .

Posteriormente SCHEITHAUER e TERNO (1995) definem a propriedade de arredondamento para cima modificada (*MIRUP - Modified Integer Round-Up Property*), neste caso a propriedade se verifica quando $z_{PI} \leq \lceil z_{PL} \rceil + 1$. Neste trabalho os autores conjecturam que o problema de corte unidimensional apresenta a propriedade MIRUP.

STADTLER (1990) apresenta um procedimento heurístico de arredondamento das soluções. Inicialmente é utilizada a heurística FFD para se obter uma boa solução inicial, em seguida é feito o arredondamento para cima da variável com a respectiva parte fracionária mais alta, que passa a ficar fixa. O problema original é reformulado e reotimizado, o processo é repetido até que a solução seja inteira.

Stadtler faz referência a padrões que geram demanda em excesso, ele adota a postura de decrementar em uma unidade o valor da variável inteira que indica o número de vezes que este padrão deve ser utilizado, ou seja, da variável do problema principal associada ao padrão. Esta postura pode levar a demandas não cumpridas, neste caso ele utiliza a heurística FFD (página 28 deste capítulo) para a conclusão de seu procedimento.

WÄSCHER e GAU (1996) efetuaram testes numéricos com 4000 exemplos gerados aleatoriamente, além de proporem variações na heurística de Stadtler.

PINTO (1999), estuda o problema de corte bidimensional inteiro por meio da Técnica de Geração de Colunas de GILMORE e GOMORY (1961, 1963), apêndice B. Após a resolução do problema contínuo, ela usa a postura de truncar todas as variáveis (arredondá-las para baixo), em seguida um problema residual é formulado a partir da demanda remanescente. Este novo problema é então resolvido e o processo é repetido até que o truncamento das variáveis forneça uma solução nula. Ao final, é bem possível que uma pequena demanda ainda deva ser atendida, isso é feito por meio de uma heurística gulosa. Ressaltamos que os padrões dos problemas residuais devem ser construídos restritos à demanda remanescente. Este é o procedimento que adotamos para obter uma solução inteira, porém, não ótima para o PCBA.

GLOVER (1978) descreve um *branch-and-bound* paramétrico, produzindo custos para as ramificações da árvore, esta estratégia parece ser interessante e recentemente tem sido explorada por vários pesquisadores para obter uma solução inteira de um problema de programação linear. VANCE *et al.* (1994), sugerem o emprego de um *branch-and-bound* que envolve a resolução de geração de colunas (veja apêndice B) pela Decomposição de DANTZIG e WOLFE (1959) em cada nó da árvore. Posteriormente VANCE (1996) sugere um *branch-and-price*, que desconsidera conjuntos de colunas que tem suas respectivas variáveis com valor nulo na solução ótima. Em seguida, a otimalidade do problema é checada por meio de um subproblema que tenta identificar colunas a entrar na corrente base, que uma vez encontradas são adicionadas ao problema original que é então reotimizado. A ramificação ocorre quando nenhuma coluna é encontrada e a integralidade do problema ainda não está satisfeita. Para um estudo mais detalhado desta abordagem recomendamos o trabalho de BARNHART *et al.* (1998).

CARVALHO (1998) também utiliza esta estratégia com uma formulação alternativa na resolução de problemas de corte.

NITSCHE *et al.* (1999) descrevem dois modelos para o problema de corte de estoque, que segundo eles apresentam algumas vantagens quando relaxados para o caso contínuo, o que pode ser proveitoso na abordagem do *branch-and-price*. Nesta mesma linha de raciocínio citamos o trabalho de BABAYEV e MARDANOV (1994), onde os autores descrevem um procedimento que reduz o número de variáveis em problemas de programação linear inteiro.

Um trabalho muito interessante é o de VANDERBECK (2000), onde o autor discute a decomposição de Dantzig-Wolfe para problemas lineares inteiros e descreve critérios de ramificação para um *branch-and-price*.

Os trabalhos de AKKER *et al.* (2000) e JOHNSON *et al.* (2000) são outros dois que discutem a utilização de *branch-and-price* para resolver problemas lineares inteiro.

Em seus trabalhos, FARIAS *et al.* (2000) estudam critérios especiais de ramificação em *branch-and-price*.

Para encerrar o assunto sobre soluções inteiras de problemas lineares destacamos o trabalho de MACULAN *et al.* (1999) que sugerem um método *branch-and-price* com geração de colunas, utilizando os métodos primal e dual simplex. Os autores têm especial intereste em problemas de redes telefônicas.

3.2 – Corte de Estoque com várias barras em estoque, em quantidades Ilimitadas

Apresentamos agora o caso em que o estoque é composto por várias barras, cujas quantidades de cada tipo de barra é suficiente para atender toda a demanda. Uma situação em que este tipo de problema pode ser aplicado, ocorre em indústrias onde os objetos são produzidos por máquinas diferentes, cuja capacidade de produção é alta. Uma outra seria em casos onde objetos de vários tamanhos são adquiridos no mercado, cuja oferta é grande.

Relembrando os dados dos itens:

Dados dos itens:

n: número total de itens;

ℓ_i: largura do item i, $i = 1, \ldots, n$;

d_i: demanda do item i, $i = 1, \ldots, n$;

Os dados do estoque são:

Dados do estoque:

m: número total de tipos de objetos (barras) em estoque;

L_r : largura do objeto (barra) r, $r = 1, \ldots, m$;

c_r : custo do objeto (barra) r, $r = 1, \ldots, m$;

No modelo matemático deste caso os padrões de corte devem ser definidos para cada tipo de objeto (barra) em estoque. A variável de decisão é x_j^r que define o número de vezes que o padrão j é usado para cortar um objeto do tipo r.

1.a etapa Defina os padrões de corte para cada barra $r = 1, \ldots, m$, isto é, todas as soluções do sistema (9.1 - 9.2):

$$\sum_{i=1}^{n} \ell_i a_i^r \leq L_r \qquad (9.1)$$

$$a_i^r \geq 0, \text{ inteiro}, i = 1, \ldots, n \qquad (9.2)$$

Seja p_r o total das possíveis soluções do sistema anterior, e considere a próxima etapa.

2.a etapa Resolva o Problema de Otimização Linear Inteiro (10.1 – 10.3), onde uma coluna $(a_{1j}^r, \ldots, a_{nj}^r)^T$ é solução do sistema (9.1 – 9.2).

$$\text{minimizar} \quad \sum_{j=1}^{p_1} c_1 x_j^1 + \cdots + \sum_{j=1}^{p_m} c_m x_j^m \qquad (10.1)$$

sujeito a:

$$\begin{aligned} &\sum_{j=1}^{p_1} a_{1j}^1 x_j^1 + \cdots + \sum_{j=1}^{p_m} a_{1j}^m x_j^m = d_1 \\ &\quad \vdots \qquad\qquad\qquad \vdots \qquad \vdots \\ &\sum_{j=1}^{p_1} a_{nj}^1 x_j^1 + \cdots + \sum_{j=1}^{p_m} a_{nj}^m x_j^m = d_n \end{aligned} \qquad (10.2)$$

$$x_j^r \geq 0, \text{inteiro}, r = 1, \ldots, m, j = 1, \ldots, p_r \qquad (10.3)$$

3.3 – Corte de Estoque com várias barras em estoque, em quantidades Limitadas

Como no caso anterior temos várias barras em estoque, porém, em quantidades limitadas. Este problema é observado em indústrias onde os objetos são adquiridos com antecedência e estocados, devido à instabilidade de se obter a matéria-prima no mercado, por exemplo, bobinas de aço. Numa outra situação, podem ser produzidos, contudo, a capacidade de produção é limitada.

Os padrões são definidos como os do caso precedente, e na segunda etapa uma disponibilidade e_r, $r = 1, \ldots, m$, deve ser considerada sobre o estoque. O modelo matemático torna-se:

$$\text{minimizar} \quad \sum_{j=1}^{p_1} c_1 x_j^1 + \cdots + \sum_{j=1}^{p_m} c_m x_j^m \qquad (11.1)$$

sujeito a:

$$\begin{aligned} &\sum_{j=1}^{p_1} a_{1j}^1 x_j^1 + \cdots + \sum_{j=1}^{p_m} a_{1j}^m x_j^m = d_1 \\ &\quad \vdots \qquad\qquad\quad \vdots \quad \vdots \qquad (11.2) \\ &\sum_{j=1}^{p_1} a_{nj}^1 x_j^1 + \cdots + \sum_{j=1}^{p_m} a_{nj}^m x_j^m = d_n \end{aligned}$$

$$\begin{aligned} &\sum_{j=1}^{p_1} x_j^1 \qquad\qquad\qquad \le e_1 \\ &\qquad\quad \ddots \qquad\qquad \vdots \qquad (11.3) \\ &\qquad\qquad\qquad \sum_{j=1}^{p_m} x_j^m \le e_m \end{aligned}$$

$$x_j^r \ge 0, \text{inteiro}, \; r = 1, \ldots, m, \; j = 1, \ldots, p_r \qquad (11.4)$$

Em todos os modelos apresentados fica claro que o número de padrões viáveis a serem investigados é extremamente alto. Assim, o número de variáveis dos modelos é enorme, podendo em problemas práticos chegar facilmente a vários milhões ou bilhões. Felizmente estes modelos apresentam uma estrutura particular (as colunas têm uma lei de formação, 9.1 e 9.2) que permite um trabalho implícito das colunas da matriz de restrições pelo emprego da Técnica de Geração de Colunas, apêndice B.

4 – Textos importantes sobre Corte e Empacotamento

A seleção de um subconjunto de padrões de corte viáveis e a análise da função objetivo é uma abordagem clássica, cuja dificuldade é justamente a definição deste subconjunto. As formulações de EISEMANN (1957) e PIERCE (1966) baseiam-se nela.

Um texto clássico (KANTOROVICH, 1960), tradução de um trabalho em russo de 1937 do mesmo autor, aborda a necessidade de se fazer um planejamento consciente dos processos de produção.

GILMORE e GOMORY (1961, 1963, 1965, 1966) foram pioneiros ao descreverem a Técnica de Geração de Colunas que faz uso da estrutura particular de um problema de corte, onde as colunas da matriz de restrição obedecem a uma lei de formação bem definida. Paralelamente, DANTZIG e WOLFE (1959) descrevem uma decomposição para problemas de programação linear, também baseada na estratégia de gerar colunas, entretanto, ela difere da técnica desenvolvida por Gilmore-Gomory (veja apêndice B). Todos estes textos são ricos em conteúdo, portanto, referências singulares.

HAESSLER (1975, 1980) sugere modificações na formulação dada por GILMORE e GOMORY (1961, 1963), controlando a troca de padrões de corte por uma penalização da função objetivo e pela introdução de uma margem de produção para os itens, o que veio melhorar a qualidade das soluções, entretanto, a resolução do problema ficou computacionalmente prejudicada. O modelo de Haessler tem o seguinte aspecto:

$$\text{minimizar} \quad c_1 \sum_{j=1}^{p} T_j x_j + c_2 \sum_{j=1}^{p} \delta(x_j) \qquad (12.1)$$

sujeito a:

$$d_i^{min} \leq \sum_{j=1}^{p} a_{ij} x_j \leq d_i^{max}, \quad i=1,\ldots,n \qquad (12.2)$$

$$x_j \geq 0, a_{ij} \geq 0 \text{ e inteiros}, i = 1, \ldots, n, j = 1, \ldots, p \qquad (12.3)$$

Nele d_i^{min} e d_i^{max} são respectivamente as demandas mínima e máxima de cada item $i=1, \ldots, m$, e a_{ij} é o número de itens do tipo i no padrão j. No objetivo, T_j é a perda de material (*trim-loss*) associada ao padrão j, c_1 é o custo de material perdido, e c_2 é o custo pela troca de um padrão (onde $\delta(x_j)=1$ se $x_j > 0$ e 0 no caso contrário).

Observe que a função δ no objetivo do modelo de Haessler tem o papel de reduzir o número de **tipos** de padrões a serem utilizados. Nesta mesma linha de raciocínio existe o trabalho de McDIARMID (1999) que visa à redução do número de padrões de corte.

GOLDEN (1976) publica um artigo de revisão sobre o problema de corte unidimensional, revendo a abordagem de GILMORE e GOMORY (1961, 1963) e propondo o emprego do método do subgradiente.

JOHNSTON (1981) relata a importância destes problemas para o caso da indústria de papel, e HAESSLER (1971) descreve um modelo não linear e um procedimento heurístico que gera, seqüencialmente, padrões de corte e "níveis de uso" até que todas as demandas sejam cumpridas. A cada iteração a pesquisa é dependente das demandas ainda não cumpridas, de modo que, um "melhor" padrão de corte é escolhido. SWEENEY e HAESSLER (1990) descrevem um procedimento heurístico para o caso em que as bobinas apresentam uma graduação na qualidade do papel.

Um estudo mais aprofundado sobre o corte de bobinas de papel é encontrado em JARDIM CAMPOS e MACULAN (1988, 1995), os autores utilizam as técnicas de relaxação lagrangiana e do subgradiente. Também é abordado o problema do sequenciamento dos padrões, cujo objetivo é diminuir o tempo de acerto de facas.

Outros aspectos relevantes na questão do sequenciamento são a redução do número de pilhas abertas no corte de placas de madeira reconstituída, e a minimização do tempo gasto para obter os itens que compõem a encomenda de cada cliente no corte de lâminas de vidro. YANASSE (1997) apresenta um texto onde descreve estes aspectos. Citamos também os trabalhos: (YANASSE, 1996), (YANASSE *et al.*, 1998, 1999), (BECCENERI e SOMA, 1998), (LIMEIRA, 1998) e (LINHARES, 1998).

HINXMAN (1980) apresenta um trabalho de revisão sobre várias abordagens, e observa que o emprego de heurísticas, mesmo funcionando satisfatoriamente pode produzir resultados inaceitáveis para algum exemplo do problema. De fato, conforme vem relatando Haessler em seus trabalhos, tendo em vista sua experiência em diversos problemas concretos, uma heurística simples não é capaz de prever as inúmeras situações que surgem, justificando a construção de procedimentos específicos ao problema. O desenvolvimento de algoritmos híbridos parece ser uma postura satisfatória, conforme descreve CINTRA (1998a e b).

DYCKHOFF (1990) propõe uma tipologia para os problemas de corte e empacotamento. Mais tarde DYCKHOFF e FINKE (1992) publicam um livro que trata em detalhes do assunto e traz uma vasta e rica bibliografia.

DAGLI (1990) revê alguns métodos e discute a possibilidade de se empregar técnicas de Inteligência Artificial aos problemas de corte e empacotamento. BEASLEY (1998) relata o emprego de algoritmos genéticos em problemas de corte e empacotamento e FOERSTER e WÄSCHER (1996) utilizam a técnica do *Simulated Aneealing* na questão do sequenciamento de padrões.

GEMMILL e SANDERS (1990) estudam um problema estocástico, onde é preciso determinar as dimensões e quantidades de objetos a serem mantidos em estoque, visando à minimização de perdas de material. CHU e ANTONIO (1999) descrevem algoritmos aproximativos para resolver o problema de corte de estoque, visando à minimização da perda de material e do tempo de corte.

Um outro interesse muito comum em problemas de corte e empacotamento é combinar objetivos conflitantes, por exemplo, minimizar uma combinação de custos de processamento com custos de armazenamento, e ainda, maximizar a produção. WÄSCHER (1990) descreve uma formulação para problemas com objetivos múltiplos e discute sua implementação. T. G. ROCHA (1978) e T. G. ROCHA *et al.* (1981a e b) apresentam estudos com objetivos conflitantes no armazenamento de bobinas de cabos telefônicos.

M. N. ROCHA (1996, 1997) estuda o problema do corte de perfis na confecção de esquadrias de alumínio, efetuando uma comparação dos resultados entre a Técnica de Geração de Colunas de GIMORE e GOMORY (1961, 1963) e a heurística FFD. O autor relata que na prática, as empresas envolvidas preferiram utilizar as soluções obtidas pela heurística FFD devido à simplicidade da estrutura dos padrões de corte, muito embora, as perdas eram em geral superiores que as dos padrões obtidos por Geração de Colunas.

Em geral os métodos de resolução dos problemas de corte e empacotamento são por meio de heurísticas. ZANAKIS, EVANS e VAZACOUPOULOS (1989), num artigo de revisão sugerem uma classificação, tendo em vista as aplicações práticas, baseada nas seguintes técnicas:

Heurísticas de construção: constroem a solução passo a passo;

Heurísticas de otimização local: efetuam uma pesquisa local na "vizinhança" de uma solução viável;

Heurísticas de decomposição: dividem o problema original em subproblemas dependentes entre si (geração de colunas), ou independentes (decomposição Lagrangiana);

Heurísticas de partição: dividem o problema original em subproblemas independentes, de modo que suas soluções são agregadas para formar uma solução do problema original;

Heurísticas de restrição do espaço de soluções: restringem o espaço de soluções no intuito de tornar o problema menos árduo;

Heurísticas de relaxação do espaço de soluções: a relaxação lagrangiana e a agregação de restrições são exemplos;

Programação matemática: a resolução de um problema de programação linear com posterior arredondamento da solução é um exemplo;

Existem ainda muitos outros textos sobre Corte e Empacotamento de igual importância, recomendamos a bibliografia contida em (DYCKHOFF e FINKE, 1992).

5 – O caso Bidimensional

Nesta seção faremos um relato bastante restrito de problemas de corte em duas dimensões, uma das razões é o fato do PCBA ser unidimensional, uma outra é que o caso bidimensional é muito mais rico em dificuldades e situações especiais a serem estudadas, não obstante, fornecemos uma boa literatura para um estudo aprofundado deste caso.

Considere objetos retangulares de dimensões (L, W) e m itens de dimensões (ℓ_i, w_j) a serem cortados destes objetos. GILMORE e GOMORY (1965) apresentam um método simples para a obtenção dos itens, consistindo de duas etapas. Na primeira delas é resolvido o seguinte problema:

$$v_j = \text{máximo} \sum_{i \in W_j} u_i \lambda_{ij} \qquad (13.1)$$

sujeito a:

$$\sum_{i \in W_j} \ell_i \lambda_{ij} \leq L,\ j = 1, \ldots, m \text{ e } W_j = \{ i \mid w_i \leq w_j \} \qquad (13.2)$$

$$\lambda_{ij} \geq 0 \text{ e inteiro},\ i \in W_j,\ j = 1, \ldots, m \qquad (13.3)$$

Observe que são construídas m tiras de dimensões (L, w_j), onde λ_{ij} é o número de vezes que a largura ℓ_i aparece numa tira (L, w_j). Numa segunda etapa é considerada a variável μ_j, número de tiras do tipo (L, w_j) arranjadas num retângulo (L, W), e resolvido o seguinte problema:

$$\text{maximizar} \sum_{j=1}^{m} v_j \mu_j \qquad (14.1)$$

sujeito a:

$$\sum_{j=1}^{m} w_j \mu_j \leq W \qquad (14.2)$$

$$\mu_j \geq 0 \text{ e inteiro},\ j = 1, \ldots, m \qquad (14.3)$$

Com este procedimento é obtido um padrão de corte guilhotinado em 2-estágios, isto é, os itens podem ser obtidos por meio de cortes guilhotinados, aqueles que ao serem efetuados produzem dois novos retângulos, além do mais, são em 2-estágios, o que significa que apenas uma mudança no sentido dos cortes é permitida, figura 3a.

Quando os itens são obtidos por meio de cortes não guilhotinados o padrão é denominado não guilhotinado, um exemplo é dado na figura 3b.

Na figura 3c é dado um exemplo de um padrão de corte guilhotinado 3-estágios, observe que para obter os itens do tipo 1 são necessários três tipos de corte guilhotinados. GILMORE e GOMORY (1965) apresentam, em adição, um algoritmo para padrões n-estágios.

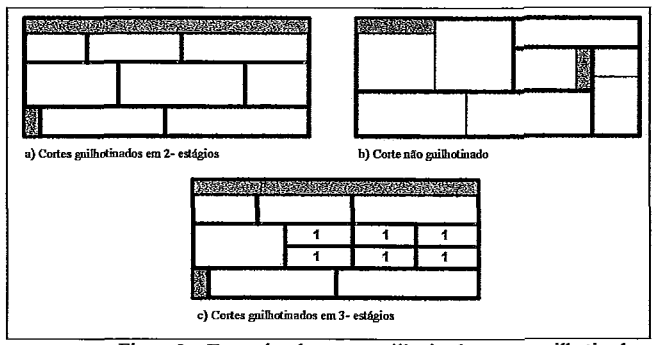

Figura 3 – Exemplos de cortes guilhotinados e não guilhotinados.

CHRISTOFIDES e WHITLOCK (1977) descrevem um *branch-and-bound* para o problema restrito (o número de vezes que um item aparece num padrão de corte é limitado). Neste trabalho são discutidos um procedimento de enumeração dos padrões e um procedimento de programação dinâmica para o caso irrestrito. WANG (1983) apresenta um procedimento heurístico para o caso restrito baseado em combinações verticais e horizontais dos retângulos demandados.

OLIVEIRA e FERREIRA (1990) também propõem uma heurística para o caso restrito, e ALVARENGA e MACULAN (1986) apresentam uma heurística baseada nas idéias de GILMORE e GOMORY (1965).

MORABITO e ARENALES (1995) apresentam um artigo que compara a performance de duas heurísticas: a primeira delas foi proposta por BEASLEY (1985) e é baseada em programação dinâmica, a segunda utiliza as idéias de GILMORE e GOMORY (1965). Ainda em 1995, e posteriormente em 1996, Morabito e Arenales sugerem a representação de um padrão de corte como sendo um caminho completo num grafo "e/ou" (na verdade uma árvore), cujas arestas (na verdade arcos) relacionam um nó com um subconjunto de nós. Em MORABITO (1992) e ARENALES (1993) esta abordagem em grafo "e/ou" é estudada com maiores detalhes.

YANASSE *et al.* (1990) sugerem um algoritmo heurístico para um problema bidimensional em que a demanda deve ser cumprida exatamente, onde é combinado um procedimento de geração dos padrões e um esquema de enumeração para ordenar as placas do estoque.

DOWSLAND (1990) estuda o problema do carregamento automatizado de paletes, onde um grafo representa o *layout* de uma camada no carregamento, podendo ser dividido em *sub-layouts* (as componentes conexas do grafo).

FARLEY (1990a) apresenta um estudo comparativo entre as placas do estoque e os estilos dos padrões, em seguida FARLEY (1990b) apresenta uma heurística baseada na abordagem de GILMORE e GOMORY (1965) para um problema na indústria de lonas.

COFFMAN e SHOR (1990) efetuam uma análise comparativa entre alguns algoritmos para problemas em duas dimensões.

BISCHOFF e MARRIOT (1990) efetuam uma avaliação comparativa entre quatorze critérios heurísticos usados no carregamento de contêineres. GEHRING *et al.* (1990) também estudam o mesmo problema e sugerem uma heurística especializada, diferente de GEORGE e ROBINSON (1980) e HAESSLER e TALBOT (1990) que definem subconjuntos de itens e efetuam empacotamento por camadas.

DOWSLAND e DOWSLAND (1992, 1995) apresentam um artigo de revisão das técnicas utilizadas nos problemas de empacotamento. BISCHOFF *et al.* (1995) propõem uma heurística para o carregamento de paletes com itens não homogêneos.

CUNG *et al.* (1997) sugerem um *branch-and-bound* para resolver o problema restrito exatamente, consistindo numa modificação do algoritmo desenvolvido por VISWANATHAN e BAGCHI (1993), conhecido como *VB-algorithm*, pela introdução de *bounds* mais refinados. Na verdade HIFI (1997) já havia feito modificações neste mesmo algoritmo que passou a ser denominado *MVB-algorithm*.

Muitos outros pesquisadores têm apresentado heurísticas para problemas bidimensionais destacamos: FAYARD e ZISSIMOPOULOS (1995) estudam problemas irrestritos, DAZA *et al.* (1995) estudam problemas restritos, KRÖGER (1995) utiliza uma abordagem baseada em algoritmos genéticos e AMARAL e WRIGHT (2001) apresentam um novo e eficiente algoritmo para o caso restrito, além de descreverem novas propriedades para padrões de corte guilhotinados.

6 - O Problema do *Bin-Packing*

Sua descrição é simples e pode ser comparada à do *Cutting-Stock*: considere um número ilimitado de caixas (*bin*) B_j $j = 1, 2, 3, \ldots$ de capacidade W e n itens de tamanhos w_i, $0 < w_i \leq W$, $i = 1, \ldots, n$, que devem ser empacotados no menor número possível de caixas sem exceder a capacidade W.

Em problemas práticos podem ocorrer outras operações (transporte, armazenamento, etc) antes que o empacotamento global seja concluído. ALOISE e MACULAN (1991) estudam esta situação e apresentam uma classe de algoritmos aproximativos decrescentes para o problema. SILVA e SOMA (2001), apresentam duas heurísticas para o empacotamento de *bins* tridimensionais, onde é considerado o problema da estabilidade do empacotamento.

Para descrevermos as heurísticas mais conhecidas de resolução do *Bin-Packing*, que podem ser encontradas em CAMPELLO e MACULAN (1994), adotaremos dois procedimentos:

Procedimento CaixaNova$(p, B_p, c(B_p), w_k)$;{ $c(B_p)$: capacidade utilizada do *bin* B_p }
Início
$p := p+1$;
$c(B_p) := w_k$;
$B_p := \{ k \}$;
Fim;

Procedimento CaixaUsada$(p, B_p, c(B_p), w_k)$;
Início
$c(B_p) := c(B_p) + w_k$;
$B_p := B_p \cup \{ k \}$;
Fim;

6.1 - Heurística NF (*Next-Fit*)

Ao ser iniciado o empacotamento de um *bin* B_j não são mais consideradas caixas B_i, $i < j$, mesmo que eventualmente seja possível empacotar algum item numa destas caixas.

Procedimento NF$(n, w_1, \ldots, w_n, W)$;
Início
$c(B_1) := w_1$;
$p := 1$;
$B_1 := \{ 1 \}$;
Para $j = 2$ até n faça
Se $(W - c(B_p)) \geq w_j$ então CaixaUsada$(p, B_p, c(B_p), w_j)$
senão CaixaNova$(p, B_p, c(B_p), w_j)$;
fimPara;
Fim;

6.2 - Heurística NFD (*Next-Fit Decreasing*)

Este procedimento é idêntico ao anterior, porém, a lista de itens é ordenada de forma não crescente $(w_1 \geq w_2 \geq \ldots \geq w_n)$.

6.3 - Heurística FF (*First-Fit*)

Neste caso a heurística aproveita os espaços ociosos das caixas já utilizadas, em outras palavras, para qualquer item k considere B_j o *bin* não vazio de maior índice e verifique se existe algum *bin* B_i, $1 \leq i \leq j$, tal que $w_k \leq W - c(B_i)$. Caso ele não exista empacote o item k numa nova caixa.

```
Procedimento FF(n, w1, ..., wn, W);
Início
        c(B1) : = w1;
        p: = 1;
        B1 : = { 1 };
        Para j = 2 até n faça
                encontrou: = 0; k: = 1;
                Enquanto (k ≤ p) e (encontrou = 0) faça
                        Se wj ≤ W − c(Bk) então
                                CaixaUsada(k, Bk, c(Bk), wj);
                                encontrou: = 1;
                        fimSe
                        senão k: = k + 1;
                        Se encontrou = 0 então CaixaNova(k, Bk, c(Bk), wj);
        fimPara;
Fim;
```

6.4 - Heurística FFD (*First-Fit Decreasing*)

O procedimento é idêntico ao anterior, porém, a lista de itens é ordenada de forma não crescente $(w_1 \geq w_2 \geq \ldots \geq w_n)$.

6.5 - Heurística BF (*Best-Fit*)

Neste caso o procedimento procura minimizar os espaços ociosos das caixas já utilizadas, em outras palavras, para qualquer item k considere $N_{2(I)}$ o *bin* não vazio de maior índice e verifique se existe algum *bin* B_r $(1 \le r \le j)$, tal que $f_r = \text{mínimo}\{f_i = (W - c(B_i)) - w_k > 0, 1 \le i \le j\}$. Caso ele não exista empacote o item k numa nova caixa.

Procedimento BF$(n, w_1, \ldots, w_n, W)$;
Início
    $c(B_1) := w_1$;
    p: = 1;
    $B_1 := \{ 1 \}$;
    Para j = 2 até n faça
        $f_r = \min_{1 \le k \le p} \{f_k = W - c(B_k) - w_j \mid f_k > 0\}$
        Se $f_r > 0$ então CaixaUsada$(k, B_k, c(B_k), w_j)$
        Senão CaixaNova$(k, B_k, c(B_k), w_j)$;
    fimPara;
Fim;

6.6 - Heurística BFD (*Best-Fit Decreasing*)

O procedimento é idêntico ao anterior, porém, a lista de itens é ordenada de forma não crescente $(w_1 \ge w_2 \ge \ldots \ge w_n)$.

No pior caso a razão de performance das heurísticas anteriores é estabelecida pelo seguinte resultado, cuja demonstração pode ser vista em JOHNSON *et al.* (1974):

Teorema Seja I um exemplo qualquer do Problema do *Bin-Packing*, considere $x^*(I)$ o número ótimo de caixas. Nestas condições os melhores limites superiores são:

a) $x^{FF}(I)$ ou $x^{BF}(I) \le \frac{17}{10}x^*(I) + 2$ b) $x^{FFD}(I)$ ou $x^{BFD}(I) \le \frac{11}{9}x^*(I) + 4$

Por meio de testes exaustivos, GOLDEN (1976) observou que as soluções produzidas pelas heurísticas são em geral boas, entretanto, alguns exemplos apresentaram soluções perto do pior caso. Sem dúvida o desenvolvimento de heurísticas é uma arte. HEISE e WÄSCHER (1996) descrevem um gerador para o problema do *Bin-Packing*, e FEKETE e SCHEPERS (1998) descrevem um procedimento simples de obter limitantes inferiores do problema. SCHWERIN e WÄSCHER (1999) apresentam novos limitantes inferiores para o problema.

ZHANG *et al.* (2000) apresentam um algoritmo aproximativo para o problema do *Bin-Packing* com razão absoluta no pior caso igual a $\frac{3}{2}$. Este é o melhor resultado possível a menos que $P = NP$, veja apêndice C sobre complexidade.

O Problema de Empacotamento também é amplamente estudado pelo interesse teórico, como pelas aplicações destacamos: BISCHOFF e RATCLIFF (1995) que abordam o problema do carregamento de contêineres, AVRIEL *et al.* (2000) que abordam o problema do acondicionamento de contêineres em porões de navios, NELIβEN (1995) estuda o problema de carregamento de caixas em paletes e GEORGE *et al.* (1995) que tratam do carregamento de itens cilíndricos num contêiner.

Esperamos ter cumprido nossa proposta de um texto com idéias básicas sobre os problemas de corte e empacotamento, e ao mesmo tempo destacar uma literatura rica para um estudo mais aprofundado do assunto. Destacamos uma vez mais que em DYCKHOFF e FINKE (1992) encontra-se uma vasta e rica bibliografia sobre corte e empacotamento.

No Brasil, vários estudos sobre problemas de corte e empacotamento vêm sendo realizados ao longo dos últimos dez anos, gerando dissertações e teses como esta, além de artigos e aplicativos computacionais. Em 1999 a revista brasileira *Pesquisa Operacional*, editada pela SOBRAPO, dedicou a edição número 2, do volume 19, a Problemas de Corte e Empacotamento (ARENALES *et al.*, 1999). Desde 1996 são realizadas "oficinas de estudo" que promovem o encontro dos vários pesquisadores brasileiros, divulgando seus estudos em problemas de corte e empacotamento.

A matemática nos ensina que quem não acredita é cego, mas, ela também ensina que é necessário questionar.

Capítulo 2

O Problema de Corte de Bobinas de Aço (PCBA)

1 – Introdução (características técnicas do problema)

Algumas empresas do ramo da metalurgia confeccionam tubos de aço para diversas aplicações, por exemplo, tubos para caldeiras, para a indústria automobilística, para a construção civil, para a confecção de bicicletas, etc. Estes tubos são produzidos a partir de fitas de aço, que por sua vez são cortadas de bobinas de aço armazenadas na própria empresa, ou adquiridas no mercado. A figura 4 ilustra uma bobina, fitas que podem ser obtidas dela, e os tipos de tubo que podem ser confeccionados.

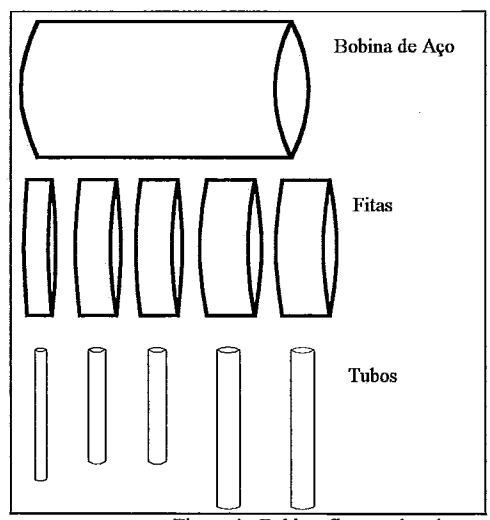

Figura 4 - Bobina, fitas e tubos de aço.

As bobinas do estoque são identificadas pelo seu "peso" (varia de 12000 Kg a 13500 Kg), sua largura (varia de 1100 mm a 1200 mm), espessura do aço (varia de 0,90 mm a 5,10 mm) e pelo teor de carbono do aço segundo os critérios da *SAE (Society of Automotive Engineers)*, os mais utilizados são: SAE 1008 (0,08 % de carbono na liga), SAE 1010 (0,10 % de carbono na liga), SAE 1012 (0,12 % de carbono na liga), SAE 1017 (0,17 % de carbono na liga) e SAE 1021 (0,21 % de carbono na liga).

Os tubos são identificados pelo seu diâmetro em mm (que determinará a largura da fita a ser utilizada na sua confecção), pelo seu comprimento em mm e pelo tipo do aço que em geral deve sofrer laminação a frio (cerca de 40 % da produção passa por este processo). As encomendas associadas aos tubos são feitas em Kg, e em alguns casos muito específicos por unidades.

O esquema de corte (neste texto esquema de corte e padrão de corte são sinônimos) é efetuado em duas fases devido à laminação e o recorte, pois, as máquinas encarregadas dos processos possuem limitações técnicas. Como conseqüência, as fitas devem ser combinadas entre si numa nova entidade denominada bobina intermediária, que sofrerá ou não a laminação. Por outro lado, as fitas não podem ser combinadas indiscriminadamente.

Por exemplo, digamos que as três primeiras fitas da figura 4 componham uma bobina intermediária e que as duas outras fitas componham uma segunda. Suponha que o aço da bobina tenha 1,20 mm de espessura e que as fitas da primeira bobina intermediária precisem ser laminadas a uma espessura de 1,00 mm, enquanto, as da segunda bobina intermediária devem ser laminadas a uma espessura de 0,90 mm (ou num outro exemplo não serem laminadas). A figura 5 ilustra o que acabamos de descrever.

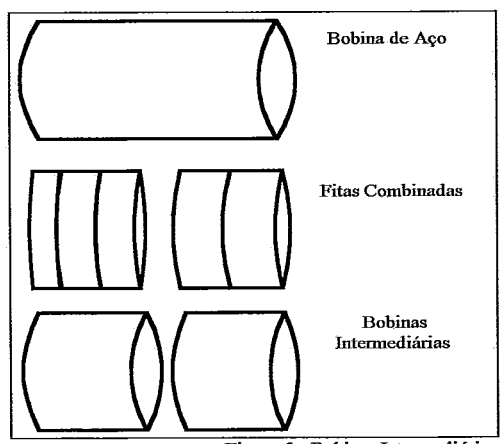

Figura 5 - Bobinas Intermediárias.

Além do processo de laminação, no processo de recorte existem limitações técnicas que obrigam dividir uma bobina em bobinas intermediárias, maiores detalhes serão dados logo mais.

Assim sendo, dois processos de corte estão determinados, do primeiro resultarão as bobinas intermediárias e do segundo as fitas, figura 6. Observe que os padrões de corte do PCBA podem ser estruturados em **compartimentos** (bobinas intermediárias) formados pela combinação de fitas compatíveis entre si.

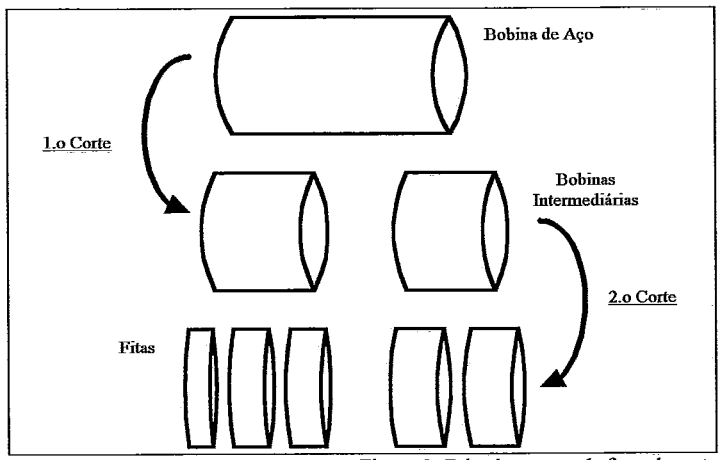

Figura 6 - Primeira e segunda fases de corte.

Podemos simplificar a linha de produção através do esquema da figura 7.

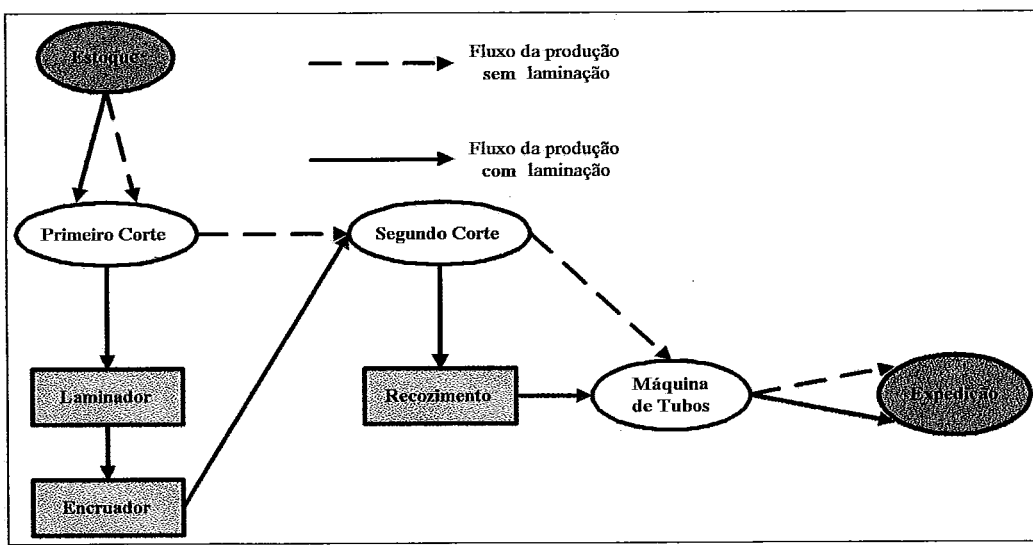

Figura 7 - Esquema simplificado da linha de produção.

A máquina que efetua o primeiro corte (as empresas denominam corte) é alimentada a partir do salão de estoque, onde existe uma ponte rolante (com capacidade de 15000 Kg) para a manipulação das bobinas.

O processo de corte é obtido por cisalhamento do aço através de "discos de corte" (normalmente denominado pelas empresas de corte *sliter*). Por razões técnicas do maquinário o número de cortes numa bobina é limitado (no máximo 12 discos de corte). Nesta etapa existem perdas intrínsecas nas laterais da bobina para eliminar as irregularidades das bordas (varia de 6 mm a 8 mm), figura 8. O tempo de preparação da máquina é cerca de 40 minutos, e o tempo médio para o corte de uma bobina é cerca de 15 minutos.

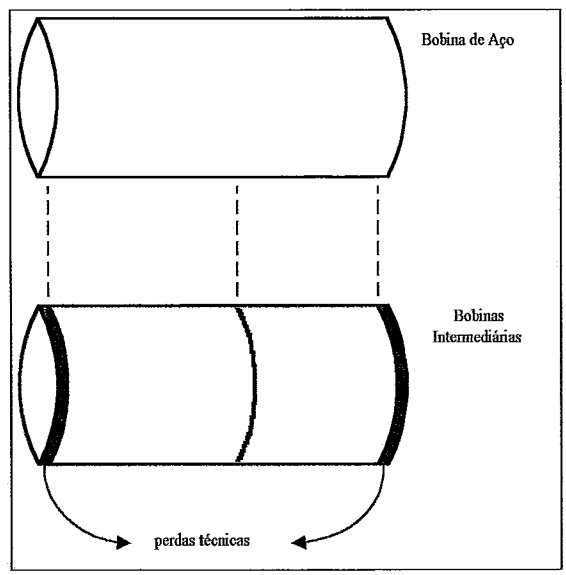

Figura 8 - Ilustração das perdas técnicas.

Uma máquina especial é responsável pelo segundo corte (as empresas denominam recorte, que também é feito por cisalhamento) e nela são obtidas as fitas. Existem duas razões para não obter as fitas diretamente na máquina de corte: a primeira é a laminação, e a segunda é a precisão dos cortes.

Similar ao primeiro corte, os discos do recorte são em número limitado (é possível efetuar no máximo 8 cortes numa bobina intermediária) e existe uma perda intrínseca nas laterais das bobinas intermediárias (varia de 3 mm a 5 mm).

A máquina de recorte atende apenas bobinas intermediárias com no máximo 600 mm de largura. Os tempos de preparação e recorte são os mesmos da etapa de corte.

A seguir descrevemos o processo de laminação que se dá à temperatura ambiente (laminação a frio). A laminação é executada por meio de cilindros de laminação que efetuam pressão sobre a lâmina de aço, figura 9.

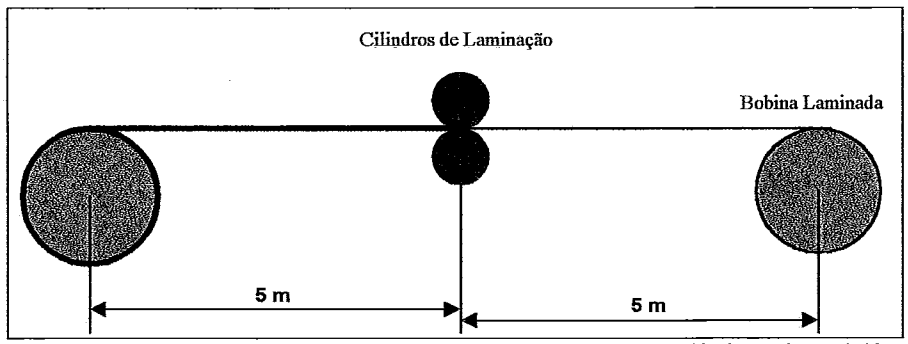

Figura 9 - Laminação a frio do aço de uma bobina.

Por razões técnicas, a laminação de uma bobina intermediária é feita em passos, alternadamente para frente e para trás, até que a espessura desejada seja alcançada (a redução é cerca de 15 a 20 % em cada passo, e de 50 a 60 % no total). Este processo faz com que a dureza do aço aumente, bem como o comprimento da bobina intermediária.

O objetivo principal da laminação é proporcionar certas características mecânicas ao aço, de acordo com a finalidade dos tubos, por exemplo, o aço de tubos que suportam a pressão de fluídos ou gases deve possuir características físicas que são adquiridas por meio da laminação a frio. Em alguns casos a laminação é usada devido a uma deficiência do estoque.

Além de ter um custo considerável, esta operação também produz perdas técnicas (existe uma folga com cerca de 5 m entre a bobina intermediária e os cilindros de laminação, figura 9). Não é possível laminar uma bobina intermediária com menos de 154 mm e mais de 456 mm de largura, daí a razão de dividir uma bobina do estoque.

Associado ao laminador, há o encruador, cuja finalidade é corrigir imperfeições causadas na superfície do aço após o processo de laminação. O encruador é uma máquina que efetua uma laminação bem mais delicada e precisa que o laminador, e embora seja importante no processo de confecção dos tubos, não interfere no processo de cortagem.

Em seguida, são obtidas as fitas que sofrem um processo de recozimento, cujo propósito é corrigir alguns fatores físicos indesejáveis adquiridos pelo aço durante todos os processos.

As fitas são acondicionadas em fornos especiais, onde permanecem por um certo período, e em seguida, são lentamente resfriadas até a temperatura ambiente, o processo todo leva cerca de 24 horas e pode facilmente constituir um gargalo na produção.

Tanto as bobinas intermediárias quanto as fitas, são manipuladas por duas pontes rolantes (uma com capacidade de 10000 Kg e outra de 15000 Kg), o que não introduz novas restrições na construção dos padrões.

A próxima etapa consiste na obtenção dos tubos. O tempo de preparação do maquinário responsável pela confecção dos tubos varia de 2,5 a 3,5 horas. A última etapa da linha de produção é a embalagem dos tubos e armazenamento para expedição.

Em síntese, os diversos aspectos da linha de produção aqui citados apontam para um problema complexo e que foge do contexto clássico, até porque, o esquema de corte se dá em duas fases. Além disso, a necessidade do processo de laminação induz padrões de corte nada evidentes, visto que nem todas as fitas são compatíveis entre si. Em visita a uma empresa do ramo, constatamos uma perda de material da ordem de 10 %, o que correspondia a uma média de 500 toneladas de aço ao mês.

2 – Revisão da Literatura sobre o PCBA

Problemas de corte em duas fases não são muito estudados na literatura. Antes de tratarmos especificamente do PCBA, descreveremos rapidamente um problema que HAESSLER (1979) identificou numa indústria de filmes plásticos. Por razões técnicas, uma bobina de filme deve ser cortada em duas bobinas intermediárias, de onde serão obtidas as fitas de filme. O modelo que Haessler escreveu para o problema é o seguinte:

$$\text{minimizar} \quad \sum_{j=1}^{p_1} x_j \qquad (15.1)$$

sujeito a:

$$\sum_{j=1}^{p} a_{ij} x_j = d_i, \; i = 1, \ldots, m \qquad (15.2)$$

$$x_j \geq 0 \text{ e inteiro}, \; j = 1, \ldots, p \qquad (15.3)$$

No modelo acima x_j é o número de vezes que o padrão j será utilizado, a_{ij} é o número de vezes que o item i aparece no padrão j e d_i é a demanda do item i.

A largura de um item i é dada por ℓ_i, das bobinas por L e a largura máxima admissível para a bobina intermediária k é L_k. Um padrão de corte $j = 1, \ldots, p$ será considerado viável quando:

$$\sum_{i,k} a_{ij}^k \ell_i \leq L \qquad (16.1)$$

$$\sum_{i} a_{ij}^k \ell_i \leq L_k, \text{ para todo } k \qquad (16.2)$$

$$a_{ij}^k \geq 0 \text{ e inteiro, para todo } i \text{ e } k \qquad (16.3)$$

O autor descreve um procedimento heurístico para gerar os padrões de corte do problema e também considera as modificações introduzidas por ele (HAESSLER, 1975, 1980) no modelo de Geração de Colunas de GILMORE e GOMORY (1961, 1963).

Observe que (16.1 – 16.3) não modela o Problema da Mochila Compartimentada descrito resumidamente na página 4 da introdução desta tese, e mais bem detalhado no capítulo 4.

Para o PCBA, FERREIRA *et al.* (1990) descrevem um procedimento heurístico que consiste dos seguintes passos:

Heurística (Ferreira-Neves-Castro)

1. Defina o fator 1 (demanda atual multiplicada pela respectiva largura) para cada item, e ordene estes fatores em ordem não crescente.
2. Escolha um subconjunto de itens (5 a 9 itens são os valores típicos) segundo o fator 1, e construa de 3 bobinas intermediárias.
3. Ordene a lista de itens escolhidos no passo 2 de forma não crescente, segundo suas larguras.
4. Gere padrões de corte segundo a heurística FFD.
5. Verifique se padrões viáveis (aqueles que não produzem excesso de demanda) foram gerados. Caso contrário, incremente o número de bobinas intermediárias e a lista de itens escolhidos no passo 2. Retorne ao passo 3.
6. Defina o fator 2 (combinar alto uso com baixa perda) para cada padrão e selecione o "melhor" padrão viável.

7. Verifique se o padrão selecionado no passo 6 é "aceitável" (considerar o agrupamento dos itens e a definição das bobinas intermediárias). Caso contrário, faça esta verificação para o próximo melhor padrão viável. Se nenhum padrão aceitável foi encontrado, incremente o número de bobinas intermediárias e a lista de itens escolhidos no passo 2. Retorne ao passo 3.
8. Use o padrão selecionado e atualize as demandas. Se houver demanda remanescente retorne ao passo 1, caso contrário PARE.

Os autores relatam que bons padrões são gerados por meio deste procedimento, entretanto, muitos padrões podem ser eventualmente descartados após sua construção, como descrevem os passos 5 e 7. Entendemos que é fundamental a construção de padrões que efetivamente sejam aceitáveis para uso.

PEREIRA (1993) descreve para o PCBA com a formulação de um programa linear inteiro que tenta descrever os compartimentos dos padrões, porém, computacionalmente inviável. Um *branch-and-bound* foi experimentado para uma relaxação do modelo, entretanto, o autor relata que os resultados foram insatisfatórios.

HOTO (1996) e HOTO e ARENALES (1996, 1997) adotaram um procedimento heurístico um pouco mais elaborado que o de Ferreira-Neves-Castro, procurando gerar padrões compartimentados viáveis, vejamos os passos:

Heurística (Hoto-Arenales)

1. Efetue o agrupamento dos itens compatíveis entre si, gerando grupos N_k de itens.

   Defina a demanda inicial $\tilde{d}_i = d_i$, $i \in N_k$, $k=1, \ldots, K$.

2. Para cada item de um agrupamento N_k defina sua "eficiência" por:

$$\lambda_{ik} = \left(\tilde{d}_i \ell_i\right)\left(\frac{e_k}{\min\limits_r \left\{E_r \mid e_k \le E_r\right\}}\right) \quad \begin{matrix} i \in N_k \\ r=1, \ldots, n \\ k=1, \ldots, K \end{matrix}$$

   Onde $\tilde{d}_i$ é a demanda atual do item i, ℓ_i é sua largura, e_k é a espessura do aço dos itens do grupo N_k e E_r é a espessura do aço da bobina r.

3. Selecione o grupo N_s, tal que $\gamma_s = \max_{k=1}^{K}\left\{\gamma_k = \max_{i \in N_k}\{\lambda_{ki}\}, \text{ para cada } k\right\}$.

4. Calcule P_{med} (média aritmética dos "pesos" das bobinas do estoque), considere NF o número máximo admissível de facas, L_{util} a largura útil de uma bobina (largura da bobina menos as larguras das bobinas intermediárias já construídas nela) e L_{max} a largura máxima admissível para uma bobina intermediária que será construída pela resolução do seguinte problema da mochila restrito, onde a_i é o número de itens do tipo i:

$$\text{maximizar} \sum_{i \in N_s} \lambda_{si} a_i$$

sujeito a:

$$\sum_{i \in N_s} \ell_i a_i \leq L_{max}$$

$$\sum_{i \in N_s} a_i \leq NF - 1$$

$$0 \leq a_i \leq \frac{L_{util}}{P_{med}} \frac{\tilde{d}_i}{\ell_i}, \; i \in N_s$$

a_i inteiro, $i \in N_s$

Se um padrão foi construído vá ao passo 5, caso contrário retorne ao passo 2.

5. Selecione bobinas do estoque de modo a não exceder a demanda dos itens do padrão construído no passo anterior.
6. Atualize as demandas. Se existe demanda remanescente retorne ao passo 2, caso contrário PARE.

A geração dos padrões no procedimento heurístico de Hoto-Arenales é mais elaborada que a de Ferreira-Neves-Castro, pois, os compartimentos (bobinas intermediárias) são construídos por meio de um problema da mochila restrito, além do mais, cada padrão construído é efetivamente utilizável.

CARVALHO (1991) e CARVALHO e RODRIGUES (1994, 1995) resolvem o PCBA pela Técnica de Geração de Colunas, apêndice B. Os autores encontram a solução continua do problema, objetivando a minimização do número de bobinas intermediárias. Em seguida, a solução contínua é arredondada para baixo e a heurística FFD é utilizada para completar a demanda residual. O modelo matemático dos autores é o seguinte:

$$\text{minimizar} \quad c_{perda}\sum_{j\in J} T_j x_j + c_{lam}\sum_{j\in J}\eta_j x_j + c_1\sum_{j\in J}\delta(x_j) + c_2\,\eta \qquad (17.1)$$

sujeito a:

$$d_i^{min} \le \sum_{j\in J} a_{ij}x_j \le d_i^{max},\ i=1,\ldots,n \qquad (17.2)$$

$$x_j \ge 0 \text{ e inteiro}, j\in J \qquad (17.3)$$

No modelo acima o conjunto J é formado pelos índices de padrões de corte válidos em duas fases. O parâmetro c_{perda} é o custo associado a perda de material T_j produzida pelo padrão j, c_{lam} é o custo associado ao processo de laminação, η_j é o número de bobinas intermediárias no padrão j, c_1 é o custo associado ao acerto de um padrão na primeira fase de corte e $\delta(x_j)$ têm valor unitário se o padrão j é utilizado e nulo no caso contrário.

Por fim, c_2 é o custo associado ao acerto na segunda fase de corte e onde η é o número total de bobinas intermediárias, concluindo d_i^{min} e d_i^{max} são respectivamente as demandas mínima e máxima de cada item $i=1,\ldots,m$.

Para gerar uma coluna, analogamente definir um padrão para o corte em duas fases, Carvalho e Rodrigues resolvem o seguinte problema da mochila:

$$\text{maximizar} \quad \sum_{i=1}^{n}\sum_{k=1}^{n_i} u_{ik}\,y_{ik} \qquad (18.1)$$

sujeito a:

$$\sum_{i=1}^{m}\sum_{k=1}^{n_i} L_{ik}\,y_{ik} \le L \qquad (18.2)$$

$$y_{ik} \ge 0 \text{ e inteiro}, i=1,\ldots,n,\ k=1,\ldots,n_i \qquad (18.3)$$

Na mochila (18.1 – 18.3), para a bobina intermediária $k = 1,\ldots, n_i$ definida pelo item i: u_{ik} é o custo relativo obtido por meio dos multiplicadores simplex (veja apêndice B), y_{ik} é o número de vezes que a bobina intermediária aparece no padrão, $L_{ik} = \ell_i a_{ik}$ é a largura da bobina intermediária k definida pelo item i (cada bobina intermediária é definida por apenas um tipo de item), onde ℓ_i é a largura do item i e a_{ik} é o número de itens tipo i na bobina intermediária k , finalmente L é a largura de uma bobina do estoque.

Para resolver a mochila anterior é preciso que as bobinas intermediárias estejam todas definidas. A proposta dos autores foi construir para cada item i todas as bobinas intermediárias homogêneas (compostas por apenas um tipo de item). Exemplificaremos esta idéia com um exemplo ilustrativo, onde as bobinas do estoque possuem largura de 30 unidades, e as bobinas intermediárias devem ter suas larguras entre 10 unidades e 15 unidades. Os dados dos itens estão na tabela 1.

	Agrupamento 1		**Agrupamento 2**	
Item (i)	1	2	3	4
Utilidade (u_i)	31	17	65	30
Largura (ℓ_i)	6	10	11	4

Tabela 1 – Dados dos itens agrupados em dois subconjuntos.

Assim, segundo a proposta de CARVALHO e RODRIGUES (1994, 1995) serão examinadas apenas as seguintes bobinas intermediárias:

1) Para o item 1, uma bobina intermediária de largura $L_{11} = 2*6 = 12$ e utilidade $u_{11} = 62$.
2) Para o item 2, uma bobina intermediária de largura $L_{21} = 1*10 = 10$ e utilidade $u_{21} = 17$.
3) Para o item 3, uma bobina intermediária de largura $L_{31} = 1*11 = 11$ e utilidade $u_{31} = 65$.
4) Para o item 4, uma bobina intermediária de largura $L_{41} = 3*4 = 12$ e utilidade $u_{41} = 90$.

Resolvendo a mochila (18.1 – 18.3) o padrão encontrado é composto por duas unidades da bobina intermediária de largura $L_{41} = 3*4 = 12$, somando uma utilidade de 180 unidades e uma perda de 6 unidades.

Neste mesmo exemplo, entretanto, observamos que esta estratégia não considera padrões mais expressivos, por exemplo, os dois itens do agrupamento 2 podem ser combinados para formar uma bobina intermediária de largura 15, e que pode ser repetida 2 vezes na largura da bobina do estoque.

O padrão compartimentado que acabamos de descrever fornece uma utilidade de 190 unidades e perda nula, sendo melhor que aquele gerado pela proposta de Carvalho e Rodrigues.

Na figura 10 está ilustrado o padrão gerado pelo modelo (18.1 – 18.3) de Carvalho e Rodrigues e o padrão que é descartado pelo mesmo modelo.

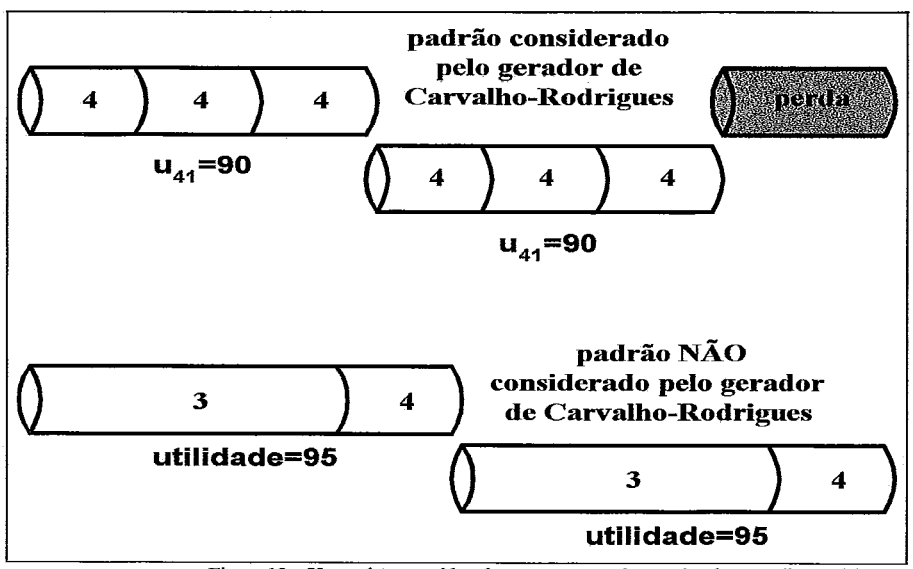

Figura 10 – Um padrão considerado e outro não pelo gerador de Carvalho-Rodrigues.

Concluindo, fica claro, a partir de um exemplo bastante simplificado, que a mochila (18.1 – 18.3) não caracteriza todos os padrões compartimentados, pois, o espaço de busca foi reduzido. No capítulo 4 apresentamos um modelo que caracteriza efetivamente todos os padrões de corte compartimentados e propomos um método para resolvê-lo, além de fornecermos uma estratégia alternativa.

HOTO *et al.* (1998) sugerem a formulação do PCBA como um problema de corte 1.5-dimensional, onde as bobinas do estoque podem ser fracionadas, ou seja, mais de um padrão é utilizado numa mesma bobina para evitar demanda em excesso. A figura 11 ilustra como se processaria o corte de uma bobina, onde x_j^r é a variável que terá o papel de medir a quantidade em mm a ser cortada da bobina r, segundo o padrão de corte j, e a_{ij}^r é o número de fitas do tipo i obtidas da bobina r, segundo o padrão de corte j.

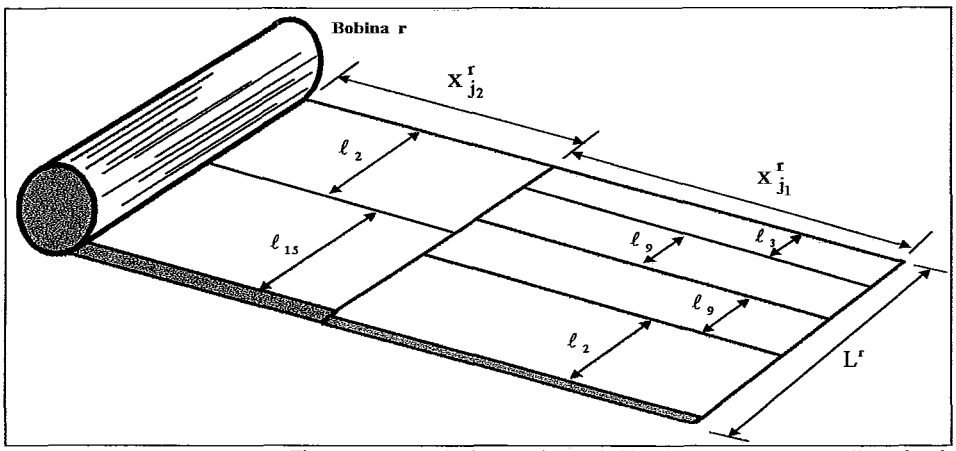

Figura 11 - Exemplo do corte de uma bobina de aço para o caso 1.5-dimensional.

Ainda consideraremos d_i a demanda em Kg da fita i, ℓ_i sua largura em mm, P^r o peso em Kg da bobina r, L^r a largura em mm da bobina r, C^r o comprimento em mm da bobina r, veja figura 12, e finalmente $\mu^r = \frac{P^r}{L^r C^r}$ sua densidade superficial.

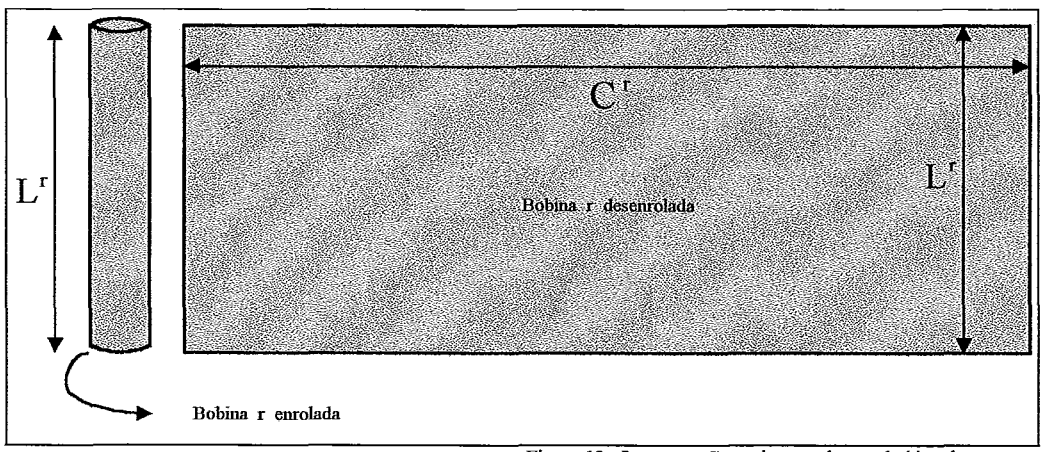

Figura 12 - Largura e Comprimento de uma bobina de aço.

O modelo 1.5-dimensional de HOTO *et al.* (1998) para o PCBA tem o seguinte aspecto:

$$\text{minimizar} \quad \sum_{r=1}^{m}\sum_{j=1}^{p_r} c_j^r x_j^r \qquad (19.1)$$

sujeito a:

$$\sum_{r=1}^{m}\sum_{j=1}^{p_r} \mu^r \ell_i a_{ij}^r x_j^r = d_i\,,\ i=1,\ldots,n \qquad (19.2)$$

$$\sum_{j=1}^{p} x_j^r \le C^r\,,\ r=1,\ldots,m \qquad (19.3)$$

$$x_j^r \ge 0\,,\ r=1,\ldots,m\,,\ j=1,\ldots,p_r \qquad (19.4)$$

No modelo acima em c_j^r estão agregados custos por perdas de aço (em Kg) e custos operacionais associados às bobinas intermediárias. Com esta proposta os autores eliminaram a questão do excesso de produção, entretanto, fitas de diâmetro muito reduzido são geradas o que inviabiliza a utilização delas na máquina de tubos.

MARQUES (2000) e MARQUES e ARENALES (1999, 2001) resolvem o PCBA por meio de uma heurística de repetição exaustiva (HINXMAN, 1980) que consiste em:

Heurística Gulosa de Repetição Exaustiva

Passo 1
Gere o melhor padrão de corte que não exceda a demanda.

Passo 2
Utilize este padrão o máximo de vezes possível.
Se não for possível usar o padrão PARE.

Passo 3
Atualize a demanda, se houver demanda remanescente retorne ao passo 1.

Para gerar um padrão compartimentado, os autores propuseram três heurísticas que serão descritas no capítulo 4.

3 – Uma nova estratégia para resolver o PCBA

A estratégia que adotamos para resolver o PCBA é a Técnica da Geração de Colunas de GILMORE e GOMORY (1961, 1963) descrita no apêndice B, e nosso objetivo será o de minimizar as perdas de aço. O modelo matemático que adotamos é apresentado a seguir:

$$\text{minimizar} \quad \sum_{r=1}^{m}\sum_{j=1}^{p_r} c_j^r x_j^r \qquad (20.1)$$

sujeito a:

$$\sum_{r=1}^{m}\sum_{j=1}^{p_r} \frac{P^r}{L^r} \ell_i a_{ij}^r x_j^r = d_i, \; i = 1, \ldots, n \qquad (20.2)$$

$$x_j^r \geq 0 \text{ , e inteiro } r = 1, \ldots, m, \; j = 1, \ldots, p_r \qquad (20.3)$$

No modelo (20.1 – 20.3) a variável de decisão x_j^r representa o número de vezes que o padrão compartimentado j é utilizado para cortar bobinas do tipo r. Os demais dados são os mesmos descritos para a abordagem 1.5-dimensional.

Até aqui não há nada de realmente novo na abordagem que escolhemos, mas, um dos pontos centrais, talvez o fundamental, na Técnica de Geração de Colunas é justamente a maneira como um padrão é gerado.

Fitas	Largura	Espessura Inicial	Espessura Final	Tipos de Aço
tipo 1	200 mm	2,00 mm	1,10 mm	*SAE* 1008
tipo 2	230 mm	2,00 mm	1,10 mm	*SAE* 1008 ou *SAE* 1010
tipo 3	90 mm	2,00 mm	1,10 mm	*SAE* 1008
tipo 4	115 mm	2,00 mm	1,20 mm	*SAE* 1008
tipo 5	105 mm	2,00 mm	1,10 mm	*SAE* 1010
tipo 6	105 mm	1,50 mm	0,90 mm	*SAE* 1010 ou *SAE* 1012
tipo 7	200mm	1,50 mm	0,90 mm	*SAE* 1012
tipo 8	120 mm	1,50 mm	0,90 mm	*SAE* 1010 ou *SAE* 1012
tipo 9	90 mm	2,00 mm	2,00 mm	*SAE* 1008
tipo 10	70 mm	2,00 mm	2,00 mm	*SAE* 1008
tipo 11	65 mm	2,00 mm	2,00 mm	*SAE* 1008 ou *SAE* 1010
tipo 12	105 mm	2,00 mm	2,00 mm	*SAE* 1008
tipo 13	135 mm	1,00 mm	1,00 mm	*SAE* 1010 ou *SAE* 1012
tipo 14	205 mm	1,00 mm	1,00 mm	*SAE* 1010
tipo 15	125 mm	1,00 mm	1,00 mm	*SAE* 1010
tipo 16	115 mm	1,00 mm	1,00 mm	*SAE* 1010
tipo 17	180 mm	1,00 mm	1,00 mm	*SAE* 1010
tipo 18	53 mm	1,50 mm	1,50 mm	*SAE* 1010 ou *SAE* 1012
tipo 19	105 mm	1,50 mm	1,50 mm	*SAE* 1012
tipo 20	90 mm	1,50 mm	1,50 mm	*SAE* 1010 ou *SAE* 1012

Tabela 2 – Características de um subconjunto de fitas (extraídas de dados reais).

Na tabela 2 ilustramos dados reais de um conjunto de fitas. A largura e o tipo de aço de cada fita, assim como a redução da espessura do aço por meio da laminação, são previamente estabelecidos em sintonia com o tipo de tubo que deverá ser confeccionado, e como mencionamos anteriormente isso constitui uma exigência tecnológica.

Assim, cada fita apresenta uma espessura inicial (espessura da bobina em estoque de onde ela será cortada) e uma espessura final (espessura final da fita). Quando um tipo de tubo não requer o processo de laminação do aço, a fita associada apresenta espessuras inicial e final iguais. Em alguns casos específicos, certos tubos podem admitir mais de um tipo de redução da espessura do aço que o constitui, isto é, o tipo de fita associada pode apresentar mais de uma espessura inicial ou final.

Para construirmos um padrão compartimentado, consideraremos agrupamentos entre fitas compatíveis (fitas que possuem as mesmas reduções de espessuras, e cujo tipo de aço sejam equivalentes).

Seja G(V, E) o seguinte grafo: cada vértice de V representará uma fita e cada aresta de E é tal que, se uma fita pode ser agrupada a outra (segundo suas compatibilidades), existe uma aresta entre os vértices correspondentes, figura 13.

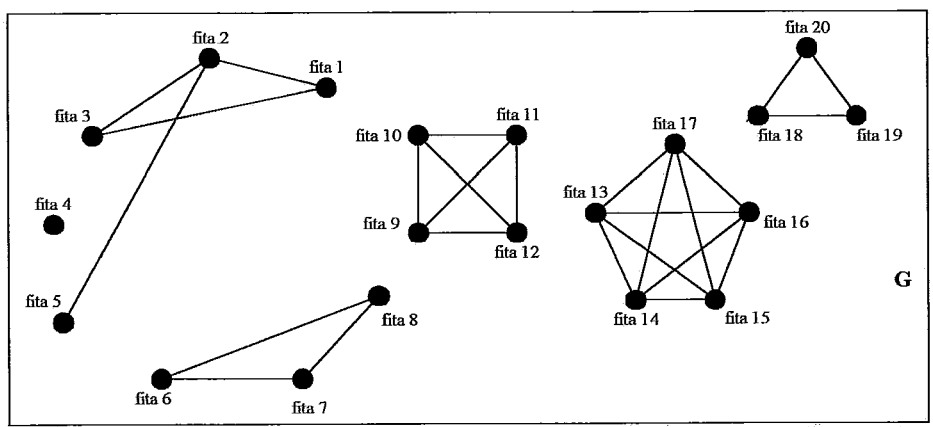

Figura 13 - Grafo do agrupamento das fitas da tabela 2.

Considere $N=\{1, 2, ..., 20\}$ o conjunto dos índices das fitas e observe pelo grafo G que aquelas a serem laminadas podem ser agrupadas em quatro subconjuntos: $N_1=\{1, 2, 3\}$, $N_2=\{4\}$, $N_3=\{5\}$ e $N_4=\{6, 7, 8\}$.

Note que $N_1 = \{1, 3\}$, $N_2 = \{2, 5\}$, $N_3 = \{4\}$ e $N_4 = \{6, 7, 8\}$ é um outro agrupamento e que cada um destes subconjuntos é uma clique ou um vértice isolado de G. Ainda observando o grafo G notamos que as fitas que não serão laminadas podem ser agrupadas em três subconjuntos (novamente são cliques ou vértices isolados): $N_5 = \{9, 10, 11, 12\}$, $N_6 = \{13, 14, 15, 16, 17\}$ e $N_7 = \{18, 19, 20\}$.

Em geral a definição dos agrupamentos não é única, porém, uma vez definidos os agrupamentos, os padrões compartimentados são construídos por meio de uma Mochila Compartimentada, cujos detalhes de modelagem e procedimentos computacionais encontram-se no capítulo 4.

Determinar uma solução ótima do modelo (20.1 – 20.3) não é uma tarefa evidente, pois, trata-se de um Problema de Programação Linear Inteiro. No capítulo anterior apresentamos algumas das técnicas que vêm sendo exploradas no sentido de encontrar soluções inteiras. A heurística que adotamos para obter uma solução inteira é a seguinte:

Heurística para obter uma solução Inteira do PCBA

1. Resolver a relaxação contínua do Problema (20.1 – 20.3);
2. Arredondar para baixo (truncar) a solução contínua obtida e armazenar a solução inteira obtida pelo arredondamento;
3. Definir a demanda residual (demanda remanescente) e um novo Programa Linear Inteiro, que denominamos Problema Residual;
4. Resolver a relaxação contínua do Problema Residual (gerar padrões restritos pela nova demanda);
5. Arredondar para baixo (truncar) a solução residual contínua obtida;
6. Se a solução residual contínua NÃO for nula armazene a solução inteira obtida pelo arredondamento (agregue esta solução inteira naquela já obtida de problemas anteriores, pois, novos padrões podem surgir) e retorne ao passo 3. Caso contrário, se ainda houver demanda residual, gere padrões compartimentados restritos a esta demanda.

Neste capítulo apresentamos as características técnicas do PCBA e revisamos os procedimentos que já foram empregados na sua resolução. Vimos que nenhum deles garante uma solução ótima para o problema. Observamos que problemas de corte em duas fases são pouco explorados na literatura, e que suas dificuldades são análogas às de problemas em uma fase, entretanto, a definição dos padrões de corte é mais elaborada.

Descrevemos a estratégia que utilizamos para resolver o PCBA, sobre tudo o procedimento heurístico que utilizamos para obter uma solução inteira. Deixamos em aberto a maneira como geramos um padrão de corte compartimentado, pois, os detalhes matemáticos e experimentais deste mecanismo estão descritos no capítulo 4.

A condição humana é uma lavoura, enquanto o preconceito uma erva daninha.
Há pessoas que confundem direitos com excesso de ética e deveres com a falta.

Capítulo 3

Problemas Clássicos da Mochila

1 - Introdução

Os Problemas da Mochila definem uma classe de problemas da programação linear inteira e são classificados na literatura, segundo a sua complexidade de resolução, como problemas NP-árduos (GAREY e JOHNSON, 1979) e (MARTELLO e TOTH, 1990). Inúmeros problemas de importância reconhecida têm uma mochila como subproblema, um exemplo típico é o procedimento de geração de colunas no contexto corte e empacotamento, portanto, eles também são problemas NP-árduos (veja resumo sobre complexidade no apêndice C).

A idéia básica de um problema da mochila consiste na escolha de um subconjunto de itens, cada qual com uma correspondente utilidade e um peso que define o quanto este item utilizará da capacidade da mochila. Como ilustração, imagine um alpinista que ao escalar uma montanha precisa organizar sua mochila, selecionando itens de seu interesse, de modo que, a soma da utilidade de sua mochila seja máxima, ou um muambeiro que tem uma cota fixa a ser gasta e precisa decidir sobre a escolha de itens que lhe proporcione o melhor lucro, ou ainda na escolha de pequenos pedaços de ferro a serem cortados de uma barra de ferro maior, de modo que, a perda de material seja mínima.

Neste capítulo apresentamos um resumo de problemas da mochila comumente encontrados na literatura, e quase todas elas podem ser mais bem estudas em (MARTELLO e TOTH, 1990) , (PISINGER, 1995) e (ARENALES e editores, 1997). Em casos mais específicos é feita a referência do respectivo trabalho.

Problemas da mochila tem sido o alvo de intensos estudos de vários pesquisadores por suas aplicações, bem como por interesse teórico. Diferentes tipos de mochilas podem ocorrer, de modo que, não seria inconveniente utilizarmos o termo "família de mochilas", destacamos alguns tipos:

- Problema da Mochila 0-1;
- Problema da Mochila Restrita;
- Problema da Mochila Irrestrita;
- Problema da Soma de Subconjuntos (uma particular Mochila 0-1);
- Problema da Designação Generalizada;
- Problema da Múltipla Escolha;
- Problema com Múltiplas Mochilas;
- Problema da Mochila com Coeficientes Variáveis;
- Problema da Mochila Encapsulada;
- Problema do Troco e o *Bin-Packing*;

2 - O Problema da Mochila 0-1

Considere uma coleção de itens indexados pelo conjunto $N=\{1, 2, \ldots, n\}$. A cada item está associado um "valor de utilidade" $u_i \in \mathbb{Z}_+$, $i \in N$, e um "peso" $p_i \in \mathbb{Z}_+$, $i \in N$, que representa o quanto este item utilizará da capacidade $c \in \mathbb{Z}_+^*$ da mochila. Seja $a_i = 1$ se o item i foi escolhido para compor a mochila, e $a_i = 0$ no caso contrário. O problema pode ser escrito como:

$$\text{maximizar} \quad \sum_{i=1}^{n} u_i a_i \qquad (21.1)$$

sujeito a:

$$\sum_{i=1}^{n} p_i a_i \leq c \qquad (21.2)$$

$$a_i \in \{0, 1\}, \; i=1, \ldots, n \qquad (21.3)$$

Para que a solução do problema não seja trivial vamos admitir que $\sum_{i=1}^{n} p_i > c$ e que $p_i \leq c$, $i \in N$. Considere que os itens estejam ordenados de forma não crescente segundo a razão das utilidades pelos respectivos pesos, e determine o primeiro item que ao compor a mochila, exceda a capacidade da mesma, ou seja:

$$\frac{u_1}{p_1} \geq \frac{u_2}{p_2} \geq \cdots \geq \frac{u_n}{p_n} \quad \text{e encontre k, tal que, } k = \min\left\{ j \in N \left| \sum_{i=1}^{j} p_i > c \right. \right\}$$

O item k é denominado "item crítico". DANTZIG (1957), num trabalho pioneiro, sugere um algoritmo guloso para a resolução do problema, ele verifica que $\sum_{i=1}^{k} u_i + \left\lfloor \left(c - \sum_{i=1}^{k} p_i \right) \frac{u_{k+1}}{p_{k+1}} \right\rfloor$ é um limitante superior, onde $\lfloor x \rfloor$ é a função maior inteiro menor ou igual a x (função piso). Dantzig observou que uma solução ótima da relaxação contínua é dada por $a_i = 1 \; (i = 1, \ldots, k-1)$, $a_k = \frac{c - \sum_{i=1}^{k-1} p_i}{p_k}$ e $a_i = 0 \; (i = k+1, \ldots, n)$.

Desde então muitos pesquisadores têm dedicado estudos a este problema, destacamos os seguintes trabalhos: (FAYARD e PLATEAU, 1975, 1982), onde é feito um estudo comparativo de métodos para resolver problemas da mochila e apresentado um algoritmo, (LAURIÈRE, 1976), onde também é descrito um novo algoritmo, (MACULAN, 1983), onde o autor calcula e testa limitantes por meio da Relaxação Lagrangiana, (VILLELA e BORNSTEIN, 1983, 1987), onde é descrito e testado novos limitantes, (PLATEAU e ELKINEL, 1985), onde é descrito um algoritmo híbrido para o problema, além de muitos outros.

BALAS e ZEMEL (1980) observaram que uma vez resolvida à relaxação contínua, geralmente uma solução ótima inteira do problema envolve a mudança de variáveis próximas àquela associada ao item crítico. Por exemplo, considere o problema a seguir:

$$\begin{aligned} &\text{maximizar } a_1 + 2a_2 + a_3 + 2a_4 + a_5 + a_6 \\ &\text{sujeito a:} \\ &a_1 + 3a_2 + 2a_3 + 5a_4 + 4a_5 + 6a_6 \leq 9 \\ &a_1, a_2, a_3, a_4, a_5, a_6 \in \{0, 1\} \end{aligned}$$

A solução ótima da relaxação contínua é dada por (1, 1, 1, 0.6, 0, 0), o índice do item crítico é 4, e uma solução ótima inteira é dada por (1, 1, 0, 1, 0, 0).

Esta propriedade tem sido investigada e é conhecida na literatura como *core problem*. Para detalhes deste problema recomendamos (MARTELLO e TOTH, 1990, pag 57) e (SOMA e TOTH, 1999).

No apêndice E existe um pseudocódigo de um Algoritmo Guloso que resolve o Problema da Mochila 0-1.

Nos trabalhos de MARTELLO e TOTH (1990) , PISINGER (1995) e ARENALES *et al.* (1997) são encontrados estudos mais detalhados desta modalidade de mochila.

3 - Os Problemas da Mochila Restrita e Irrestrita

Se no caso anterior a questão não for simplesmente a escolha de itens apropriados, mas a determinação de um número limitado de itens, a mochila é denominada restrita (com variáveis canalizadas) e o modelo toma o seguinte aspecto:

$$\text{maximizar} \quad \sum_{i=1}^{n} u_i a_i \qquad (22.1)$$

sujeito a:

$$\sum_{i=1}^{n} p_i a_i \leq c \qquad (22.2)$$

$$0 \leq a_i \leq b_i, \; i=1, \ldots, n \qquad (22.3)$$

$$a_i \text{ inteiro}, \; i=1, \ldots, n \qquad (22.4)$$

São admitidas as seguintes condições para evitar soluções triviais:

$$\sum_{i=1}^{n} b_i p_i > c \qquad (22.5)$$

$$b_i p_i \leq c, \; i=1, \ldots, n \qquad (22.6)$$

A mochila irrestrita ocorre quando $b_i \to +\infty$ para cada $i \in N$, neste caso a condição (22.5) sempre é verdadeira, mas, (22.6) é trocada por $p_i \leq c, \; i=1, \ldots, n$.

No apêndice E existe um pseudocódigo de um Algoritmo Guloso que resolve o Problema da Mochila Restrita, no mesmo apêndice também podem ser encontrados pseudocódigos para o Problema Irrestrito.

MARTELLO e TOTH (1977a) propuseram um eficiente algoritmo que traz várias modificações significativas em relação ao algoritmo proposto por HOROWITZ e SAHNI (1974), veja ARENALES *et al.* (1997) onde são descritos os dois algoritmos. Uma das modificações é o cálculo de limitantes superiores melhores que o Limitante de Dantzig (DANTZIG, 1957). Vejamos como é calculado o Limitante de Martello-Toth.

Sejam: $\overline{c} = c - \left\lfloor \frac{c}{p_1} \right\rfloor p_1$, $c' = \overline{c} - \left\lfloor \frac{\overline{c}}{p_2} \right\rfloor p_2$ e $z' = \left\lfloor \frac{c}{p_1} \right\rfloor u_1 + \left\lfloor \frac{\overline{c}}{p_2} \right\rfloor u_2$.

O limitante Z^{MT} de MARTELLO e TOTH (1990, pag 93) é escrito como:

$$Z^{MT} = \max \{Z^0, Z^1\} \qquad (23)$$

Em (23) $Z^0 = z' + \left\lfloor c' \frac{u_3}{p_3} \right\rfloor$ e $Z^1 = z' + \left\lfloor \left(c' + \left\lceil \frac{p_2 - c'}{p_2} \right\rceil p_1 \right) \frac{u_2}{p_2} - \left\lceil \frac{p_2 - c'}{p_1} \right\rceil u_1 \right\rfloor$,

onde $\lceil \, . \, \rceil$ é a função menor inteiro (função teto).

Em alguns casos existe o interesse por Mochilas Irrestritas, onde a restrição física da mochila é solicitada na igualdade, em outras palavras:

$$\text{maximizar} \quad \sum_{i=1}^{n} u_i a_i \qquad (24.1)$$

sujeito a:

$$\sum_{i=1}^{n} p_i a_i = c \qquad (24.2)$$

$$a_i \geq 0 \text{ e inteiro, } i = 1, \ldots, n \qquad (24.3)$$

Para detalhes sobre mochilas desta natureza recomendamos o trabalho de YANASSE e SOMA (1987), onde é descrito um eficiente algoritmo para resolver o problema (veja apêndice E), e o trabalho de ZHU e BROUGHAN (1998) sobre a agregação de equações diofantinas, além é claro de MARTELLO e TOTH (1990).

Na seção 7 apresentamos um caso particular desta mochila conhecido como Problema do Troco, onde $u_i = -1$, portanto, não mais um inteiro não negativo, mesmo assim, ainda é possível resolver o problema como mostra GLOVER (1965).

4 - Os Problemas da Soma de Subconjuntos, da Designação e da Múltipla Escolha

A soma de subconjuntos consiste simplesmente do caso particular de mochila 0-1 em que $u_i = p_i$, $i \in N$.

Os algoritmos para resolução de mochilas dependem do cálculo de limitantes inferiores e superiores, e estes por sua vez, baseiam-se na existência de ganhos marginais distintos, o que não é o caso desta mochila, pois, as utilidades e pesos são iguais, dificultando o cálculo de limitantes.

Dentro do contexto corte e empacotamento esta particular mochila aparece na resolução de subproblemas.

DIETRICH e ESCUDORE (1989) relatam a determinação de limitantes superiores para o problema da Soma de Subconjuntos. SOMA e TOTH (1999) descrevem um algoritmo com uma nova proposta para a determinação do item crítico, normalmente efetuada nos algoritmos exatos. Para estudos mais detalhados recomendamos (MARTELLO e TOTH, 1990), e em (ARENALES *et al.*, 1997) pode ser obtido um pseudocódigo de um algoritmo que resolve o problema.

Para ilustrar o Problema da Designação Generalizada considere n tarefas a serem designadas para m máquinas, de modo que, u_{ij} será a utilidade da tarefa (do item) i se designado para a máquina (para a mochila) j de capacidade c_j, e p_{ij} representará o quanto a tarefa (item) i ocupará da capacidade c_j da máquina (mochila) j, se a ela for designada. A variável de decisão é $a_{ij} = 1$ se a tarefa (o item) i for designada para a máquina (a mochila) j, e $a_{ij} = 0$ no caso contrário. O modelo é escrito como:

$$\text{maximizar} \quad \sum_{j=1}^{m}\sum_{i=1}^{n} u_{ij}a_{ij} \qquad (25.1)$$

sujeito a:

$$\sum_{i=1}^{n} p_{ij}a_{ij} \le c_j,\ j=1, \ldots, m \qquad (25.2)$$

$$\sum_{j=1}^{m} a_{ij}=1,\ i=1, \ldots, n \qquad (25.3)$$

$$a_{ij} \in \{0, 1\},\ i=1, \ldots, n,\ j=1, \ldots, m \qquad (25.4)$$

Na literatura também pode ser encontrado o caso em que são considerados custos c_{ij} por designação, ao invés de utilidades u_{ij}. Nesta situação, o objetivo (25.1) passa a ser "minimizar $\sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}a_{ij}$", e as restrições permanecem as mesmas. Para mais detalhes a respeito deste problema recomendamos (MARTELLO e TOTH, 1990).

Já o problema da Múltipla Escolha consiste na determinação de exatamente um item de cada um dos subconjuntos que compõem uma partição $\{N_1, N_2, \ldots, N_k\}$ de N, de sorte que a soma das utilidades seja maximizada e a capacidade da mochila respeitada.

O modelo desta mochila é descrito a seguir, onde $a_{ij}=1$ se o item i de N_j foi designado para a mochila j, e $a_{ij}=0$ no caso contrário. Ele pode ser encontrado em (PISINGER, 1995, pag 107) e (MARTELLO e TOTH, 1990, pag 77).

$$\text{maximizar} \quad \sum_{j=1}^{k}\sum_{i \in N_j} u_{ij}a_{ij} \qquad (26.1)$$

sujeito a:

$$\sum_{j=1}^{k}\sum_{i \in N_j} p_{ij}a_{ij} \le c \qquad (26.2)$$

$$\sum_{i \in N_j} a_{ij}=1,\ j=1, \ldots, k \qquad (26.3)$$

$$a_{ij} \in \{0, 1\},\ i \in N_j,\ j=1, \ldots, k \qquad (26.4)$$

5 - O Problema com Múltiplas Mochilas e com Coeficientes Variáveis

Resolver Múltiplas Mochilas consiste em preencher de forma ótima m mochilas $m \leq n$ com capacidades $c_1, c_2, \ldots, c_m$, ou equivalentemente em determinar m subconjuntos de $N = \{1, \ldots, n\}$ disjuntos entre si, de modo que a soma das utilidades seja máxima, admita ainda que cada item é designado a uma única mochila, ou seja, $a_{ij}=1$ se o item i é designado para a mochila j, e $a_{ij}=0$ no caso contrário, onde $j=1, 2, \ldots, m$ e $i \in N$. O modelo para o Problema com Múltiplas Mochilas é escrito como:

$$\text{maximizar} \quad \sum_{j=1}^{m}\sum_{i=1}^{n} u_i a_{ij} \qquad (27.1)$$

sujeito a:

$$\sum_{i=1}^{n} p_j a_{ij} \leq c_j, \; j=1, \ldots, m \qquad (27.2)$$

$$\sum_{j=1}^{m} a_{ij} \leq 1, \; i=1, \ldots, n \qquad (27.3)$$

$$a_{ij} \in \{0, 1\}, \; i=1, \ldots, n \; , \; j=1, \ldots, m \qquad (27.4)$$

São admitidas as seguintes condições para evitar soluções triviais:

$$\exists i \in N \left| \; p_i \leq \max_{j=1}^{m} \{c_j\} \right. \qquad (27.5)$$

$$c_j \geq \min_{i \in N} \{p_i\} \; (j=1, 2, \ldots, m) \qquad (27.6)$$

$$\sum_{i=1}^{n} p_i > c_j \; (j=1, 2, \ldots, m) \qquad (27.7)$$

Uma vez mais recomendamos o trabalho de MARTELLO e TOTH (1990) para maiores detalhes.

Considere agora um problema em que n peças possam ser escolhidas para um simples processo de produção numa máquina, por exemplo, um torno.

O tempo total disponível para processamento é T, seja t_i, i=1, 2, ..., n, o tempo para produzir cada peça, dependente de uma condição v_i do torno (no caso, poderia ser a velocidade com que o torno trabalha). Assuma que o lucro f_i associado a cada peça também depende de v_i.

O problema consiste em selecionar peças, determinando uma condição ótima de trabalho do torno, de modo que o lucro total seja maximizado. Este problema foi estudado por HITOMI (1970) e revisto posteriormente por SUZUKI (1978) que o denomina Problema da Mochila Generalizada (com coeficientes variáveis).

O modelo da Mochila Generalizada apresentado por SUZUKI (1978) tem o seguinte aspecto, onde $a_i = 1$ se a peça i é escolhida, e $a_i = 0$ no caso contrário:

$$\text{maximizar} \quad \sum_{i=1}^{n} f_i(v_i)a_i \qquad (28.1)$$

sujeito a:

$$\sum_{i=1}^{n} g(v_i)a_i \le T \qquad (28.2)$$

$$\alpha_i \le v_i \le \beta_i, \; i=1, \ldots, n \qquad (28.3)$$

$$a_i \in \{0, 1\}, \; i=1, \ldots, n \qquad (28.4)$$

Em seu trabalho, HITOMI (1970) determinou $f_i(v_i)=(A_i - B_i)/(v_i - C_i v_i)$ e $g_i(v_i) = (a_i - b_i)/(v_i - c_i v_i)$, além de considerar a condição (28.3) no seguinte formato:

$$v_i \ge 0, \; i=1, \ldots, n \qquad (29)$$

6 – O Problema da Mochila Encapsulada

Na verdade o problema é conhecido na literatura como *Nested Knapsack Problem*, e sua idéia consiste em construir cápsulas (ou casulos) de tamanhos predefinidos que deverão abrigar itens de interesse. No interior de cada cápsula podem ser construídas novas cápsulas, também de tamanhos predefinidos, e assim sucessivamente. Esta modalidade de mochila, embora muito semelhante à Mochila Compartimentada (capítulo 4), onde são construídos compartimentos, difere substancialmente pelo fato das **cápsulas** terem seus tamanhos **fixos**, ao passo que os **compartimentos** apresentam tamanhos **flexíveis**.

O exemplo a seguir ilustra uma mochila encapsulada: itens devem ser selecionados para serem levados até uma região montanhosa de difícil acesso. No início eles são transportados no vagão de um trem até um ponto em que é possível apenas o transporte rodoviário. Daí por diante, os itens são transportados por meio de pequenos furgões até um outro ponto em que o transporte só pode ser feito por carregadores. A carga de cada carregador deve ser proveniente de um (e apenas um) dos furgões. Em cada sistema de transporte (via trem, furgão e carregador) existe uma capacidade de carga e custo de transporte associado. O problema consiste em selecionar itens para as respectivas mochilas em cada estágio, de modo que, seja maximizada a utilidade dos itens menos os custos de transporte.

Essencialmente, o problema descrito representa uma situação em que os itens selecionados no primeiro estágio (transporte via trem) devem ser novamente selecionados nos estágios subseqüentes (transporte via furgões e carregadores), e ainda mais, todo item selecionado para a mochila do primeiro estágio deve ser incluído em uma (e somente uma) mochila de cada um dos estágios subseqüentes.

JOHNSTON e KHAN (1995) estudam esta mochila aplicada em problemas de corte e empacotamento e em problemas de confecção de cabos de fibra ótica. Os autores descrevem limitantes superiores e apresentam um modelo desta modalidade de mochila, cujas variáveis de decisão são: x_{ij}^{s} que é igual a 1 se o item i foi designado para a mochila j no estágio s, e igual 0 no caso contrário, e y_{j}^{s} que é igual a 1 se a mochila j é utilizada no estágio s, e igual a 0 no caso contrário, onde $i, j = 1, \ldots, m$ e $s = 1, \ldots, k$. Também são considerados os parâmetros γ^{s} e c^{s} que representam, respectivamente, o custo e a capacidade das mochilas do estágio $s = 1, \ldots, k$.

O modelo apresentado pelos autores tem o seguinte aspecto:

$$\text{maximizar} \quad \sum_{i=1}^{m} u_i x_{i1}^1 - \sum_{j=1}^{m}\sum_{s=2}^{k} \gamma^s y_j^s \qquad (30.1)$$

sujeito a:

$$\sum_{i=1}^{m} p_i x_{i1}^1 \le c \qquad (30.2)$$

$$\sum_{i=1}^{m} p_i x_{ij}^s \le c^s y_j^s,\ j = 1, \ldots, m,\ s = 2, \ldots, k \qquad (30.3)$$

$$x_{i1}^1 \le \sum_{j=1}^{m} x_{ij}^s,\ i = 1, \ldots, m,\ s = 1, \ldots, k \qquad (30.4)$$

$$\max_j (x_{pj}^s + x_{qj}^s) \le \max_j (x_{pj}^{s-1} + x_{qj}^{s-1}) \qquad (30.5)$$

para $s = 3, \ldots, k$ e $p, q = 1, \ldots, m,\ p \ne q$

$$x_{ij}^s,\ y_j^s \in \{0, 1\},\ i, j = 1, \ldots, m,\ s = 1, \ldots, k \qquad (30.6)$$

7 – O Problema do Troco e o *Bin-packing*

O Problema do Troco pode ser ilustrado pela situação em que o caixa de um estabelecimento comercial precisa devolver um troco de valor c, usando o menor número de moedas disponíveis em número limitado para cada tipo.

O modelo do Problema do Troco é escrito como:

$$\text{minimizar} \quad \sum_{i=1}^{n} a_i \qquad (31.1)$$

sujeito a:

$$\sum_{i=1}^{n} p_i a_i = c \qquad (31.2)$$

$$0 \le a_i \le b_i,\ i = 1, \ldots, n \qquad (31.3)$$

$$a_i \text{ inteiro},\ i = 1, \ldots, n \qquad (31.4)$$

Aqui também são admitidas as condições (22.5) e (22.6). Observe que o problema é um caso particular da mochila (24.1 – 24.3) da seção 3. Este problema pode ser mais bem estudado em (MARTELLO e TOTH, 1990, pag 221).

O *Bin-Packing*, descrito na seção 6 do capítulo 1 desta tese, pode ser modelado com a terminologia dos problemas da mochila. Cada item deve ser designado a uma mochila (*bin*) de capacidade c, de um total de n mochilas, visando utilizar o menor número de mochilas. As variáveis de decisão são $a_{ij}=1$ se o item i for designado para a mochila (*bin*) j, e $a_{ij}=0$ no caso contrário, e também $y_j=1$ se a mochila (*bin*) foi utilizada, e $y_j=0$ no caso contrário, onde $i, j \in N$. O modelo do *Bin-packing* tem o seguinte formato:

$$\text{minimizar} \quad \sum_{j=1}^{n} y_j \qquad (32.1)$$

sujeito a:

$$\sum_{i=1}^{n} p_i a_{ij} \leq c y_j, \; j=1,\ldots,n \qquad (32.2)$$

$$\sum_{j=1}^{n} a_{ij} = 1, \; i=1,\ldots,n \qquad (32.3)$$

$$a_{ij}, y_j \in \{0,1\}, \; i,j=1,\ldots,n \qquad (32.4)$$

8 – Considerações Finais

O emprego da Programação Dinâmica foi intensamente estudado em Problemas da Mochila, destacamos os trabalhos de BELLMAN (1957, 1962) e também de GILMORE e GOMORY (1966). KOLESAR (1967) efetuou experimentos com *branch-and-bound* e desde então o método foi bastante explorado.

No trabalho de HOROWITZ e SAHNI (1974) os autores descrevem que problemas da mochila são separáveis no sentido de que os itens podem ser divididos em dois subconjuntos, de modo que, em cada um deles todas as soluções viáveis podem ser enumeradas e posteriormente agregadas. Eles descrevem um algoritmo que efetua enumeração implícita, posteriormente melhorado por MARTELLO e TOTH (1977a), conforme já mencionamos. Esta propriedade abriu a perspectiva da utilização de processamento paralelo e resultados neste sentido foram obtidos por KINDERVATER e LENSTRA (1986). Recentemente SANCHES e SOMA (2001) apresentaram um algoritmo que é uma paralelização ótima e adaptativa do algoritmo de HOROWITZ e SAHNI (1974).

Destacamos algumas outras importantes contribuições, entre muitas mais de igual relevância que não são citadas aqui:

- A busca em profundidade sugerida por TARJAN (1972);

- O algoritmo de redução, onde é possível fixar o valor de algumas variáveis, sugerido por INGARGIOLA e KORSH (1973);

- A heurística polinomial de SOMA *et al.* (1995);

- Estudos relativos à complexidade relatados por PISINGER (1995);

- O algoritmo baseado numa estratégia de agregação proposto por BABAYEV *et al.* (1997);

- Novas modalidades de mochila, por exemplo, HITOMI (1970), JOHNSTON e KHAN (1995) e HOTO *et al.* (2001);

- O algoritmo para as k-melhores soluções de um problema da mochila, sugerido por YANASSE *et al.* (2000) com aplicações em cortes bidimensionais;

- Os trabalhos de Martello-Toth que são referências singulares para qualquer estudo que se refere ao assunto, destacamos (MARTELLO e TOTH, 1977a, 1990).

Esperamos com este capítulo ter apresentado de forma simples e concisa as principais modalidades de mochila que podem ser encontradas na literatura. O próximo capítulo tratará de uma nova modalidade de mochila que identificamos ao estudar o PCBA.

Na matemática o infinito é um Deus que aparece e é descrito de várias formas.

Capítulo 4

O Problema da Mochila Compartimentada

1 - Introdução

O problema de se construir compartimentos numa mochila tem aparecido em alguns problemas de corte como já destacamos no capítulo 2, porém, em nenhum daqueles trabalhos sua formulação matemática foi efetivamente apresentada. Neste capítulo apresentaremos o modelo do Problema da Mochila Compartimentada e um procedimento computacional capaz de resolvê-lo. Vale ressaltar que esta modalidade de mochila não aparece nos trabalhos de MARTELLO e TOTH (1990) e LIN (1998).

Em trabalho a ser publicado, HOTO *et al.* (2001) apresentam a formulação matemática do problema e discutem um procedimento que examina todas as soluções do problema, além de proporem uma heurística.

Uma formulação de um caso particular do problema, apresentada na seção 6 deste capítulo, é discutida por HOTO *et al.* (1999). Recentemente, MARQUES (2000) e MARQUES e ARENALES (2001) apresentaram alguns procedimentos heurísticos para o caso restrito que também descreveremos neste capítulo.

Vejamos um exemplo que ilustra a Compartimentação de uma Mochila:

Exemplo Considere uma mochila de capacidade igual a 21, que deve ser preenchida com itens da tabela 3, tal que as combinações lineares dos pesos dos itens do agrupamento 1 devem ter valor entre 5 e 15, já as provenientes do agrupamento 2 entre 6 e 12.

	Agrupamento 1			**Agrupamento 2**		
Item (i)	1	2	3	4	5	6
Utilidade (u_i)	9	11	8	5	6	8
Peso (p_i)	5	9	7	3	6	10

Tabela 3 – Dados dos itens agrupados em dois subconjuntos.

Observe que os itens estão agrupados em 2 subconjuntos, pois, os itens 1, 2 e 3 não devem ser misturados com os itens 4, 5 e 6. Resumindo, o preenchimento da mochila deverá obedecer as seguintes condições:

Condição 1: $5 \leq \sum_{i=1}^{3} p_i a_i \leq 15$ e $6 \leq \sum_{i=4}^{6} p_i a_i \leq 12$

Condição 2: $\sum_{i=1}^{6} p_i a_i y_j \leq 21$, onde y_j é o número de freqüência do compartimento j.

A condição 1 refere-se à construção de "compartimentos no interior da mochila", cujas capacidades variam de 5 a 15 para itens combinados do primeiro agrupamento, e 6 a 12 para itens combinados do segundo. A condição 2 refere-se à restrição física da mochila. Na tabela 4 estão indexados os compartimentos com suas respectivas capacidades.

	Agrupamento 1											**Agrupamento 2**						
Compartimento	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Capacidade	5	6	7	8	9	10	11	12	13	14	15	6	7	8	9	10	11	12

Tabela 4 – Capacidades dos compartimentos.

Dado um compartimento j de capacidade w_j e seu agrupamento associado, determinamos sua utilidade v_j por meio de um simples problema da mochila. Assim, para o agrupamento 1 temos 11 mochilas a considerar, $j = 1, \ldots, 11$, onde cada uma delas tem o seguinte aspecto:

$$\text{maximizar} \quad v_j = \sum_{i=1}^{3} u_i a_i$$

sujeito a:

$$\sum_{i=1}^{3} p_i a_i = w_j$$

$$a_i \geq 0 \text{ e inteiro}, \; i = 1, 2, 3$$

Observe que pode não haver uma combinação linear inteira dos pesos dos itens com valor igual à capacidade do compartimento, ou seja, a mochila anterior pode não ter solução. Neste caso, dizemos que o compartimento não é construtivo, no sentido de que ele não pode ser construído, e definimos como nula a sua utilidade, por exemplo:

1) O compartimento 7, que tem capacidade 11 não é construtivo, pois, ele não pode ser construído com os itens do agrupamento 1. Sua utilidade é nula.
2) O compartimento 8, que tem capacidade 12 é construtivo e sua utilidade é 17, pois, pode ser construído como: $\sum_{i=1}^{3} p_i a_i = 5.1 + 9.0 + 7.1 = 12$.
3) O compartimento 10, que tem capacidade 14 é construtivo e sua utilidade é 20, pois, pode ser construído como: $\sum_{i=1}^{3} p_i a_i = 5.1 + 9.1 + 7.0 = 14$.

Observe ainda que $\sum_{i=1}^{3} p_i a_i = 5.0 + 9.0 + 7.2 = 14$ é uma outra maneira de construir o compartimento 10, porém, adotamos a postura de escolher a combinação linear inteira que produz o maior valor de utilidade para o compartimento, maiores detalhes serão apresentados na próxima seção. Na tabela 5 estão marcados os compartimentos que podem ser construídos com os itens do respectivo agrupamento.

A existência de itens "dominados" por outros é um fato conhecido e útil na resolução de Problemas da Mochila, por exemplo, (GILMORE e GOMORY, 1961, 1963), (MARTELLO e TOTH, 1990, pag 78) e (CARVALHO e RODRIGUES, 1995).

No nosso exemplo, considere o compartimento 15, cuja capacidade é 9 e a utilidade é 15, e o compartimento 16, cuja capacidade é 10 e a utilidade é 8. No caso, existe certamente uma solução ótima em que o compartimento 16 não faz parte. Isso nos sugere escrever a seguinte definição:

Definição 1 Um compartimento construtivo de índice j é dominado quando existe um compartimento construtivo de índice h < j, associado ao mesmo agrupamento do compartimento de índice j, tal que a utilidade v_h do compartimento de índice h é maior ou igual à utilidade v_j do compartimento de índice j ($v_h \geq v_j$). Um compartimento **é dominante** quando ele **não é dominado**.

Na tabela 5 estão marcados os compartimentos dominantes do exemplo.

	Agrupamento 1											Agrupamento 2						
Compartimento	**1**	2	**3**	4	**5**	**6**	7	**8**	9	**10**	**11**	**12**	13	14	**15**	**16**	17	**18**
Capacidade	**5**	6	**7**	8	**9**	**10**	11	**12**	13	**14**	**15**	6	7	8	9	10	11	12
Utilidade	**9**	0	**8**	0	**11**	**18**	0	**17**	0	**20**	**27**	10	0	0	15	8	0	20
Construtivo	•		•		•	•		•		•	•	•			•	•		•
Dominante	•				•	•				•	•	•			•			•
Peso Item	5				9			7				3			6			10
Utilidade Item	9				11			8				5			6			8
Item	1				2			3				4			5			6

Tabela 5 – Compartimentos construtivos e dominantes.

Finalmente, uma solução do exemplo, indicando uma utilidade total de 37 para a mochila, é dada por:

1) Compartimento 1: $a_1 = 1, a_2 = 0, a_3 = 0$, aparecendo 3 vezes na mochila.

2) Compartimento 12: $a_4 = 2, a_5 = 0, a_6 = 0$, aparecendo 1 vez na mochila.

Uma solução alternativa, de igual utilidade, seria:

1) Compartimento 11: $a_1 = 3, a_2 = 0, a_3 = 0$, aparecendo 1 vez na mochila.

2) Compartimento 12: $a_4 = 2, a_5 = 0, a_6 = 0$, aparecendo 1 vez na mochila.

Na figura 14 ilustramos padrões de corte compartimentados que poderiam ser representados pelas soluções do exemplo que acabamos de descrever. Os compartimentos hachurados estão associados ao agrupamento 2 e os demais ao agrupamento 1. Note que os dois padrões compartimentados produzem os mesmos itens, porém, com diferentes compartimentos.

Observe que decidir qual preenchimento da mochila é melhor, o que representa o padrão 1 ou o que representa o padrão 2, depende do tipo de objetivo inicial que se pretende. Por exemplo, se também é desejado reduzir o número de compartimentos na mochila, a solução alternativa que representa o padrão 2 é sem dúvida melhor.

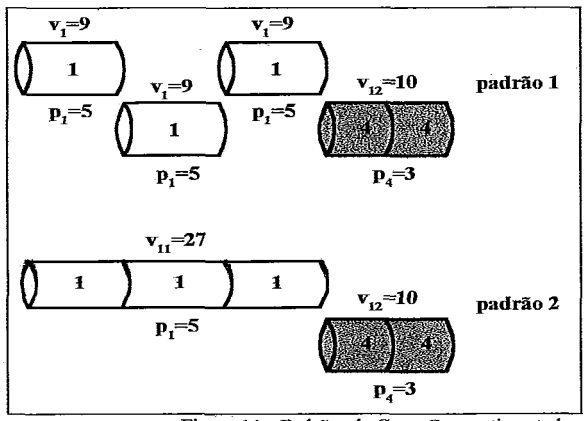

Figura 14 – Padrões de Corte Compartimentados.

Este objetivo poderia ser conseguido atribuindo um custo pela utilização de um compartimento, por exemplo, assuma por simplicidade, que os compartimentos associados ao agrupamento 1 possuem o mesmo custo de valor 3, e que os compartimentos associados a agrupamento 2 possuem o mesmo custo de valor 2. Neste caso, a primeira solução é descartada em favor da solução alternativa, pois, o custo no primeiro caso seria 11 e no segundo seria 5.

2 – Formulação Matemática do Problema da Mochila Compartimentada

Formalizando as idéias do exemplo anterior, considere uma mochila de capacidade c, onde $N = \{1, \ldots, n\}$ é o conjunto de índices dos itens de interesse, e sejam as utilidades u_i e os pesos p_i inteiros positivos para $i \in N$. Admita que nem todos os itens sejam compatíveis entre si, de modo que, uma partição $\{N_1, \ldots, N_k\}$ de N deverá ser considerada. Na mochila deverão ser construídos compartimentos para abrigar itens dos subconjuntos N_s, e cujas capacidades são limitadas entre dois valores inteiros, um mínimo d_s^{min} e um máximo d_s^{max}, $s = 1, \ldots, k$.

Para cada subconjunto N_s, considere o subconjunto V_s dos índices dos compartimentos construtivos a partir dos itens de N_s, tal que $V_p \cap V_q = \varnothing$ para $p, q = 1, \ldots, k$, $p \neq q$, como fizemos no exemplo anterior.

Observe que cada índice $j \in V_s$ refere-se a um único compartimento de capacidade w_j que deve ser definido com itens indexados pelo subconjunto N_s, porém, um compartimento de índice $j \in V_p$ e outro de índice $h \in V_q$, $p \neq q$, podem ter capacidades iguais, $w_j = w_h$. Para o exemplo descrito na introdução deste capítulo temos $V_1 = \{1, \ldots, 11\}$ e $V_2 = \{12, \ldots, 18\}$. Pela tabela 4, podemos observar que os compartimentos j=5 de V_1, e h=15 de V_2 possuem capacidades $w_5 = w_{15} = 9$.

Note ainda que em cada subconjunto V_s, compartimentos de índices distintos não possuem a mesma capacidade. Assim, quando nos referirmos a um compartimento j de capacidade w_j está subentendido que existe um único $s(j) \in \{1, \ldots, k\}$, tal que $j \in V_{s(j)}$, e que o compartimento deve ser construído com itens indexados pelo subconjunto $N_{s(j)}$, por exemplo, para os compartimentos j=5 e h=15 de capacidades $w_5 = w_{15} = 9$, temos s(5)=1 e s(15)=2.

Seja a_{ij} a variável que representará o número de itens i no compartimento j. A seguinte condição deverá ser obedecida:

$$w_j = \sum_{i \in N_s} p_i a_{ij},\ d_s^{min} \leq w_j \leq d_s^{max},\ j \in V_s \qquad (32)$$

Seja y_j o número de vezes que o compartimento j aparece na mochila, cujo custo pela sua utilização é um inteiro não negativo γ_j. O Problema da Mochila Compartimentada, doravante designado por PMC, é escrito como:

Modelo do PMC

$$\text{maximizar}\quad z = \sum_{j \in V_1}\left(\left(\sum_{i \in N_1} u_i a_{ij}\right) - \gamma_j\right) y_j + \cdots + \sum_{j \in V_k}\left(\left(\sum_{i \in N_k} u_i a_{ij}\right) - \gamma_j\right) y_j \qquad (33.1)$$

sujeito a:

$$\sum_{j \in V_1}\sum_{i \in N_1} p_i a_{ij} y_j + \cdots + \sum_{j \in V_k}\sum_{i \in N_k} p_i a_{ij} y_j \leq c \qquad (33.2)$$

$$d_s^{min} \leq \sum_{i \in N_s} p_i a_{ij} \leq d_s^{max},\ j \in V_s,\ s = 1, \ldots, k \qquad (33.3)$$

$a_{ij}, y_j \geq 0$ e inteiros,
$i \in N = N_1 \cup \cdots \cup N_k,\ j \in V = V_1 \cup \cdots \cup V_k$ (33.4)

A seguinte condição é admitida para evitar soluções triviais do PMC:

$$p_i \leq d_s^{max} \leq c, \forall i \in N_s, s = 1, \ldots, k \qquad (34.1)$$

Admitimos também a seguinte ordenação sob os ganhos marginais dos itens:

$$\frac{u_i}{p_i} \geq \frac{u_{i+1}}{p_{i+1}} \text{ para } i, i+1 \in N_s, s = 1, \ldots, k \qquad (34.2)$$

Observe que (33.2) é a restrição física da mochila. Denominamos (33.3) de "restrições de compartimentação", e os subconjuntos N_s de "agrupamentos".

O PMC tem aparecido em alguns problemas de corte como mencionamos na introdução desta tese. Destacamos CARVALHO e RODRIGUES (1994, 1995), página 40 do capítulo 2.

Os autores utilizam a Técnica de Geração de Colunas de GILMORE e GOMORY (1961, 1963), de modo que, para gerar um padrão de corte em duas fases, o espaço de busca é reduzido. Neste sentido, eles consideram os seguintes conjuntos de capacidades para os compartimentos:

$$C_i = \left\{ w_{ij} \in \mathbb{Z}_+^* \mid w_{ij} = a_{ij} p_i,\ d_i^{min} \leq w_{ij} \leq d_i^{max},\ a_{ij} \geq 0 \text{ e inteiro} \right\},\ i = 1, \ldots, n$$

Seja V_i o conjunto dos índices dos compartimentos, cujas capacidades encontram-se em C_i, $i = 1, \ldots, n$, tais que, $V_p \cap V_q = \varnothing$ para $p, q = 1, \ldots, n$, $p \neq q$.

Observe que a proposta de CARVALHO e RODRIGUES (1994, 1995) é construir para cada item i todos os compartimentos compostos por apenas um tipo de item. Em seguida, os autores propõem resolver a seguinte mochila:

Modelo de Carvalho-Rodrigues

maximizar $\sum_{j \in V_1} u_1 a_{1j} y_j + \cdots + \sum_{j \in V_n} u_n a_{nj} y_j$

sujeito a:

$$\sum_{j \in V_1} w_{1j} y_j + \ldots + \sum_{j \in V_n} w_{nj} y_j \leq c$$

$$y_j \geq 0 \text{ e inteiro}, j \in V_1 \cup \cdots \cup V_n$$

No modelo anterior, u_j é a utilidade dos itens (obtida por meio dos multiplicadores Simplex) e y_j é o número de vezes que o compartimento j aparece na mochila. Note então que o modelo usado pelos autores é um caso particular do modelo do PMC apresentado nesta tese.

3 – Resolução do Problema da Mochila Compartimentada

No PMC tanto a função objetivo, como a restrição física da mochila são não lineares. Para resolver o problema sugerimos decompô-lo em duas etapas: na primeira serão construídos todos os compartimentos dominantes com suas respectivas utilidades, e na segunda serão selecionados aqueles que definirão a compartimentação.

Antes de descrevermos estas duas etapas, identificaremos os compartimentos construtivos, como comentamos durante a resolução do exemplo dado no início deste capítulo.

Relembrando, um compartimento é construtivo quando existe uma combinação linear inteira e não negativa dos pesos dos itens do agrupamento associado, que é igual à capacidade do compartimento. Para eliminar os compartimentos que certamente não podem ser construídos podemos utilizar um procedimento bem conhecido que pode ser encontrado em (CHRISTOFIDES e WHITLOCK, 1977). A seguinte propriedade é válida:

Propriedade 1

Considere o compartimento $j \in V_{s(j)}$ de capacidade $w_j > d^{min}_{s(j)}$ e seu agrupamento $N_{s(j)}$ associado. Se existe algum item $r \in N_{s(j)}$, tal que o compartimento de capacidade $w_j - p_r$ é construtivo, então o compartimento de capacidade w_j é construtivo.

Prova Por hipótese temos $w_j - p_r = \sum_{i \in N_{s(j)}} p_i a_i$, de onde concluímos que

$$w_j = p_r(a_r + 1) + \sum_{\substack{i \in N_{s(j)} \\ i \neq r}} p_i a_i .$$ □

A seguir, apresentamos um algoritmo que seleciona compartimentos construtivos.

INICOMP – Algoritmo para Seleção de Compartimentos Construtivos

1. Inicialização
  1.1 faça $W := \varnothing$;

2. Exclusão dos Compartimentos Não Construtivos
  2.1 para s := 1 até k faça
    2.2 determine $p_{min} := \min\{p_i \mid i \in N_s\}$;
    2.3 faça $C := \{0, p_{min}\}$;
    2.4 para $w := p_{min} + 1$ até d_s^{max} faça
      2.5 faça $i := \min N_s$;
      2.6 enquanto $i \leq \max N_s$ faça
        2.7 se $w - p_i \in C$ então
          2.8 faça $C := C \cup \{w\}$;
          2.9 faça $i := 2.\max N_s$;
        fim se 2.7;
        2.10 $i := i + 1$;
      fim enquanto 2.6;
    fim para 2.4;
    2.11 $C := C - \{0\}$;
    2.12 se $p_{min} \leq d_s^{min}$ então faça $C := C - \{w \in C \mid w < d_s^{min}\}$;
    2.13 construa o subconjunto $W_s \subseteq V_s$ dos índices dos compartimentos, cujas capacidades encontram-se em C;
    2.14 faça $W := W \cup W_s$;
  fim para 2.1;

Observe que uma vez sendo disjuntos os conjuntos V_s, também serão os conjuntos W_s, $s = 1, \ldots, k$. Assim, no conjunto W estão todos os índices dos compartimentos, cujas capacidades possuem pelo menos uma combinação linear exata dos pesos dos itens indexados por N_s. Vejamos agora as duas etapas da decomposição.

3.1 – Primeira Etapa da Decomposição

Note que $W_s \subseteq V_s$, $s = 1, \ldots, k$, desta forma, para cada compartimento construtivo $j \in W = W_1 \cup \ldots \cup W_k$ existe um único agrupamento $N_{s(j)}$ associado (veja exemplo inicial). Nesta etapa da decomposição do problema, definiremos as utilidades de todos os compartimentos construtivos por meio do seguinte problema da mochila:

$$\text{maximizar} \quad v_j = \sum_{i \in N_{s(j)}} u_i a_{ij} \qquad (35.1)$$

sujeito a:

$$\sum_{i \in N_{s(j)}} p_i a_{ij} = w_j \qquad (35.2)$$

$$a_{ij} \geq 0 \text{ e inteiro, } i \in N_{s(j)} \qquad (35.3)$$

O problema (35.1 – 35.3) é resolvido pelo algoritmo de Yanasse-Soma (YANASSE e SOMA, 1987), cujo pseudocódigo está descrito no apêndice E. Basicamente o algoritmo resolve o problema de maior capacidade $w_j = d_{s(j)}^{max}$, onde são calculadas as soluções das mochilas com $w_j = d_{s(j)}^{min}, d_{s(j)}^{min}+1, \ldots, d_{s(j)}^{max}-1$.

3.2 – Segunda Etapa da Decomposição

Seja $W' \subseteq W \subseteq V = V_1 \cup \ldots \cup V_k$ o conjunto dos índices dos compartimentos dominantes, para os quais encontramos v_j por meio do problema (35.1 – 35.3). Para determinarmos a compartimentação da mochila, basta resolver o seguinte problema da mochila:

$$\text{maximizar} \quad \sum_{j \in W'} (v_j - \gamma_j) y_j \qquad (36.1)$$

sujeito a:

$$\sum_{j \in W'} w_j y_j \leq c \qquad (36.2)$$

$$y_j \geq 0 \text{ e inteiro, } j \in W' \qquad (36.3)$$

O problema (36.1 – 36.3) também pode ser resolvido pelo algoritmo de Yanasse-Soma usado na resolução do problema (35.1 – 35.3), porém, deve-se efetuar uma pequena modificação que admite a restrição física da mochila no formato de desigualdade, e não igualdade. O pseudocódigo desta modificação também pode ser encontrado no apêndice E. O custo γ_j do compartimento j pode ser definido após a sua construção. Este custo depende sobre tudo dos itens que define o compartimento em questão. Observe que o problema (35.1 – 35.3) pode apresentar soluções alternativas, assim teríamos várias possibilidades para o custo de utilização do compartimento j, e todas elas devem ser consideradas.

Afirmamos que é suficiente escolher apenas aquela que proporciona o menor custo, pois, as demais escolhas são dominadas pela de menor custo na resolução da mochila (36.1 – 36.3).

A seguir, apresentamos um algoritmo com as idéias que foram descritas.

COMPEX – Algoritmo para Compartimentação Exata

1. Inicialização
   1.1 forneça os dados da mochila: c e (u_i, p_i) para $i \in N$;
   1.2 forneça os dados da compartimentação: $(N_s, d_s^{min}, d_s^{max})$ para $s = 1, \ldots, k$;
   1.3 construa $W = W_1 \cup \cdots \cup W_k$ (procedimento INICOMP);

2. Determinação das Utilidades dos Compartimentos
   2.1 para s := 1 até k faça
      2.2 ordene os itens segundo as eficiências: $\frac{u_i}{p_i} \geq \frac{u_{i+1}}{p_{i+1}}$ para $i, i+1 \in N_s$;
      2.3 resolva a mochila (35.1 – 35.3) com $w_j = d_s^{max}$,
         pelo algoritmo de Yanasse-Soma;
      2.4 para $w_j := d_s^{min}$ até d_s^{max} faça
         2.5 se o compartimento j de capacidade w_j é construtivo
            <u>então</u> recupere o objetivo v_j da mochila (35.1 – 35.3) e calcule γ_j, $j \in W_s$;
         fim se 2.5
      fim para 2.4;
   fim para 2.1;

3. Determinação da Compartimentação
   3.1 construa o subconjunto W' dos índices dos compartimentos
      dominantes, de modo que, $W' = \{1, \ldots, p'\}$;
   3.2 ordene os compartimentos segundo as eficiências: $\frac{v_j}{w_j} \geq \frac{v_{j+1}}{w_{j+1}}$ para $j, j+1 \in W'$;
   3.3 resolva a mochila (36.1 – 36.3) e encontre y^*;

4. Construção dos Compartimentos Escolhidos
   4.1 para j := 1 até p' faça
      4.2 se $y_j^* \neq 0$ então
         4.3 identifique o agrupamento N_s associado ao compartimento escolhido;
         4.4 recupere a solução da mochila (35.1 – 35.3);
         4.5 armazene o compartimento construído;
      fim se 4.2;
   fim para 4.1;

Finalizando a seção, mostraremos que o algoritmo anterior encontra o ótimo da Mochila Compartimentada.

Propriedade 2

O algoritmo COMPEX encontra uma solução ótima do Problema da Mochila Compartimentada (PMC), quando ela existe.

Prova Para cada compartimento $j \in V$, existe um único $s(j) \in \{1, \ldots, k\}$ que depende de j, tal que $j \in V_{s(j)}$, visto que os conjuntos V_s, $s = 1, \ldots, k$, foram definidos como uma partição de V. Seja w_j a capacidade deste compartimento j e considere $\Omega_j = \left\{ (a_{ij})_{i \in N_{s(j)}} \middle| w_j = \sum_{i \in N_{s(j)}} p_i a_{ij} \right\}$ o conjunto de todos os vetores que representam uma combinação linear dos pesos dos itens com índices em $N_{s(j)}$ igual a w_j.

Note que para algum j é possível que Ω_j seja vazio, porém, estamos admitindo que exista Ω_j não vazio, pois, do contrário o PMC não teria solução. Suponha que Ω_j tenha m_j vetores e considere $v_j^h = \sum_{i \in N_{s(j)}} u_i a_{ij}$, $h = 1, \ldots, m_j$ os possíveis valores de utilidades para o compartimento j que tem capacidade w_j. Escolhendo $v_j = \max\{v_j^1, \ldots, v_j^{m_j}\}$ garantimos que o valor v_j é dominante sob as demais utilidades. Este mesmo raciocínio se aplica ao custo γ_j de utilização do compartimento j, sendo suficiente para concluir que uma solução ótima do problema (36.1 – 36.3) também será do modelo (33.1 – 33.4) do PMC. □

4 – Uma Heurística de Compartimentação

O objetivo da heurística é contornar a resolução da mochila no passo 2 do COMPEX e, assim, acelerar o processo de compartimentação. Nossa sugestão é substituí-la pelo simples cálculo do limitante de MARTELLO e TOTH (1990, pag 93), apresentado no capítulo anterior, que fornecerá uma boa aproximação superior para as utilidades dos compartimentos

Com esta estratégia não poderemos calcular o custo γ_j pela utilização do compartimento j na mochila, pois, ele depende dos itens que define o compartimento em questão. Assim, adotaremos a postura de definir o mesmo custo γ_s para todos os compartimentos indexados em W_s com itens indexados em N_s, $s = 1, \ldots, k$.

No passo 3 do COMPEX é feita a escolha dos compartimentos que devem compor a mochila. O fato de não conhecermos os reais valores das utilidades dos compartimentos nos impede de determinar os verdadeiros compartimentos dominados, de modo que, o subconjunto W' do problema (36.1 – 36.3) pode, no caso de usarmos limitantes, não indexar os verdadeiros compartimentos dominantes. Assim, resolveremos a mochila (36.1 – 36.3) em W ao invés de W', quando estivermos usando limitantes para as utilidades dos compartimentos, evitando o descarte indevido de alguns deles.

Vejamos agora o procedimento heurístico para construir uma solução viável do PMC:

COMPMT – Algoritmo para Compartimentação com Limitantes de Martello-Toth

1. Inicialização
   1.1 forneça os dados da mochila: c e (u_i, p_i) para $i \in N$;
   1.2 forneça os dados da compartimentação: $(N_s, d_s^{min}, d_s^{max}, \gamma_s)$ para $s = 1, \ldots, k$;
   1.3 construa $W = W_1 \cup \cdots \cup W_k$ (procedimento INICOMP);

2. Determinação dos Limitantes das Utilidades dos Compartimentos
   2.1 para s := 1 até k faça
      2.2 ordene os itens segundo as eficiências: $\frac{u_i}{p_i} \geq \frac{u_{i+1}}{p_{i+1}}$ para $i, i+1 \in N_s$;
      2.3 para $w_j := d_s^{min}$ até d_s^{max} faça
         2.4 se o compartimento de capacidade w_j é construtivo
            <u>então</u> calcule Z_{MT} (limitante de Martello-Toth), faça $v_j = Z_{MT}$ e
               faça $\gamma_j := \gamma_s$, $j \in W_s$;
         fim se 2.4
      fim para 2.3;
   fim para 2.1;

3. Determinação da Compartimentação
   3.1 redefina os dados dos compartimentos com índices em $W = W_1 \cup \cdots \cup W_k$, de modo que, $W = \{1, \ldots, p\}$;
   3.2 ordene os compartimentos segundo as eficiências: $\frac{v_j}{w_j} \geq \frac{v_{j+1}}{w_{j+1}}$ para $j, j+1 \in W$;
   3.3 resolva a mochila (36.1 – 36.3) e encontre y^*;

4. Construção dos Compartimentos Escolhidos
   4.1 para $j := 1$ até p faça
      4.2 se $y_j^* \neq 0$ então resolva a mochila (35.1 – 35.3);
      4.3 armazene o compartimento construído;
   fim para 4.1;

No passo 4 do COMPMT poderíamos atualizar as utilidades e os custos de utilização dos compartimentos selecionados e, em seguida, retornar ao passo 3 para que fosse feita uma nova compartimentação da mochila, resultando num novo passo 4:

Passo 4 do COMPMT com atualização dos Compartimentos

4. Construção dos Compartimentos Escolhidos
   4.1 **atualizar := falso;**
   4.2 para cada $y_j^* \neq 0$ faça
      4.3 identifique o agrupamento N_s associado ao compartimento escolhido;
      4.4 se a utilidade do compartimento escolhido é um limitante
         <u>então</u>
            4.5 resolva a mochila (35.1 – 35.3);
            4.6 atualize a utilidade do compartimento com o valor de v_j e calcule γ_j, $j \in W_s$;
            4.7 exclua de W compartimentos dominados;
            4.8 se **atualizar = falso** <u>então</u> **atualizar := verdade;**
         <u>senão</u> armazene o compartimento construído;
      fim se 4.4;
   fim para 4.2;
   4.9 se **atualizar = verdade** <u>então</u> **retorne ao passo 3;**

Ressaltamos que esta nova estratégia pode não ser adequada, pois, no pior caso o COMPMT atualizaria os dados de todos os compartimentos e seu tempo computacional ficaria prejudicado, não obstante, possa encontrar uma solução antes de atualizar as utilidades e os custos de utilização de todos os compartimentos construtivos.

Além do mais, no COMPEX as utilidades dos compartimentos são obtidas dinamicamente, de modo que com este novo passo 4, o tempo computacional do COMPMT pode ser superior ao do COMPEX. Em vista disso decidimos **não efetuar** esta modificação no COMPMT.

5 - Resultados Computacionais pelo COMPEX e COMPMT

Na tabela 6 resumimos os resultados que obtivemos para 900 exemplos de mochilas compartimentadas, geradas aleatoriamente. A capacidade de todas as mochilas é 1200, os coeficientes p_i foram gerados no intervalo $[3, d_s^{max}]$, um parâmetro $0<\alpha<1$ foi gerado aleatoriamente para definirmos os coeficientes $u_i = 10\alpha p_i + \beta$, onde β é uma constante. Isso foi feito para que as utilidades ficassem relacionadas com os pesos. O custo por utilizar um compartimento foi considerado como nulo.

Os algoritmos foram implementados em Delphi e executados num pentium II, 450 Mhz e com 160 Mb de RAM. Os exemplos foram agrupados em 3 grandes categorias, segundo o número de compartimentos, e cada uma foi dividida em duas outras, segundo o número de agrupamentos. A menor compartimentação é composta por 30 compartimentos e 3 agrupamentos com 5 itens em cada. A maior compartimentação é formada por 6000 compartimentos e 20 agrupamentos com 100 itens em cada. As colunas da tabela 6 são:

d ^{min} e ***d*** ^{max} : *respectivamente, o limite mínimo e máximo dos capacidades dos compartimentos*;

Agrup : *número de agrupamentos da compartimentação*;

Itens : *número de itens num agrupamento da compartimentação*;

Igualdade : *percentual de exemplos em que as soluções do* COMPEX *e* COMPMT *foram iguais*;

Tempo : *tempo médio de execução de um exemplo*;

Perda : *percentual de perda (espaço ocioso) na mochila*;

Dif : *percentual de diferença entre o valor da solução do* COMPEX *e do* COMPMT *calculado pela expressão* $100.\left|(1-\frac{\mathrm{Val}_{\mathrm{COMPMT}}}{\mathrm{Val}_{\mathrm{COMPEX}}})\right|$, *onde* $\mathrm{Val}_{\mathrm{COMPEX}}$ *é o valor do objetivo obtido pela solução do* COMPEX, *e* $\mathrm{Val}_{\mathrm{COMPMT}}$ *o valor do objetivo obtido pela solução do* COMPMT;

***c* = 1200**						***COMPEX***		***COMPMT***	
d^{min}	d^{max}	***Agrup***	***Itens***	***Dif***	***Igualdade***	***Tempo***	***Perda***	***Tempo***	***Perda***
31	40	3	5	0,54%	74,00%	0,110 seg	0,23%	0,050 seg	0,41%
			40	0,29%	22,00%	0,110 seg	---	0,050 seg	0,20%
			100	0,17%	10,00%	0,112 seg	---	0,055 seg	---
		20	5	0,19%	68,00%	0,330 seg	---	0,083 seg	0,11%
			40	0,05%	14,00%	0,375 seg	---	0,095 seg	0,06%
			100	0,01%	8,00%	0,380 seg	---	0,110 seg	0,01%
51	100	3	5	0,29%	76,00%	0,110 seg	0,26%	0,045 seg	0,48%
			40	0,15%	52,00%	0,126 seg	---	0,069 seg	0,09%
			100	0,07%	24,00%	0,129 seg	---	0,071 seg	0,03%
		20	5	0,29%	62,00%	0,470 seg	0,03%	0,170 seg	0,18%
			40	0,17%	34,00%	0,625 seg	---	0,330 seg	0,05%
			100	0,02%	24,00%	0,741 seg	---	0,379 seg	0,02%
151	450	3	5	1,10%	60,00%	0,173 seg	0,84%	0,088 seg	1,68%
			40	0,38%	22,00%	0,361 seg	0,11%	0,197 seg	0,30%
			100	0,33%	20,00%	0,380 seg	0,02%	0,218 seg	0,15%
		20	5	0,90%	58,00%	1,050 seg	0,17%	0,550 seg	0,64%
			40	0,40%	16,00%	4,547 seg	0,01%	3,507 seg	0,22%
			100	0,29%	18,00%	6,150 seg	---	3,570 seg	0,12%

Tabela 6 – Resultados numéricos de 900 exemplos.

Os resultados obtidos mostram que o algoritmo COMPMT (heurística) é competitivo em relação ao COMPEX (exato) para problemas em que o número de compartimentos é alto, porém, ressaltamos que o desempenho deste algoritmo pode não ser satisfatório para alguma classe de problemas, visto que, trata-se de uma heurística. Nos problemas maiores, o COMPMT obteve menos soluções ótimas, entretanto, a diferença dos objetivos não se mostrou significativa.

O gráfico a seguir está fragmentado em três partes de acordo com o número de compartimentos: no primeiro fragmento são exemplos com 30 e 200 compartimentos, no segundo fragmento são exemplos com 150 e 1000 compartimentos e no terceiro fragmento são exemplos com 900 e 6000 compartimentos.

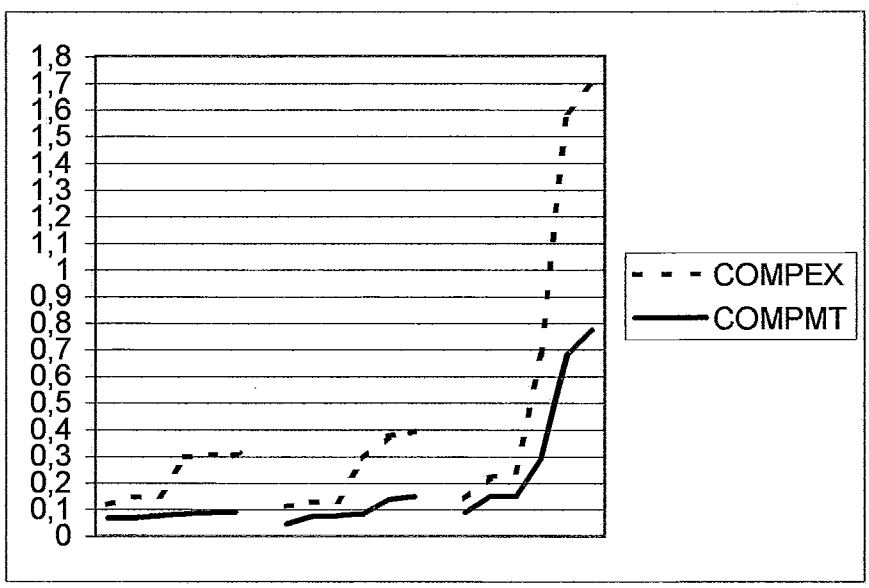

Figura 15 - Gráfico do tempo entre COMPEX e COMPMT para compartimentos.

O próximo gráfico está fragmentado em duas partes de acordo com o número de agrupamentos: 5 para o primeiro fragmento e 20 para o segundo.

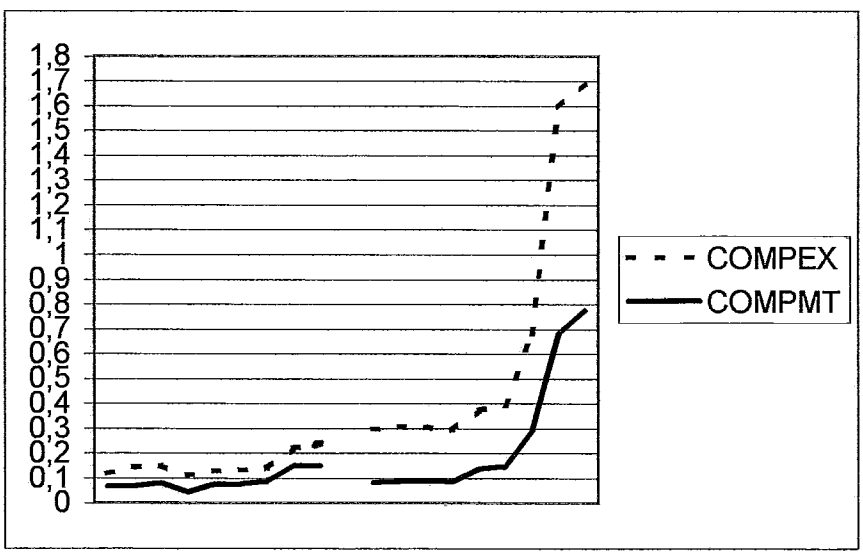

Figura 16 - Gráfico do tempo entre COMPEX e COMPMT para agrupamentos.

6 – O Problema da Mochila Compartimentada 0-1

Este é o caso em que cada compartimento será utilizado apenas uma vez, ou então não será utilizado. O PMC 0-1 pode ser resolvido pelo COMPMT ou pelo COMPEX, basta alterar a mochila do passo 3 para uma mochila 0-1.

Modelo do PMC 0-1

$$\text{maximizar} \quad z = \sum_{j \in V_1} \left(\left(\sum_{i \in N_1} u_i a_{ij} \right) - \gamma_j \right) y_j + \cdots + \sum_{j \in V_k} \left(\left(\sum_{i \in N_k} u_i a_{ij} \right) - \gamma_j \right) y_j \qquad (37.1)$$

sujeito a:

$$\sum_{j \in V_1} \sum_{i \in N_1} p_i a_{ij} y_j + \cdots + \sum_{j \in V_k} \sum_{i \in N_k} p_i a_{ij} y_j \le c \qquad (37.2)$$

$$d_s^{min} \le \sum_{i \in N_s} p_i a_{ij} \le d_s^{max}, \; j \in V_s, \; s = 1, \ldots, k \qquad (37.3)$$

$$y_j \in \{0, 1\}, \; a_{ij} \ge 0 \text{ e inteiros}, \quad i \in N = N_1 \cup \cdots \cup N_k, \; j \in V = V_1 \cup \cdots \cup V_k \qquad (37.4)$$

HOTO *et al.* (1999) descreve um caso particular do PMC 0-1 apresentado acima, para o qual o autor restringe a escolha dos compartimentos a apenas um por agrupamento, e cujo modelo é apresentado a seguir:

Um Caso Particular do PMC 0-1

$$\text{maximizar} \quad z = \sum_{j \in V_1} \left(\left(\sum_{i \in N_1} u_i a_{ij} \right) - \gamma_1 \right) y_j + \cdots + \sum_{j \in V_k} \left(\left(\sum_{i \in N_k} u_i a_{ij} \right) - \gamma_k \right) y_j \qquad (38.1)$$

sujeito a:

$$\sum_{j \in V_1} \sum_{i \in N_1} p_i a_{ij} y_j + \cdots + \sum_{j \in V_k} \sum_{i \in N_k} p_i a_{ij} y_j \le c \qquad (38.2)$$

$$d_s^{min} \le \sum_{i \in N_s} p_i a_{ij} \le d_s^{max}, \; j \in V_s, \; s = 1, \ldots, k \qquad (38.3)$$

$$\sum_{j \in V_s} y_j = 1, \; s = 1, \ldots, k \qquad (38.4)$$

$$y_j \in \{0, 1\}, \; a_{ij} \ge 0 \text{ e inteiros}, \quad i \in N = N_1 \cup \cdots \cup N_k, \; j \in V = V_1 \cup \cdots \cup V_k \qquad (38.5)$$

No modelo acima foi introduzida a restrição (38.4) que é justamente a responsável em exigir apenas um compartimento por agrupamento. Além das condições

(34.1) $p_i \le d_s^{max} \le c, \forall i \in N_s, s = 1, \ldots, k$ e (34.2) $\frac{u_i}{p_i} \ge \frac{u_{i+1}}{p_{i+1}}$ para $i, i+1 \in N_s$, $s = 1, \ldots, k$ já admitidas, foram introduzidas mais duas condições:

$$\sum_{s=1}^{k} d_s^{max} > c \qquad (39.1)$$

$$\frac{u_{\min N_s}}{p_{\min N_s}} \ge \frac{u_{\min N_{s+1}}}{p_{\min N_{s+1}}},\ s, s+1 \in \{1, \ldots, k\} \qquad (39.2)$$

A condição (39.1) é para evitar soluções triviais do problema (38.1 – 38.5), e (39.2) assegura uma ordenação sob os agrupamentos que será usada no cálculo de limitantes superiores. Um *branch-and-bound* é descrito para este caso particular do PMC. Para isso considere a seguinte propriedade:

Propriedade 4

Considere $j(1) \in V_1$, ..., $j(k) \in V_k$ os índices dos compartimentos escolhidos numa solução ótima (a^*, x^*) do problema (38.1 – 38.5). Nestas condições, para qualquer $h \in N = N_1 \cup \cdots \cup N_k$ vale a seguinte desigualdade:

$$c - \left(\sum_{i \in N_1} p_i a_i^* y_{j(1)}^* + \cdots + \sum_{i \in N_k} p_i a_i^* y_{j(k)}^* \right) < p_h$$

Em particular a desigualdade vale para $p_h = \min_{i \in N} \{p_i\}$.

Prova De fato, caso contrário a_h^* poderia ser incrementado de uma unidade. □

Definição 2 (soluções sensíveis)

Denominamos sensível uma solução viável (a, x) do problema (38.1 – 38.5), verificando a seguinte desigualdade: $c - \sum_{i \in N_1} p_i a_i^* y_1^* + \cdots + \sum_{i \in N_k} p_i a_i^* y_k^* \ge \min_{i \in N} \{p_i\}$.

Observe que se uma solução do problema (38.1 – 38.5) é sensível, então pela propriedade 4 ela não tem chance de ser uma solução ótima.

Definição 3 (justaposição de agrupamentos)

Dois agrupamentos N_s e N_{s+1} são justapostos quando:

$$\frac{u_{\max N_s}}{p_{\max N_s}} \geq \frac{u_{\min N_{s+1}}}{p_{\min N_{s+1}}}, \text{ para } s, s+1 \in \{1, \ldots, k\}$$

Definição 4 (pseudointersecção de agrupamentos)

Dois agrupamentos N_s e N_t admitem pseudo-intersecção quando:

$$\frac{u_{\min N_t}}{p_{\min N_t}} > \frac{u_{\max N_s}}{p_{\max N_s}}, \text{ para } s < t \text{ com } s, t \in \{1, \ldots, k\}$$

Para ilustrar a pseudointersecção de agrupamentos considere o exemplo inicial apresentado neste capítulo, cujos dados encontram-se na tabela 3. Note que para o primeiro agrupamento a ordenação dos itens, segundo seus ganhos marginais, é dada por $\frac{9}{5} \geq \frac{11}{9} \geq \frac{8}{7}$. Para o segundo agrupamento temos $\frac{5}{3} \geq \frac{6}{6} \geq \frac{8}{10}$. Neste caso os agrupamentos possuem pseudo-intersecção, pois, $\frac{u_{\min N_2}}{p_{\min N_2}} = \frac{5}{3} > \frac{8}{7} = \frac{u_{\max N_1}}{p_{\max N_1}}$.

Por simplicidade assumiremos $d_1^{\min} = \cdots = d_k^{\min} = \alpha$ e $d_1^{\max} = \cdots = d_k^{\max} = \beta$ e faremos a construção do *branch-and-bound*, enumerando implicitamente as variáveis y_j, $j \in V_s$, $s = 1, \ldots, k$. Cada nó da árvore armazena o valor máximo de uma variável a_{ij} que ora será calculado pela restrição de compartimentação, ora pela restrição física da mochila (será escolhida a mais restritiva).

Para definir um nó arbitrário da árvore, considere um item de índice $h \in N_s$ para algum $s = 1, \ldots, k$. Sejam $j(1) \in V_1$, ..., $j(s) \in V_s$ os índices dos compartimentos construídos (lembre que será construído apenas um compartimento por agrupamento).

A variável $a_{h\,j(s)}$ deve ser limitada superiormente por:

$$\beta_h = \min\left\{\left\lfloor\left(c - \sum_{i\in N_1} p_i a_{i\,j(1)} - \ldots - \sum_{\substack{i\in N_s \\ i\le h-1}} p_i a_{i\,j(s)}\right)\Big/ p_h\right\rfloor, \left\lfloor\left(\beta - \sum_{\substack{i\in N_s \\ i\le h-1}} p_i a_{i\,j(s)}\right)\Big/ p_h\right\rfloor\right\}$$

Ela deve ser limitada inferiormente por:

$$\alpha_h = \max\left\{0, \left\lceil\left(\alpha - \sum_{\substack{i\in N_s \\ i\le h-1}} p_i a_{i\,j(s)}\right)\Big/ p_h\right\rceil\right\}$$

Nas expressões acima, $\lfloor\,.\,\rfloor$ é a função maior inteiro menor ou igual a ... (função piso) e, $\lceil\,.\,\rceil$ é a função menor inteiro maior ou igual a ... (função teto). Observe que $\alpha_h \le a_{h\,j(s)} \le \beta_h$ com $h \in N_s$, para algum $s = 1, \ldots, k$. Inferiormente estamos exigindo muito de $a_{h\,j(s)}$ e deixando de examinar potenciais soluções, desta forma iremos permitir que $0 \le a_{h\,j(s)} \le \beta_h$, a menos que $a_{h\,j(s)}$ seja a última variável do compartimento, quando iremos impor que $\alpha_h \le a_{h\,j(s)} \le \beta_h$ para garantir que o compartimento tenha capacidade superior a α.

Quanto ao critério de poda da árvore, digamos que $\hat{a}$ seja a melhor solução encontrada até o momento, caminharemos desde o final do ramo que define esta solução, em direção a raiz da árvore, até obtermos o primeiro índice r (que será o maior), tal que $\hat{a}_{r\,j(s)} > \alpha_r$, $r \in N_s$, $r < n$. Definiremos uma nova solução (um novo ramo) $\bar{a}$ da seguinte forma:

$$\bar{a}_{i\,j(s)} = \hat{a}_{i\,j(s)},\ i = 1, \ldots, r-1 \qquad (40.1)$$

$$\bar{a}_{r\,j(s)} = \hat{a}_{r\,j(s)} - 1 \qquad (40.2)$$

Para $i \ge r+1$ faremos uma estimativa do valor de $\bar{z}$. Suponha que $r+1 \in N_s$ e considere $\lambda_{r+1}^{s+1} = \max\left\{\dfrac{u_{r+1}}{p_{r+1}}, \dfrac{u_{\min N_{s+1}}}{p_{\min N_{s+1}}}\right\}$. Note que $r \in N_s$, ou $r \in N_{s-1}$, calcularemos a estimativa de $\bar{z}$ para o primeiro caso, sendo o segundo análogo.

$$\overline{z}=\sum_{i\in N_1} u_i\overline{a}_{i\,j(1)}\overline{y}_{j(1)}+\ldots+\sum_{i\in N_s} u_i\overline{a}_{i\,j(s)}\overline{y}_{j(s)}+\ldots+\sum_{i\in N_k} u_i\overline{a}_{i\,j(k)}\overline{y}_{j(k)}-\sum_{s=1}^{k}\gamma_s \Rightarrow$$

$$\overline{z}=\sum_{i\in N_1} u_i\overline{a}_{i\,j(1)}\overline{y}_{j(1)}+\ldots+\sum_{\substack{i\in N_s\\ i\le r}} u_i\overline{a}_{i\,j(s)}\overline{y}_{j(s)}+\sum_{\substack{i\in N_s\\ i\ge r+1}} u_i\overline{a}_{i\,j(s)}\overline{y}_{j(s)}+\ldots+$$

$$+\sum_{i\in N_k} u_i\overline{a}_{i\,j(k)}\overline{y}_{j(k)}-\sum_{s=1}^{k}\gamma_s$$

Substituindo (40.1) e (40.2) na expressão anterior teremos:

$$\overline{z}=\sum_{i\in N_1} u_i\hat{a}_{i\,j(1)}\hat{y}_{j(1)}+\ldots+\sum_{\substack{i\in N_s\\ i<r}} u_i\hat{a}_{i\,j(s)}\hat{y}_{j(s)}+\left(u_r(\hat{a}_{r\,j(s)}-1)\hat{y}_{j(s)}\right)+$$

$$+\sum_{\substack{i\in N_s\\ i\ge r+1}} u_i\overline{a}_{i\,j(s)}\overline{y}_{j(s)}+\ldots+\sum_{i\in N_k} u_i\overline{a}_{i\,j(k)}\overline{y}_{j(k)}-\sum_{s=1}^{k}\gamma_s \Rightarrow$$

$$\overline{z}=\sum_{i\in N_1} u_i\hat{a}_{i\,j(1)}\hat{y}_{j(1)}+\ldots+\sum_{\substack{i\in N_s\\ i\le r}} u_i\hat{a}_{i\,j(s)}\hat{y}_{j(s)}+\left(-u_r\hat{y}_{j(s)}\right)+$$

$$+\sum_{\substack{i\in N_s\\ i\ge r+1}} u_i\overline{a}_{i\,j(s)}\overline{y}_{j(s)}+\ldots+\sum_{i\in N_k} u_i\overline{a}_{i\,j(k)}\overline{y}_{j(k)}-\sum_{s=1}^{k}\gamma_s \Rightarrow$$

$$\overline{z}\le\sum_{i\in N_1} u_i\hat{a}_{i\,j(1)}\hat{y}_{j(1)}+\ldots+\sum_{\substack{i\in N_s\\ i\le r}} u_i\hat{a}_{i\,j(s)}\hat{y}_{j(s)}+\left(-u_r\hat{y}_{j(s)}\right)+$$

$$+\lambda_{r+1}^{s+1}\left(\sum_{\substack{i\in N_s\\ i\ge r+1}} p_i\overline{a}_{i\,j(s)}\overline{y}_{j(s)}+\ldots+\sum_{i\in N_k} p_i\overline{a}_{i\,j(k)}\overline{y}_{j(k)}\right)-\sum_{s=1}^{k}\gamma_s$$

Uma vez que a expressão multiplicada por λ_{r+1}^{s+1} é limitada superiormente por:

$$c-\left(\sum_{i\in N_1} p_i\hat{a}_{ij(1)}\hat{y}_{j(1)}+\ldots+\sum_{\substack{i\in N_s\\ i\le r}} p_i\hat{a}_{ij(s)}\hat{y}_{j(s)}+\left(-p_r\hat{y}_{j(s)}\right)\right)$$

Resulta que $\bar{z}$ é limitado por:

$$\bar{z}\le\sum_{i\in N_1} u_i\hat{a}_{ij(1)}\hat{y}_{j(1)}+\ldots+\sum_{\substack{i\in N_s\\ i\le r}} u_i\hat{a}_{ij(s)}\hat{y}_{j(s)}+\left(-u_r\hat{y}_{j(s)}\right)+$$

$$+\lambda_{r+1}^{s+1}\left(c-\sum_{i\in N_1} p_i\hat{a}_{ij(1)}\hat{y}_{j(1)}-\ldots-\sum_{\substack{i\in N_s\\ i\le r}} p_i\hat{a}_{ij(s)}\hat{y}_{j(s)}-\left(-p_r\hat{y}_{j(s)}\right)\right)-\sum_{s=1}^{k}\gamma_s$$

Reorganizando a última expressão teremos finalmente que:

$$\bar{z}\le Z_1=\sum_{i\in N_1} u_i\hat{a}_{ij(1)}\hat{y}_{j(1)}+\ldots+\sum_{\substack{i\in N_s\\ i\le r}} u_i\hat{a}_{ij(s)}\hat{y}_{j(s)}+$$

$$+\lambda_{r+1}^{s+1}\left(c-\sum_{i\in N_1} p_i\hat{a}_{ij(1)}\hat{y}_{j(1)}-\ldots-\sum_{\substack{i\in N_s\\ i\le r}} p_i\hat{a}_{ij(s)}\hat{y}_{j(s)}\right)-\sum_{s=1}^{k}\gamma_s-\left(u_r-\lambda_{r+1}^{s+1}p_r\right)\hat{y}_{j(s)}$$

Muito bem, caso $Z_1<\hat{z}+1$ (todos os coeficientes são inteiros) a ramificação **não fornecerá** uma solução melhor que a existente. Obviamente o contrário não implica que $\bar{a}$ será de fato melhor que $\hat{a}$, mas, há possibilidade. Para os demais ramos em que $\bar{a}_r=\hat{a}_r-2$, ..., $\bar{a}_r=0$, os limitantes podem ser calculados pela expressão:

$$Z_m=Z_1-m\left(u_r-\lambda_{r+1}^{s+1}p_r\right)\hat{y}_{j(s)},\ m=2,\ldots,M,\ \text{onde } M=\hat{a}_r \qquad (41)$$

Observe em (41) que quando $u_r-\lambda_{r+1}^{s+1}p_r\ge 0$, segue que:

$$Z_M\le\ldots\le Z_2\le Z_1 \qquad (42.1)$$

Por outro lado quando $u_r - \lambda_{r+1}^{s+1} p_r < 0$, segue que:

$$Z_1 < Z_2 \ldots < Z_M \qquad (42.2)$$

Considere os seguintes resultados, lembre que $\lambda_{r+1}^{s+1} = \max\left\{\frac{u_{r+1}}{p_{r+1}}, \frac{u_{\min N_{s+1}}}{p_{\min N_{s+1}}}\right\}$:

Lema 1

Se $r, r+1 \in N_s$ que é justaposto N_{s+1}, então $Z_M \leq \ldots \leq Z_2 \leq Z_1$.

Prova De fato, $\frac{u_r}{p_r} \geq \frac{u_{r+1}}{p_{r+1}} \geq \frac{u_{\max N_s}}{p_{\max N_s}} \geq \frac{u_{\min N_{s+1}}}{p_{\min N_{s+1}}}$, daí $u_r - \lambda_{r+1}^{s+1} p_r \geq 0$. □

Lema 2

Se $r \in N_{s-1}$ e $r+1 \in N_s$, tal que N_{s-1} é justaposto N_s, então $Z_M \leq \ldots \leq Z_2 \leq Z_1$.

Prova De fato, como $\frac{u_r}{p_r} \geq \frac{u_{\max N_{s-1}}}{p_{\max N_{s-1}}} \geq \frac{u_{\min N_s}}{p_{\min N_s}} \geq \frac{u_{r+1}}{p_{r+1}}$, bem como $\frac{u_r}{p_r} \geq \frac{u_{\max N_{s-1}}}{p_{\max N_{s-1}}} \geq \frac{u_{\min N_s}}{p_{\min N_s}} \geq \frac{u_{\min N_{s+1}}}{p_{\min N_{s+1}}}$, segue que $u_r - \lambda_{r+1}^{s+1} p_r \geq 0$. □

Lema 3

Se r+1 é o primeiro índice de N_s, contendo pseudo-intersecção com N_{s-1}, então $Z_1 < Z_2 \ldots < Z_M$.

Prova De fato, $\frac{u_r}{p_r} = \frac{u_{\max N_{s-1}}}{p_{\max N_{s-1}}} < \frac{u_{\min N_s}}{p_{\min N_s}} = \frac{u_{r+1}}{p_{r+1}}$, daí $u_r - \lambda_{r+1}^{s+1} p_r < 0$ □

Corolário 1

Se as hipóteses do lema 1 (ou do lema 2) forem satisfeitas e $Z_1 < \hat{z} + 1$, então as ramificações associadas aos limitantes $Z_1, Z_2, \ldots, Z_M$ podem ser podadas da árvore.

Corolário 2

Se as hipóteses do lema 3 forem satisfeitas e $Z_M < \hat{z}+1$, então as ramificações associadas aos limitantes $Z_1, Z_2, \ldots, Z_M$ podem ser podadas da árvore.

A seguir apresentamos um método *branch-and-bound* que resolve o problema (38.1 – 38.5):

***Branch-and-Bound* para o PMC 0-1 proposto por HOTO *et al.* (1999)**

1. Inicialização
   Ordene os itens segundo (34.2);
   Ordene os compartimentos segundo (39.2);
   Faça r := 0 e z := 0;
   Para j := 1 até n faça $\alpha_j := 0$;

2. Ramificação
   Para j := r + 1 até n faça ;
   calcule β_j e faça $a_j := \beta_j$;
   calcule α_j;
   Se $(a_j < \alpha_j)$
   então $a_j := 0$ e $\alpha_j := 0$;
   senão Se $(j \neq \max N_s \mid j \in N_s)$ então $\alpha_j := 0$;
   Fim para
   r := n;

3. Atualizar Solução
   Se $\sum_{j \in V_1} \left(\left(\sum_{i \in N_1} u_i a_{ij} \right) - \gamma_1 \right) y_j + \cdots + \sum_{j \in V_k} \left(\left(\sum_{i \in N_k} u_i a_{ij} \right) - \gamma_k \right) y_j > z$
   então atualize a solução a e o valor de z;

4. Retorno
   Se r := 1 então PARE senão r := r - 1;
   Se $a_r := \alpha_r$
   então Reinicie o passo 4;
   senão $a_r := a_r - 1$;
   Calcule Z_1 e Z_M;
   Se for possível efetuar uma poda pelo corolário 1 ou 2
   então Reinicie o passo 4;
   senão Retorne ao passo 2;

A título de ilustração considere o exemplo a seguir, onde o custo por utilizar um compartimento é nulo:

$$\text{maximizar} \quad 3a_1 + 2a_2 + 6a_3$$

sujeito a:

$$15 \leq 4a_1 + 8a_2 \leq 31$$

$$15 \leq 9a_3 \leq 31$$

$$4a_1 + 8a_2 + 9a_3 \leq 42$$

$$a_1, a_2, a_3 \geq 0 \text{ e inteiros}$$

Uma solução do problema é (6, 0, 2) com objetivo igual a 30, observe que o primeiro compartimento ficou definido com capacidade 24 e o segundo com capacidade 18, de modo que a mochila foi totalmente ocupada. Na figura 17 podemos observar as soluções viáveis do problema. Na figura 18 está ilustrada a árvore com as podas efetuadas pelo algoritmo que acabamos de descrever.

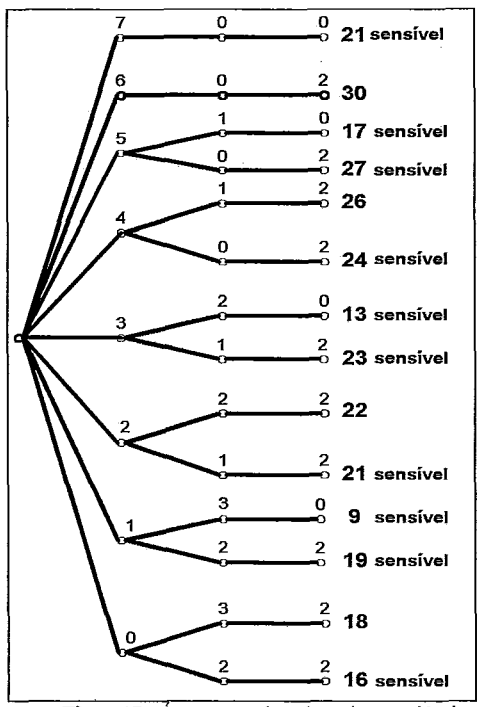

Figura 17 - Árvore completa das soluções viáveis.

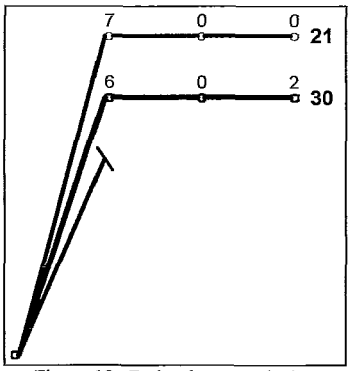

Figura 18 - Podas de ramos da árvore.

7 – O Problema da Mochila Compartimentada Restrito

O caso restrito do PMC ocorre quando é preciso limitar o número total de cada tipo de item a compor a mochila. Vejamos o modelo matemático:

$$\text{maximizar} \quad z = \sum_{j \in V_1} \left(\left(\sum_{i \in N_1} u_i a_{ij} \right) - \gamma_j \right) y_j + \cdots + \sum_{j \in V_k} \left(\left(\sum_{i \in N_k} u_i a_{ij} \right) - \gamma_j \right) y_j \qquad (43.1)$$

sujeito a:

$$\sum_{j \in V_1} \sum_{i \in N_1} p_i a_{ij} y_j + \cdots + \sum_{j \in V_k} \sum_{i \in N_k} p_i a_{ij} y_j \leq c \qquad (43.2)$$

$$d_s^{min} \leq \sum_{i \in N_s} p_i a_{ij} \leq d_s^{max}, \; j \in V_s, \; s = 1, \ldots, k \qquad (43.3)$$

$$\sum_{j \in V_1} a_{ij} y_j + \ldots + \sum_{j \in V_k} a_{ij} y_j \leq b_i, \; i \in N \; \{\textit{limita o total de itens do tipo i na mochila}\} \qquad (43.4)$$

$$\sum_{i \in N_s} a_{ij} \leq g_j, \; j \in V_s, \; s = 1, \ldots, k \; \{\textit{limita o total de itens num compartimento}\} \qquad (43.5)$$

$$\sum_{j \in V} y_j \leq f, \; V = V_1 \cup \cdots \cup V_k \; \{\textit{limita o total de compartimentos na mochila}\} \qquad (43.6)$$

$$a_{ij}, y_j \geq 0 \text{ e inteiros}, \quad i \in N = N_1 \cup \cdots \cup N_k, \; j \in V = V_1 \cup \cdots \cup V_k \qquad (43.7)$$

A condição (43.4) restringe o número total de itens do tipo i na mochila, a condição (43.5) restringe o número total de itens no compartimento j, finalmente (43.6) restringe o número de compartimentos na mochila, onde b_i, g_j e f são inteiros positivos que permitem a existência de soluções para o problema.

O PMC restrito aparece durante a resolução do PCBA e sua resolução não é evidente. Para simplificar a construção de padrões compartimentados restritos, relaxaremos as condições (43.5) e (43.6) (g_j, $f \to +\infty$), e suporemos que a variável y_j assumirá valor 1 ou 0, conforme o compartimento j é ou não escolhido para compor a mochila teremos a seguinte Mochila Compartimentada Restrita:

$$\text{maximizar} \quad z = \sum_{j \in V_1} \sum_{i \in N_1} u_i a_{ij} y_j + \cdots + \sum_{j \in V_k} \sum_{i \in N_k} u_i a_{ij} y_j \qquad (44.1)$$

sujeito a:

$$\sum_{j \in V_1} \sum_{i \in N_1} p_i a_{ij} y_j + \cdots + \sum_{j \in V_k} \sum_{i \in N_k} p_i a_{ij} y_j \le c \qquad (44.2)$$

$$d_s^{min} \le \sum_{i \in N_s} p_i a_{ij} \le d_s^{max}, \; j \in V_s, \; s = 1, \ldots, k \qquad (44.3)$$

$$\sum_{j \in V_1} a_{ij} y_j + \ldots + \sum_{j \in V_k} a_{ij} y_j \le b_i, \; i \in N \qquad (44.4)$$

$$y_j \in \{0, 1\}, \; a_{ij} \ge 0 \text{ e inteiros}, \quad i \in N = N_1 \cup \cdots \cup N_k, \; j \in V = V_1 \cup \cdots \cup V_k \qquad (44.5)$$

Nossa proposta é utilizar as idéias dos algoritmos COMPEX e COMPMT para resolver este caso do PMC restrito. Note que a restrição (44.4) impõe uma dependência entre os compartimentos, que deve ser respeitada durante a primeira etapa da decomposição do problema, onde são obtidos os valores de utilidades dos compartimentos. Resumindo, no momento em que a utilidade de um compartimento está sendo definida, deve-se observar os compartimentos que já foram examinados, pois, existe agora uma restrição com respeito ao número de itens permitido na mochila.

No passo 2 do COMPEX (primeira etapa da decomposição) a utilidade de cada compartimento construtivo j de capacidade w_j, onde $N_{s(j)}$ é seu agrupamento associado, é obtida por meio da mochila (35.1 – 35.3). Para o caso do PMC restrito que propomos resolver, a seguinte mochila restrita deverá ser considerada:

$$\text{maximizar} \quad v_j = \sum_{i \in N_{s(j)}} u_i a_{ij} \qquad (45.1)$$

sujeito a:

$$\sum_{i \in N_{s(j)}} p_i a_{ij} \le w_j \qquad (45.2)$$

$$0 \le a_{ij} \le \bar{b}_i, \; i \in N_{s(j)} \qquad (45.3)$$

$$a_{ij} \text{ inteiro}, \; i \in N_{s(j)} \qquad (45.4)$$

Na mochila (45.1 – 45.4), os valores iniciais de $\bar{b}_i$ são exatamente os valores dos limites b_i da mochila (44.1 – 44.5). A medida que os compartimentos são construídos deve-se atualizar os valores de $\bar{b}_i$, tendo em vista os compartimentos já construídos.

No passo 3 do COMPEX (segunda etapa da decomposição) os compartimentos são escolhidos por meio da mochila (36.1 – 36.3). No caso do PMC restrito sugerimos resolver a mesma mochila, porém, a variável y deverá ser binária (0 ou 1). Vejamos o algoritmo para o PMC restrito:

COMPREST – Algoritmo para Compartimentação Restrito

1. Inicialização
   1.1 forneça os dados da mochila: c e (u_i, p_i) para $i \in N$;
   1.2 forneça os dados da compartimentação: $(N_s, d_s^{min}, d_s^{max})$ para $s = 1, \ldots, k$;

2. Determinação das Utilidades dos Compartimentos
   2.1 para s := 1 até k faça
   2.2 ordene os itens: $\frac{u_i}{p_i} \geq \frac{u_{i+1}}{p_{i+1}}$ para $i, i+1 \in N_s$;
   2.3 faça $\bar{b}_i := b_i, i \in N_{s(j)}$;
   2.4 resolva a mochila (45.1 – 45.4) com $w_j = d_s^{max}$,
   pelo algoritmo de Yanasse-Soma adaptado ao caso restrito;
   2.5 faça $w_j := d_s^{min}$;
   2.6 enquanto $w_j \leq d_s^{max}$ faça
   2.7 se o compartimento j de capacidade w_j é construtivo
   então recupere o objetivo v_j da mochila (45.1 – 45.4) e calcule γ_j, $j \in W_s$;
   fim se 2.7
   2.8 se a solução da mochila é nula
   então faça $w_j := d_s^{max} + 1$; {examinar o próximo agrupamento}
   senão atualize os limites $\bar{b}_i := \bar{b}_i - a^*_{ij}, i \in N_{s(j)}$;
   fim se 2.8;
   2.9 faça $w_j := w_j + 1$;
   fim enquanto 2.6;
   fim para 2.1;

3. Determinação da Compartimentação
   3.1 construa o subconjunto W' dos índices dos compartimentos dominantes, de modo que, $W' = \{1, \ldots, p'\}$;
   3.2 ordene os compartimentos segundo as eficiências: $\frac{v_j}{w_j} \geq \frac{v_{j+1}}{w_{j+1}}$ para $j, j+1 \in W'$;
   3.3 resolva a mochila (36.1 – 36.3) e encontre y^*, porém, **a variável y deve ser 0-1**;

4. Construção dos Compartimentos Escolhidos
   4.1 para $j := 1$ até p' faça
      4.2 se $y_j^* \neq 0$ então
         4.3 identifique o agrupamento N_s associado ao compartimento escolhido;
         4.4 recupere a solução da mochila (45.1 – 45.4);
         4.5 armazene o compartimento construído;
      fim se 4.2;
   fim para 4.1;

O COMPREST é um procedimento heurístico, embora, a mochila (45.1 – 45.4) seja computada no passo 2. Como no COMPMT, podemos utilizar os limitantes de Martello-Toth para mochilas restritas, (MARTELLO e TOTH, 1990, pag 85), calculando limitantes para as utilidades dos compartimentos. Estes dois procedimentos foram implementados.

No apêndice E há pseudocódigos de dois algoritmos que podem resolver o problema (45.1 – 45.3). O primeiro deles é de um guloso, o segundo é de uma versão modificada do algoritmo de Yanasse-Soma.

O PMC Restrito tem sido o foco de trabalho de outros autores (MARQUES, 2000) e (MARQUES e ARENALES, 2001), onde é admitida a existência de um particular agrupamento N_0, cujos compartimentos associados não apresentam limitações em relação às suas capacidades. Mais precisamente, para um compartimento $j \in V_0$ formado com itens deste agrupamento adicional N_0, não é exigida a condição (44.3). Os autores desenvolveram três heurísticas para este problema. A primeira delas é denominada Heurística de Decomposição:

Heurística de Decomposição

Passo 1

1. Para cada agrupamento $s = 1, \ldots, k$, construa um compartimento segundo a mochila:

$$\text{maximizar} \quad v_s = \sum_{i \in N_s} u_i a_i$$

sujeito a:

$$\sum_{i \in N_s} p_i a_i \leq d_s^{max}$$

$$0 \leq a_i \leq b_i \text{ e inteiro}, i \in N_s$$

Defina a capacidade do compartimento como $w_s = \sum_{i \in N_s} p_i a_i^*$;

2. Para cada $j \in N_0$ faça $v_j := u_j$ e $w_j := p_j$;

3. Faça $m := k + |N_0|$;

Passo 2

Resolva o seguinte problema da mochila restrito para obter a compartimentação:

$$\text{maximizar} \quad \sum_{j=1}^{m} v_j y_j$$

sujeito a:

$$\sum_{j=1}^{m} w_j y_j \leq c$$

$$a_i^* y_j \leq b_i, i \in N_j, j = 1, \ldots, k$$

$$0 \leq y_j \leq b_j, j \in N_0$$

$$y_j \text{ inteiro}, j = 1, \ldots, m$$

A terminologia adotada nas mochilas das heurísticas de Marques-Arenales é a mesma que temos usado até aqui, isto é, a variável a_i representa o número de itens do tipo i num compartimento j de capacidade w_j, e a variável y_j o número de vezes que o compartimento j aparece na mochila, porém, os autores consideram o agrupamento N_0 que já mencionamos.

Observe no item 1 do passo 1 que são construídos k compartimentos, um de cada agrupamento $s = 1, \ldots, k$. No item 2 do mesmo passo, cada item do agrupamento adicional N_0 é visto como um compartimento. No passo 2, todos estes compartimentos são considerados na resolução da mochila. ressaltamos que o agrupamento adicional N_0 pode ser considerado nos procedimentos COMPEX, COMPMT e COMPREST, basta atribuir convenientes valores aos limites $d_0^{min} = \min_i \{p_i \,|\, i \in N_0\}$ e $d_0^{max} = c$.

A próxima heurística de Marques-Arenales é denominada Heurística do Melhor Compartimento, sua filosofia é similar a anterior, porém, a mochila do passo 2 é substituída por um procedimento aproximativo, vejamos:

Heurística do Melhor Compartimento

Passo 1

1. Faça $c_{util} := c$;
2. Para cada agrupamento $s = 1, \ldots, k$:

   2.1 Se $d_s^{min} \le c_{util} < d_s^{max}$ então $w_{comp} := c_{util}$;

   senão Se $c_{util} \ge d_s^{max}$ então $w_{comp} := d_s^{max}$;

   senão $w_{comp} := c_{util}$;

   2.2 Se $w_{comp} \ge d_s^{min}$ então construa um compartimento segundo a mochila:

$$\text{maximizar} \quad v_s = \sum_{i \in N_s} u_i a_i$$

sujeito a:

$$\sum_{i \in N_s} p_i a_i \le w_{comp}$$

$$0 \le a_i \le b_i \text{ e inteiro}, i \in N_s$$

Defina a capacidade do compartimento como $w_s = \sum_{i \in N_s} p_i a_i^*$

senão $v_s := 0$ e $w_s := w_{comp}$, $s = 1, \ldots, k$;

3. Para cada $j \in N_0$ faça
   Se a demanda do item não é nula <u>então</u> $v_j := u_j$ e $w_j := p_j$;
   <u>senão</u> $v_j := 0$ e $w_j := p_j$;
4. Faça $m := k + |N_0|$;

<u>Passo 2</u>

1. Ordene a lista de compartimentos da seguinte forma: $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \cdots \geq \frac{v_m}{w_m}$;
2. Seja $N_{s(1)}$ o agrupamento associado ao primeiro compartimento ordenado;
3. Adicione o primeiro compartimento da lista na mochila compartimentada;
4. Atualize os limites b_i dos itens que compõem o compartimento adicionado;
5. Exclua o compartimento adicionado da lista ordenada no item 1 deste passo;
6. Faça $c_{util} = c_{util} - w_1$;
7. Se $c_{util} \geq \min_{s=1}^{m} \{w_s\}$ <u>então</u> retorne ao item 2 do passo 1;
   <u>senão</u> PARE;

A terceira heurística de Marques-Arenales (MARQUES e ARENALES, 2001) é denominada Heurística dos z-melhores Compartimentos. Ela obedece aos mesmos passos da Heurística de Decomposição, entretanto, são computadas as z-melhores soluções das mochilas do passo 1, de modo que, no passo 2 são analisados (z.k) compartimentos.

7.1 – Desempenho da Heurística de Decomposição de Marques-Arenales

Segundo relatam MARQUES e ARENALES (2000, 2001) a Heurística de Decomposição foi a que apresentou melhores resultados diante as outras duas desenvolvidas pelos mesmos autores. Tendo em vista esta observação, decidimos comparar o desempenho desta heurística perante os algoritmos COMPMT e COMPEX descritos nesta tese, para isso, adotamos valores arbitrariamente altos para os limites superiores b_i, $i \in N_s$, $s = 1, \ldots, k$, isto é, programamos a Heurística de Decomposição para resolver Mochilas Compartimentadas Irrestritas. Nos exemplos gerados desconsideramos a existência do particular agrupamento N_0.

Na tabela 7 apresentamos os resultados numéricos que obtivemos para exemplos gerados aleatoriamente. O custo pela utilização de um compartimento foi considerado como nulo, e a tabela está organizada da seguinte maneira:

d^{min} e d^{max} : *respectivamente, o limite mínimo e máximo dos capacidades dos compartimentos*;

Agrup : *número de agrupamentos da compartimentação*;

Itens : *número de itens num agrupamento da compartimentação*;

Tempo : *tempo médio de execução de um exemplo*;

Perda : *percentual de perda (espaço ocioso) na mochila*;

Obj : *valor do objetivo da mochila*;

c = 1200				COMPEX			COMPMT			Decomposição		
d^{min}	d^{max}	*Agrup*	*Itens*	*Tempo*	*Perda*	*Obj*	*Tempo*	*Perda*	*Obj*	*Tempo*	*Perda*	*Obj*
31	40	3	5	0,050 seg	---	7400	0,050 seg	---	7400	0,030 seg	1,33%	7400
			40	0,060 seg	---	9900	0,050 seg	---	9900	0,037 seg	1,00%	9900
			100	0,110 seg	---	9900	0,060 seg	---	9200	0,041 seg	1,00%	9900
		20	5	0,220 seg	---	9900	0,050 seg	---	9900	0,047 seg	1,00%	9900
			40	0,270 seg	---	9900	0,060 seg	---	9200	0,055 seg	---	9200
			100	0,270 seg	---	9900	0,110 seg	---	9200	0,069 seg	---	6643
51	100	3	5	0,050 seg	---	5600	0,050 seg	---	5400	0,028 seg	2,00%	5600
			40	0,110 seg	---	10000	0,050 seg	---	10000	0,034 seg	4,00%	9600
			100	0,160 seg	---	10000	0,050 seg	---	9700	0,039 seg	4,00%	9600
		20	5	0,330 seg	---	10000	0,170 seg	---	10000	0,070 seg	4,00%	9600
			40	0,550 seg	---	10000	0,330 seg	---	8511	0,076 seg	4,00%	9600
			100	0,550 seg	---	10000	0,330 seg	---	9700	0,086 seg	---	4855
151	450	3	5	0,160 seg	---	1500	0,060 seg	---	1500	0,029 seg	5,00%	1500
			40	0,330 seg	---	10000	0,280 seg	---	10000	0,033 seg	26,00%	7400
			100	0,390 seg	---	8500	0,270 seg	---	8500	0,035 seg	25,33%	6400
		20	5	0,990 seg	---	37	0,500 seg	---	3300	0,067 seg	25,33%	2800
			40	5,110 seg	---	100	3,350 seg	---	10000	0,073 seg	26,00%	7400
			100	5,270 seg	---	100	3,400 seg	---	10000	0,084 seg	26,00%	7400

Tabela 7 – Desempenho da Heurística de Decomposição.

De acordo com os resultados da tabela 7 observamos que a Heurística de Decomposição é bastante veloz, porém, seu comportamento não foi satisfatório em alguns casos, visto que, ela está fortemente baseada na existência do particular agrupamento N_0 (veja final da página 91), onde seus resultados são melhores.

Neste capítulo apresentamos uma nova modalidade de mochila que denominamos Mochila Compartimentada. Apresentamos sua formulação matemática e descrevemos dois métodos de resolução. O primeiro resultou no algoritmo COMPEX que encontra uma solução ótima do problema, o segundo resultou no algoritmo COMPMT que se trata de uma alternativa heurística para resolver o problema.

Apresentamos também a Mochila Compartimentada 0-1 e descrevemos um algoritmo *branch-and-bound* para este caso, onde destacamos o cálculo de limitantes superiores.

Por fim, apresentamos o caso restrito que tem sido alvo de estudos comuns com outros autores. Este caso não é de resolução evidente e algumas heurísticas têm sido propostas e ainda estão sob aprimoramento, destacamos três procedimentos de MARQUES e ARENALES (2001) e o algoritmo COMPREST que é uma nova proposta para resolver o problema.

Somos todos filhos da mesma nação, da mesma união,
da mesma paixão, uns com mais e outros com menos coração.

Capítulo 5

O PMC Aplicado no PCBA

1 – Introdução

Como mencionamos no capítulo 2 a abordagem que adotamos para a resolução do PCBA é a Técnica de Geração de Colunas de Gilmore-Gomory, apêndice B. O modelo matemático (20.1 – 20.3) é o que utilizamos para o problema e ele foi apresentado na página 45 do capítulo 2, entretanto, também passamos a examinar as limitações do estoque, de modo que, introduzimos uma disponibilidade e_r para cada tipo de bobina $r = 1, \ldots, m$ como no modelo (11.1 - 11.4) do capítulo 1. O modelo matemático que estudamos para resolver o PCBA é escrito a seguir:

Modelo Matemático do PCBA

$$\text{minimizar} \quad \sum_{r=1}^{m}\sum_{j=1}^{p_r}\left(c^r_{aço}\, T^r_j + \sum_{h=1}^{H_{jr}} (c^r_{hj} y^r_{hj}) \right) x^r_j \qquad (46.1)$$

sujeito a:

$$\sum_{r=1}^{m}\sum_{j=1}^{p_r} \frac{P^r}{L^r} \ell_i\, a^r_{ij}\, x^r_j = d_i,\ i = 1, \ldots, n \qquad (46.2)$$

$$\begin{aligned} \sum_{j=1}^{p_1} x^1_j &\le e_1 \\ \ddots \quad &\vdots \\ \sum_{j=1}^{p_m} x^m_j &\le e_m \end{aligned} \qquad (46.3)$$

$$x^r_j \ge 0 \text{, e inteiro } r = 1, \ldots, m,\ j = 1, \ldots, p_r \qquad (46.4)$$

Nosso objetivo é minimizar os custos associados às perdas lineares de aço numa bobina, e custos por utilização de compartimentos, além de examinar as possibilidades de utilização do estoque, onde a demanda é dada em "Kg". O modelo (46.1 – 46.4) é mais bem compreendido com as definições dadas a seguir:

Dados das Bobinas

- P^r "peso" (a massa) da bobina (objeto) $r = 1, \ldots, m$ **(Kg)**;
- L^r largura da bobina (objeto) $r = 1, \ldots, m$ **(mm)**;
- e_r disponibilidade da bobina (objeto) $r = 1, \ldots, m$ **(unidades de bobinas)**;
- $c^r_{aço}$ custo linear do aço da bobina (objeto) $r = 1, \ldots, m$ **($/unidade de comprimento)**;
- T^r_j perda linear de aço na bobina (objeto) $r = 1, \ldots, m$, segundo o padrão compartimentado j **(unidades de comprimentos)**;

Dados das Fitas

- ℓ_i largura da fita (item) $i = 1, \ldots, n$ **(mm)**;
- d_i demanda solicitada da fita (item) $i = 1, \ldots, n$ **(Kg)**;

Dados das Bobinas Intermediárias

- c^r_{hj} custo operacional da bobina intermediária (compartimento) h na bobina r, segundo o padrão compartimentado j **($/unidade do compartimento h)**;
- H_{jr} número total de bobinas intermediárias (compartimentos) na bobina r, segundo o padrão compartimentado j **(unidades de compartimentos)**;

Variáveis

- y^r_{hj} número de bobinas intermediárias (compartimentos) h na bobina r, segundo o padrão compartimentado j **(unidades de compartimentos)**;
- x^r_j número de bobinas utilizadas do tipo r, segundo o padrão compartimentado j **(unidades de bobinas)**;
- a^r_{ij} número de fitas (itens) tipo i na bobina r, segundo o padrão compartimentado j **(unidades de fitas)**;

Para facilitar cálculos e notações, sejam $c_j^r = c_{aço}^r T_j^r + \sum_{h=1}^{H_{jr}} (c_{hj}^r y_{hj}^r)$ e $\alpha_{ij}^r = \frac{P^r}{L^r} \ell_i a_{ij}^r$. Na figura 19 ilustramos a estrutura matricial do modelo, onde está representada a matriz A e os vetores c e d, e onde $A_r = (\alpha_{ij}^r)_{n \times p_r}$, $r = 1, \ldots, m$. Ressaltamos que p_r pode chegar a vários milhões ou bilhões. Para cada bobina do tipo r examinaremos os padrões compartimentados, conforme a Técnica de Geração de Colunas.

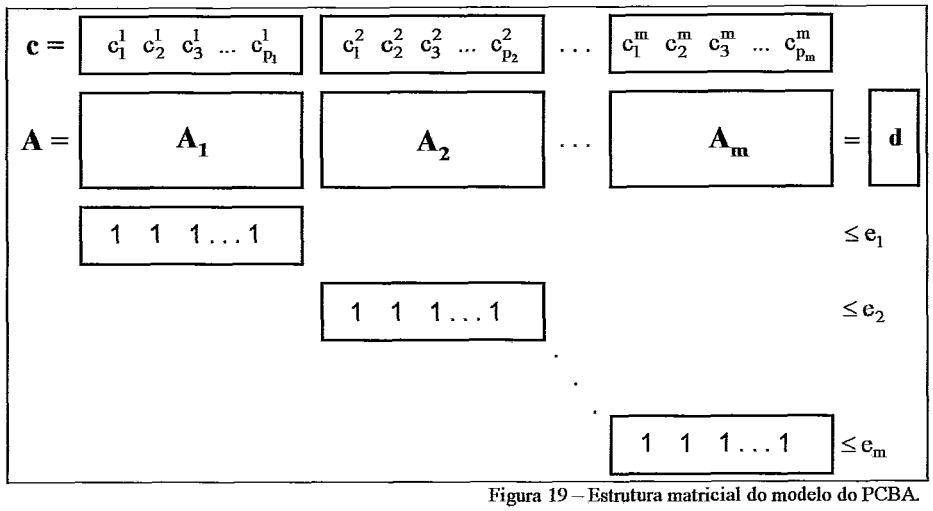

Figura 19 – Estrutura matricial do modelo do PCBA.

Observe que uma coluna do modelo do PCBA (46.1 – 46.4) possui (n + m) componentes e tem o seguinte aspecto $\alpha_j^r = \left(\alpha_{1j}^r, \alpha_{2j}^r, \ldots, \alpha_{nj}^r, 0, \ldots, 0, 1, 0, \ldots, 0\right)^T$, onde o valor unitário ocupa a posição (n + r) da coluna. Para determinarmos α_{ij}^r precisamos determinar a_{ij}^r e isso é feito por meio do PMC como já iremos mostrar, contudo, antes de prosseguir apresentamos o algoritmo Primal-Simplex para ajudar na compreensão da metodologia que usamos ao resolver o PCBA.

Algoritmo Primal-Simplex para Minimização

Fase 1
Encontre uma partição básica de A = (B, N) primal-viável.
{a matriz B é uma matriz quadrada e inversível}
{serão necessárias variáveis de folga para obter B*}*
{pode ser necessário utilizar variáveis artificiais para obter B*}*

Fase 2
Faça PARE = falso.
Enquanto PARE = falso faça:

1. Calcule o vetor linha $\pi = c_B B^{-1} = (\pi_1, \ldots, \pi_n, \pi_{n+1}, \ldots, \pi_{n+r}, \ldots, \pi_{n+m})$ dos multiplicadores Simplex.

2. Calcule a solução básica $x_B = B^{-1}d$.

3. Determine uma coluna de índice k de N para entrar na base (na matriz básica B).

   Isto é feito escolhendo aquela de menor custo relativo:

   $$c_k - \pi\alpha_k = \text{mínimo } \{ c_j - \pi\alpha_j \mid \alpha_j \text{ é uma coluna arbitrária de N}\}$$

   Teste a Otimalidade (minimização):
   Se $c_k - \pi\alpha_k \geq 0$ então PARE = verdade. *{a corrente solução é ótima}*
   Senão vá para o passo 4.

4. Determine a Direção Simplex $y_k = B^{-1}a_k$.
   Se $y_k < 0$ então PARE = verdade. *{o problema não tem solução ótima finita}*
   Senão vá para o passo 5.

5. Determine a coluna de índice ℓ de B para sair da base (α_ℓ será substituída por α_k).
   Isso é feito da seguinte forma: $\dfrac{x_{B(\ell)}}{y_\ell} = \text{mínimo} \left\{ \dfrac{x_{B(i)}}{y_i} \mid y_i > 0, i = 1, \ldots, n \right\}$.

6. Atualize a matriz básica B (coloque α_k no lugar de α_ℓ), e atualize B^{-1}.

O que inviabiliza utilizarmos o algoritmo anterior para resolver o PCBA é o exame que devemos fazer nas colunas da matriz N (passo 3), pois seu número de colunas é extremamente alto (vários milhões ou bilhões).

A idéia introduzida por GILMORE e GOMORY (1961, 1963) consiste em determinar a coluna α_k do passo 3, por meio de um problema de otimização, apêndice B. Para escrevermos este problema, primeiramente vamos investigar a natureza dos custos relativos para cada tipo de bobina $r = 1, \ldots, m$ do estoque, cuja expressão geral é dada por:

$$c_j^r - \pi\alpha_j^r = c_j^r - \left(\sum_{i=1}^{n} \pi_i \alpha_{ij}^r + \pi_{n+r}\right) =$$

$$= c_{aço}^r \, T_j^r + \sum_{h=1}^{H_{jr}} (c_{hj}^r y_{hj}^r) - \left(\sum_{i=1}^{n} \pi_i \alpha_{ij}^r + \pi_{n+r}\right) =$$

$$= c_{aço}^r \left((L^r - S_1) - (\sum_{i=1}^{n} \ell_i a_{ij}^r) - (S_2 \sum_{h=1}^{H_{jr}} y_{hj}^r) \right) + \left(\sum_{h=1}^{H_{jr}} c_{hj}^r y_{hj}^r \right) - \left(\sum_{i=1}^{n} \pi_i \alpha_{ij}^r + \pi_{n+r} \right)$$

Reorganizando o cálculo anterior, onde S_1 e S_2 são perdas técnicas, temos:

$$c_j^r - \pi\alpha_j^r =$$

$$= -\sum_{i=1}^{n} (c_{aço}^r \ell_i a_{ij}^r + \pi_i \alpha_{ij}^r) + \sum_{h=1}^{H_{jr}} (c_{hj}^r - c_{aço}^r S_2) y_{hj}^r + \left(c_{aço}^r (L^r - S_1) - \pi_{n+r} \right)$$

Substituindo $\alpha_{ij}^r = \frac{P^r}{L^r} \ell_i \, a_{ij}^r$ na expressão anterior obtemos:

$$c_j^r - \pi\alpha_j^r =$$

$$= -\sum_{i=1}^{n} (c_{aço}^r + \pi_i \frac{P^r}{L^r}) \ell_i a_{ij}^r + \sum_{h=1}^{H_{jr}} (c_{hj}^r - c_{aço}^r S_2) y_{hj}^r + \left(c_{aço}^r (L^r - S_1) - \pi_{n+r} \right)$$

Observe que podemos escrever $a_{ij}^{r} = \sum_{h=1}^{H_{jr}} a_{ih}^{jr} y_{hj}^{r}$, onde a_{ih}^{jr} é o número de itens tipo i que aparecem no compartimento h, do padrão j que é usado para cortar bobinas do tipo r.

A última expressão do custo relativo pode ser rescrita como:

$$c_j^r - \pi\alpha_j^r =$$

$$= -\left(\sum_{i=1}^{n}(c_{aço}^r + \pi_i \frac{P^r}{L^r})\ell_i \sum_{h=1}^{H_{jr}} a_{ih}^{jr} y_{hj}^r\right) + \left(\sum_{h=1}^{H_{jr}}(c_{hj}^r - c_{aço}^r S_2) y_{hj}^r\right) + \left(c_{aço}^r (L^r - S_1) - \pi_{n+r}\right)$$

Finalmente a expressão do custo relativo é escrita da seguinte forma:

$$c_j^r - \pi\alpha_j^r = -\left(\sum_{h=1}^{H_{jr}}\left((\sum_{i=1}^{n} u_i a_{ih}^{jr}) - c_h\right) y_{hj}^r\right) + \left(c_{aço}^r (L^r - S_1) - \pi_{n+r}\right) \qquad (47)$$

Onde,

$$u_i = (c_{aço}^r + \pi_i \frac{P^r}{L^r})\ell_i \text{ e } c_h = c_{hj}^r - c_{aço}^r S_2 \qquad (48)$$

Tendo em vista que devemos obter $c_k^r T_k^r - \pi\alpha_k^r$ = mínimo $\{c_j^r T_j^r - \pi\alpha_j^r \,|\, \alpha_j^r$ é uma coluna arbitrária de N$\}$, o problema que teremos de resolver é:

$$\text{minimizar } -\left(\sum_{h=1}^{H_{jr}}\left((\sum_{i=1}^{n} u_i a_{ih}^{jr}) - c_h\right) y_{hj}^r\right) + \left(c_{aço}^r (L^r - S_1) - \pi_{n+r}\right)$$

sujeito a coluna a_k^r de coeficientes a_{ih}^{jr} representar um padrão de corte compartimentado. *(49)*

Note que para cada bobina do tipo r, a parcela $c^r_{aço}(L^r - S_1) - \pi_{n+r}$ de (49) é uma constante. Além do mais, os itens do PCBA são indexados pelo conjunto $N = \{1, \ldots, n\}$, e eles devem ser agrupados, conforme descrito na página 46 do capítulo 2, em subconjuntos $\{N_1, \ldots, N_k\}$, constituindo uma partição de N. Tendo em vista estas observações, o problema que passamos a considerar para gerar uma coluna que representa um padrão compartimentado j, a ser usado para cortar bobinas do tipo $r = 1, \ldots, m$, pode ser escrito como se segue, onde $u_i = (c^r_{aço} + \pi_i \frac{P^r}{L^r})\ell_i$ e $c_h = c^r_{hj} - c^r_{aço} S_2$:

Modelo do Gerador de Padrões Compartimentados do PCBA

$$\text{maximizar} \sum_{h \in V_1} \left(\left(\sum_{i \in N_1} u_i a_{ih} \right) - c_h \right) y_h + \cdots + \sum_{h \in V_k} \left(\left(\sum_{i \in N_k} u_i a_{ih} \right) - c_h \right) y_h \qquad (50.1)$$

sujeito a:

$$\sum_{h \in V_1} (S_2 + \sum_{i \in N_1} \ell_i a_{ih}) y_h + \cdots + \sum_{h \in V_k} (S_2 + \sum_{i \in N_k} \ell_i a_{ih}) y_h \le L - S_1 \qquad (50.2)$$

$$L_{min} - S_2 \le \sum_{i \in N_s} \ell_i a_{ih} \le L_{max} - S_2, \; h \in V_s, \; s = 1, \ldots, k \qquad (50.3)$$

$$\sum_{i \in N_s} a_{ih} \le D_2, \; h \in V_s, \; s = 1, \ldots, k \quad \textit{\{limita o total de itens numa bobina intermediária\}} (50.4)$$

$$\sum_{h \in V} y_h \le D_1, \; V = V_1 \cup \cdots \cup V_k \quad \textit{\{limita o total de bobinas intermediárias no padrão\}} (50.5)$$

$$a_{ih}, y_h \ge 0 \text{ e inteiros}, \quad i \in N = N_1 \cup \cdots \cup N_k, \; h \in V = V_1 \cup \cdots \cup V_k \qquad (50.6)$$

Na página 88 do capítulo anterior apresentamos o modelo (43.1 – 43.7) do Problema da Mochila Compartimentada Restrito, observe que um padrão de corte compartimentado do PCBA deve então ser construído por meio deste problema, porém, sem a condição (43.4).

No capítulo 2 vimos que o número de cortes numa bobina intermediária (compartimento) é limitado por 8, e que o número de cortes numa bobina do estoque é limitado por 12. Assim, o número máximo de itens combinados numa bobina intermediária (compartimento) é $D_2 = 7$, e o número máximo de bobinas intermediárias (compartimentos) combinadas numa bobina do estoque é $D_1 = 11$.

Pela tabela 2, página 45 do capítulo 2, podemos observar que em geral a fita de menor largura é da ordem de 36 mm e a de maior largura 250 mm, além do mais, a largura das bobinas do estoque variam entre 1100 mm e 1200 mm. As perdas técnicas das laterais de uma bobina intermediária é da ordem de 5 mm, daí no modelo $S_2 = 10$. Já as perdas técnicas das laterais de uma bobina é da ordem de 8 mm, no modelo $S_1 = 16$. Em vista das características técnicas do PCBA o número máximo de bobinas intermediárias que podem surgir numa bobina é $\left\lfloor \frac{1200-16}{154} \right\rfloor = 7$, não violando a restrição (50.5). Já o número máximo de fitas (itens) que podem surgir numa bobina intermediária é $\left\lfloor \frac{456-10}{36} \right\rfloor = 12$, podendo violar a restrição (50.4) em cinco unidades.

Vimos no capítulo anterior que o caso restrito do PMC é difícil de ser tratado, de modo que, os procedimentos apresentados para resolver este caso do PMC são todos aproximativos (COMPREST e Heurísticas de Marques-Arenales), além de considerarem apenas a restrição (43.4) e desconsiderar as restrições (43.5) e (43.6). Vimos também que para o caso irrestrito é possível obter uma solução ótima do PMC por meio do COMPEX, ou uma boa solução viável por meio do COMPMT, conforme os resultados numéricos da tabela 6 na página 77.

Visto que a restrição (50.5) nunca é violada, e que a próxima fita de menor largura é em geral da ordem de 65 mm (tabela 2 do capítulo 2, página 45), adotamos o seguinte modelo de Mochila Compartimentada para gerar os padrões compartimentados do PCBA:

$$\text{maximizar} \sum_{h\in V_1}\left(\left(\sum_{i\in N_1} u_i a_{ih}\right)-\gamma_h\right)y_h + \cdots + \sum_{h\in V_k}\left(\left(\sum_{i\in N_k} u_i a_{ih}\right)-\gamma_h\right)y_h \qquad (51.1)$$

sujeito a:

$$\sum_{h\in V_1}(S_2 + \sum_{i\in N_1} \ell_i a_{ih})y_h + \cdots + \sum_{h\in V_k}(S_2 + \sum_{i\in N_k} \ell_i a_{ih})y_h \le L - S_1 \qquad (51.2)$$

$$L_{min} - S_2 \le \sum_{i\in N_s} \ell_i a_{ih} \le L_{max} - S_2,\ h \in V_s,\ s = 1, \ldots, k \qquad (51.3)$$

$$a_{ih},\ y_h \ge 0 \text{ e inteiros},\quad i \in N = N_1 \cup \cdots \cup N_k,\ h \in V = V_1 \cup \cdots \cup V_k \qquad (51.4)$$

2 – Encontrando uma Solução Básica Inicial para o PCBA

Não consiste uma tarefa muito difícil encontrar uma matriz básica B para iniciar o processo de Geração de Colunas do PCBA, apenas é preciso observar com cuidado alguns pontos que passaremos a descrever.

Observe de início que uma variável de folga $\mu_r \ge 0$, $r = 1, \ldots, m$, será necessária em cada uma das restrições de disponibilidade do estoque (46.3) que passam a ter o seguinte aspecto:

$$\begin{aligned} \sum_{j=1}^{p_1} x_j^1 + \mu_1 &= e_1 \\ \ddots \qquad &\vdots \\ \sum_{j=1}^{p_m} x_j^m + \mu_m &= e_m \end{aligned} \qquad (52)$$

Agora vejamos como podemos construir uma base inicial. Sem perda de generalidade vamos descrever esta base para bobinas do tipo 1, assim para cada item (fita) $j = 1, \ldots, n$ considere a seguinte coluna associada a um padrão compartimentado homogêneo (formado por apenas um tipo de item), onde o valor unitário ocupa a posição (n+1):

$$(0, \ldots, a_{jj}, \ldots, 0, 1, 0, \ldots, 0)^T,\ a_{jj} = \left\lfloor \frac{L^1 - S_1}{L_{max}} \right\rfloor \cdot \left\lfloor \frac{L_{max} - S_2}{\ell_j} \right\rfloor \qquad (53)$$

Se $x_j^1 = \frac{d_j}{a_{jj}}$, $j = 1, \ldots, n$, é tal que $\sum_{j=1}^{n} x_j^1 \leq e_1$, então temos uma solução viável e a matriz básica inicial B, de ordem (n+m), pode ser escrita como:

$$B = \begin{bmatrix} a_{11} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{nn} & 0 & \cdots & 0 \\ 1 & \cdots & 1 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix} \qquad (54)$$

Observe que a coluna associada a variável de folga μ_r tem o seguinte aspecto $(0, \ldots, 0, 0, \ldots, 1, \ldots, 0)^T$, onde o valor unitário ocupa a posição (n+r).

O custo relativo associado a uma variável de folga é mais simples que aquele associado a uma coluna gerada e que representa um padrão de corte compartimentado. Ele é obtido por meio do seguinte cálculo:

$$0 - (\pi_1, \ldots, \pi_n, \pi_{n+1}, \ldots, \pi_{n+r}, \ldots, \pi_{n+m}) \cdot (0, \ldots, 0, 0, \ldots, 1, \ldots, 0)^T = -\pi_{n+r} \qquad (55)$$

Caso $\sum_{j=1}^{n} x_j^1 > e_1$, então a solução é inviável e neste caso será necessário resolver um Problema Artificial (Fase 1 – Simplex). Para escrevermos este problema observamos que μ_1 será uma variável artificial, de modo que, $\sum_{j=1}^{n} x_j^1 - \mu_1 = e_1$. As demais restrições de disponibilidade do estoque continuam com suas respectivas variáveis de folga.

O Problema Artificial que resolveremos na Fase 1 do Simplex com Geração de Colunas é dado por:

$$\text{minimizar} \quad \mu_1 \tag{56.1}$$

sujeito a:

$$\sum_{r=1}^{m}\sum_{j=1}^{p_r}\frac{P^r}{L^r}\ell_i\, a_{ij}^r x_j^r = d_i,\ i=1,\ldots,n \tag{56.2}$$

$$\begin{aligned} \sum_{j=1}^{p_1} x_j^1 + \mu_1 &= e_1 \\ \ddots \qquad &\vdots \\ \sum_{j=1}^{p_m} x_j^m + \mu_m &= e_m \end{aligned} \tag{56.3}$$

$$x_j^r,\ y_r \geq 0\text{, e inteiro } r=1,\ldots,m,\ j=1,\ldots,p_r \tag{56.4}$$

A base inicial do Problema Artificial tem o seguinte aspecto:

$$B=\begin{bmatrix} a_{11} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{nn} & 0 & \cdots & 0 \\ 1 & \cdots & 1 & -1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{57}$$

Com respeito ao problema artificial (56.1 – 56.4), se $\mu_1 > 0$ significa que não existe uma solução básica inicial para o modelo do PCBA. Se por outro lado $\mu_1 = 0$ a solução ótima do problema artificial fornece uma base inicial para o PCBA, mas observe, caso a base final da solução ótima do problema artificial não contenha a coluna associada a variável artificial μ_1, está será uma base primal viável para o PCBA. Se em contrapartida a coluna associada a variável artificial μ_1 fizer parte da base final da solução ótima do problema artificial (degeneração), o modelo do PCBA ainda pode ser inicializado com esta base, porém, deve-se tomar o cuidado de não permitir que a variável μ_1 associada tenha valor diferente de zero.

3 – Obtendo a Matriz Inversa da Corrente Base

Para obtermos a matriz B^{-1} da corrente matriz básica B efetuamos sua atualização a partir da matriz inversa do passo anterior. Este processo de atualização é bem conhecido e está descrito no apêndice B desta tese. Quando a atualização fornecer uma matriz inversa com grande acúmulo de erros, recomendamos utilizar algum método numérico que calcule a matriz inversa.

4 – Arredondamento da Solução do PCBA

Na seção 3 do capítulo 2 apresentamos o procedimento heurístico que adotamos para encontrar uma solução inteira viável do PCBA. Relembrando o procedimento, resolvemos o modelo (46.1 – 46.4) relaxando a condição de integralidade da variável x_j^r. A seguir, truncamos (arredondamos para baixo) a solução obtida. Um problema residual é formulado a partir da demanda remanescente e resolvido, de modo que, este processo é repetido até que o arredondamento forneça uma solução residual nula. Ao final, é bem possível que uma pequena demanda ainda deva ser atendida, isso é feito por meio de uma heurística que gera padrões restritos (a mesma usada nos problemas residuais).

Em síntese, falta descrever o modelo do Problema Residual, bem como o modelo que usamos para gerar padrões restritos a demanda residual.

Seja $\bar{x}$ a solução contínua do problema (46.1 – 46.4). Armazene a solução arredondada $\lfloor \bar{x} \rfloor$ e os padrões associados a ela.

Considere $m_r \ll p_r$, o número de padrões gerados para cortar bobinas do tipo r, e sejam a seguinte demanda residual e estoque disponível:

$$\bar{d}_i = d_i - \sum_{r=1}^{m} \sum_{j=1}^{m_r} \frac{P^r}{L^r} \ell_i a_{ij}^r \lfloor \bar{x}_j^r \rfloor, \; i = 1, \ldots, n \qquad (58)$$

$$\bar{e}_r = e_r - \sum_{j=1}^{m_r} \lfloor \bar{x}_j^r \rfloor, \; r = 1, \ldots, m \qquad (59)$$

Seja agora o seguinte Problema Residual definido a partir $\overline{d}_i$ e $\overline{e}_r$, onde a variável de decisão é z_j^r e representa o número de vezes que o padrão j deve ser usado para cortar bobinas do tipo $r = 1, \ldots, m$.

Modelo do Problema Residual do PCBA

$$\text{minimizar} \quad \sum_{r=1}^{m}\sum_{j=1}^{p_r}\left(c_{aço}^r \, T_j^r + \sum_{h=1}^{H_{jr}} (c_{hj}^r y_{hj}^r) \right) z_j^r \qquad (60.1)$$

sujeito a:

$$\sum_{r=1}^{m}\sum_{j=1}^{p_r} \frac{P^r}{L^r} \ell_i \, a_{ij}^r \, z_j^r = \overline{d}_i, \; i = 1, \ldots, n \qquad (60.2)$$

$$\begin{aligned} \sum_{j=1}^{p_1} z_j^1 &\leq \overline{e}_1 \\ \ddots \quad &\vdots \\ \sum_{j=1}^{p_m} z_j^m &\leq \overline{e}_m \end{aligned} \qquad (60.3)$$

$$z_j^r \geq 0 \text{ , e inteiro } \; r = 1, \ldots, m, \; j = 1, \ldots, p_r \qquad (60.4)$$

Seja $\overline{z}$ a solução contínua do problema residual (60.1 – 60.4). Armazene a solução arredondada $\lfloor \overline{z} \rfloor$ e os padrões associados a ela. Atualize a demanda e o estoque disponível como foi feito em (58) e (59) e defina um novo problema residual. Repita este processo até que a solução residual seja nula.

Para gerar os padrões compartimentados restritos do Problema Residual (60.1 – 60.4) utilizamos o seguinte Problema da Mochila Compartimentada Restrito:

$$\text{maximizar} \sum_{h\in V_1}\left(\left(\sum_{i\in N_1} u_i a_{ih}\right)-c_h\right)y_h+\cdots+\sum_{h\in V_k}\left(\left(\sum_{i\in N_k} u_i a_{ih}\right)-c_h\right)y_h \qquad (61.1)$$

sujeito a:

$$\sum_{h\in V_1}(S_2+\sum_{i\in N_1}\ell_i a_{ih})y_h+\cdots+\sum_{h\in V_k}(S_2+\sum_{i\in N_k}\ell_i a_{ih})y_h \le L-S_1 \qquad (61.2)$$

$$L_{min}-S_2 \le \sum_{i\in N_s}\ell_i a_{ih} \le L_{max}-S_2,\ h\in V_s,\ s=1,\ldots,k \qquad (61.3)$$

$$\sum_{h\in V_1}\frac{P^r}{L^r}\ell_i a_{ih}y_h+\ldots+\sum_{h\in V_k}\frac{P^r}{L^r}\ell_i a_{ih}y_h \le \bar{d}_i,\ i\in N_s,\ s=1,\ldots,k \qquad (61.4)$$

$$a_{ih},\ y_h \ge 0 \text{ e inteiros},\quad i\in N=N_1\cup\cdots\cup N_k,\ h\in V=V_1\cup\cdots\cup V_k \qquad (61.5)$$

Lembrando que $u_i=(c^r_{aço}+\pi_i\frac{P^r}{L^r})\ell_i$ e $c_h=c^r_{hj}-c^r_{aço}S_2$, e que para cada tipo $r=1,\ldots,m$ de bobina uma mochila do tipo (61.1 – 61.5) deve ser resolvida. Na página 90 do capítulo anterior, foi apresentado o algoritmo COMPREST que pode encontrar uma solução viável para esta mochila.

Ao final, uma pequena demanda residual ainda pode existir após a resolução dos problemas residuais. Para cumprir esta demanda utilizamos a Heurística Gulosa de Repetição Exaustiva descrita na página 44 do capítulo 2, onde o COMPREST também foi usado para gerar estes últimos padrões compartimentados.

4.1 – Fracionamento das Bobinas do Estoque

Na página 44 do capítulo 2 apresentamos um modelo do PCBA que admite o fracionamento das bobinas do estoque. Como comentamos anteriormente, este modelo foi abandonado por gerar fitas com diâmetros não suportáveis pelo maquinário que confecciona os tubos.

Em visita a uma empresa do setor descobrimos que na realidade a máquina de tubos consegue trabalhar com fitas, cujo diâmetro tenha pelo menos a metade do diâmetro de uma bobina do estoque.

Assim, na resolução do PCBA admitimos que as bobinas do estoque possam ser fracionadas em duas outras bobinas, cujos "pesos" são exatamente a metade do "peso" da bobina original. Por exemplo, uma bobina do estoque que tenha 12000 Kg pode ser fracionada em duas de 6000 Kg.

Assim, o arredondamento das soluções contínuas $\bar{x}$ e $\bar{y}$ pode ser feito por meio da seguinte função $[\![.]\!]: \mathbb{R}_+ \to \mathbb{N} \cup \Omega$, onde $\Omega = \left\{ n + \frac{1}{2} \mid n \in \mathbb{N} \right\}$, definida por:

$$[\![x]\!] = \begin{cases} \lfloor x \rfloor + 0.5, \text{ se } x - \lfloor x \rfloor \geq 0.5 \\ \lfloor x \rfloor, \text{ se } x - \lfloor x \rfloor < 0.5 \end{cases} \qquad (62)$$

Na expressão (62) $\lfloor . \rfloor$ é a função maior inteiro (função piso). Observe que ao utilizar a função de arredondamento $[\![.]\!]$, surgirão em estoque bobinas fracionadas, de modo que, ao arredondar uma solução torna-se necessário identificar se o tipo de bobina em questão já sofreu fracionamento.

Caso a bobina em questão tenha sofrido o fracionamento ela não poderá ser novamente fracionada, e o arredondamento deverá ser feito como antes.

Uma estratégia que pode ser utilizada consiste em informar ao modelo do PCBA, desde o início, o fracionamento de todas as bobinas do estoque e proceder o arredondamento padrão por meio da função $\lfloor . \rfloor$ (maior inteiro). Por exemplo, ao invés de informarmos uma bobina de 12000 Kg, informamos duas de 6000 Kg, porém, sem efetuar fisicamente seu fracionamento.

O fracionamento só ocorreria de fato se o modelo indicasse o uso de padrões de corte distintos para cada bobina de 6000 Kg, entretanto, ele poderá indicar o uso de um mesmo padrão para as duas bobinas e o fracionamento não será fisicamente necessário.

Esta tática de arredondar as soluções contínuas $\bar{x}$ e $\bar{z}$ com a função $[\![.]\!]$, ou supor o fracionamento das bobinas, colabora para um cumprimento mais efetivo das demandas.

5 – Resultados Computacionais

A implementação do Método do Simplex com Geração de Colunas e dos algoritmos utilizados na construção dos padrões compartimentados resultou num aplicativo que denominamos **RollCut**. A descrição da interface deste aplicativo está descrita no apêndice D.

O aplicativo RollCut foi implementado em Delphi e executado num pentium II, 450 Mhz e com 160 Mb de RAM. Um exemplo do PCBA pode ser resolvido pelo RollCut por meio do COMPEX, que examina todas as possibilidades de padrões compartimentados em cada estágio da geração de colunas, ou por meio do COMPMT que é um procedimento heurístico para gerar os padrões. Os Problemas Residuais podem ser resolvidos por meio do algoritmo COMPREST, ou do COMPRESTMT (com limitantes de Martello-Toth), ambos são procedimentos heurísticos e foram descritos em detalhes no capítulo 4. Assim, para resolver um exemplo do PCBA podemos escolher uma das quatro combinações (são todas heurísticas) a seguir:

- **combinação 1**
COMPMT para gerar padrões compartimentados irrestritos;
COMPRESTMT para gerar padrões compartimentados restritos;

- **combinação 2**
COMPMT para gerar padrões compartimentados irrestritos;
COMPREST para gerar padrões compartimentados restritos;

- **combinação 3**
COMPEX para gerar padrões compartimentados irrestritos;
COMPRESTMT para gerar padrões compartimentados restritos;

- **combinação 4**
COMPEX para gerar padrões compartimentados irrestritos;
COMPREST para gerar padrões compartimentados restritos;

Na tabela 8 apresentamos os resultados numéricos que obtivemos na resolução de 21 exemplos do PCBA gerados aleatoriamente, cujos dados foram baseados nas condições práticas do problema descritas no capítulo 2.

As larguras foram geradas com valores distribuídos entre 55 mm e 250 mm e, as demandas dos itens foram geradas com valores variando de 87000 Kg a 99000 Kg.

O custo de bobinas intermediárias sujeitas à laminação foi considerado 50% maior que o de bobinas intermediárias isentas deste processo (este dado não foi obtido junto a uma empresa). Como estoque utilizamos três tipos de bobinas: com largura de 900 mm, de 1100 mm e de 1200 mm, todas elas com 12000 Kg. As capacidades dos compartimentos estão compreendidas entre os valores 154 mm e 456 mm.

Nas colunas **agrup**, **comp** e **itens** estão registrados respectivamente o número de agrupamentos, o número total de compartimentos e o número de itens de cada exemplo. Cada exemplo foi examinado segundo as quatro combinações possíveis, de forma que, foi medido o tempo de execução em minutos, segundos e milisegundos **(coluna tempo)**, a perda de material **(coluna perda)** e o número de bobinas intermediárias no processo **(coluna bobint)**.

Dados do Exemplo			**Combinação 1** COMPMT COMPRESTMT			**Combinação 2** COMPMT COMPREST			**Combinação 3** COMPEX COMPRESTMT			**Combinação 4** COMPEX COMPREST		
agrup	comp	itens	tempo	perda	bobint	tempo	perda	bobint	tempo	perda	bobint	tempo	perda	bobint
5	1515	32	00:22:410	---	773	01:39:410	---	768	01:57:210	---	581	05:47:890	---	579
		32	00:41:630	0,70%	748	03:11:800	0,70%	738	02:14:290	---	562	04:25:400	0,01%	545
		27	00:27:580	0,51%	711	01:08:660	0,51%	704	01:33:430	---	506	04:39:410	---	511
		32	00:57:290	0,58%	784	02:06:880	0,58%	788	01:32:000	---	696	03:25:870	---	686
		35	00:42:240	1,17%	807	02:07:420	1,19%	791	01:48:480	---	601	05:43:170	---	588
10	3030	62	01:40:350	0,04%	1472	10:28:180	---	1470	05:26:260	---	1268	16:34:650	0,02%	1275
		62	02:05:230	---	1318	10:02:420	0,01%	1330	03:50:080	---	1296	19:52:870	0,05%	1275
		52	01:32:270	0,08%	1233	10:21:700	---	1231	05:08:900	---	1198	10:38:830	---	1209
		60	02:30:440	0,38%	1492	12:41:260	0,39%	1481	04:21:340	---	1162	10:13:470	---	1173
		57	00:44:000	0,99%	1347	05:57:340	0,99%	1380	05:25:650	–	1111	16:15:750	0,03%	1084
15	4545	67	01:48:200	---	1577	06:22:330	---	1586	06:18:770	---	1341	14:10:960	0,01%	1347
		85	03:10:980	---	2014	14:43:470	---	2007	13:27:240	---	1556	29:43:870	---	1583
		82	02:37:800	0,67%	1948	15:24:400	0,67%	1957	07:49:880	---	1545	09:26:280	---	1538
		82	04:02:380	0,02%	2146	04:46:980	---	2126	09:31:280	---	1634	20:06:720	0,01%	1648
		75	02:24:340	2,77%	1676	17:29:410	0,77%	1663	08:53:770	---	1442	14:40:180	---	1431
20	6060	95	04:18:530	1,04%	2144	11:41:840	1,07%	2146	21:16:080	0,01%	1983	33:59:110	---	1989
		110	06:54:200	1,24%	2495	23:14:950	1,25%	2501	30:23:470	---	1987	38:21:380	---	1974
		120	07:23:250	4,12%	2675	41:31:640	4,18%	2674	26:38:990	---	2568	49:17:680	---	2581
		102	03:27:350	1,22%	2286	24:45:400	1,22%	2312	13:04:830	---	2175	29:05:480	---	2216
		100	08:04:990	0,01%	2482	14:24:960	---	2460	12:59:390	---	2029	30:34:180	---	2071
30	9090	142	17:15:180	---	3618	38:28:740	---	3602	32:03:770	0,01%	3200	61:07:010	---	3218

Tabela 8 – Resultados numéricos de exemplos do PCBA resolvidos pelo RollCut.

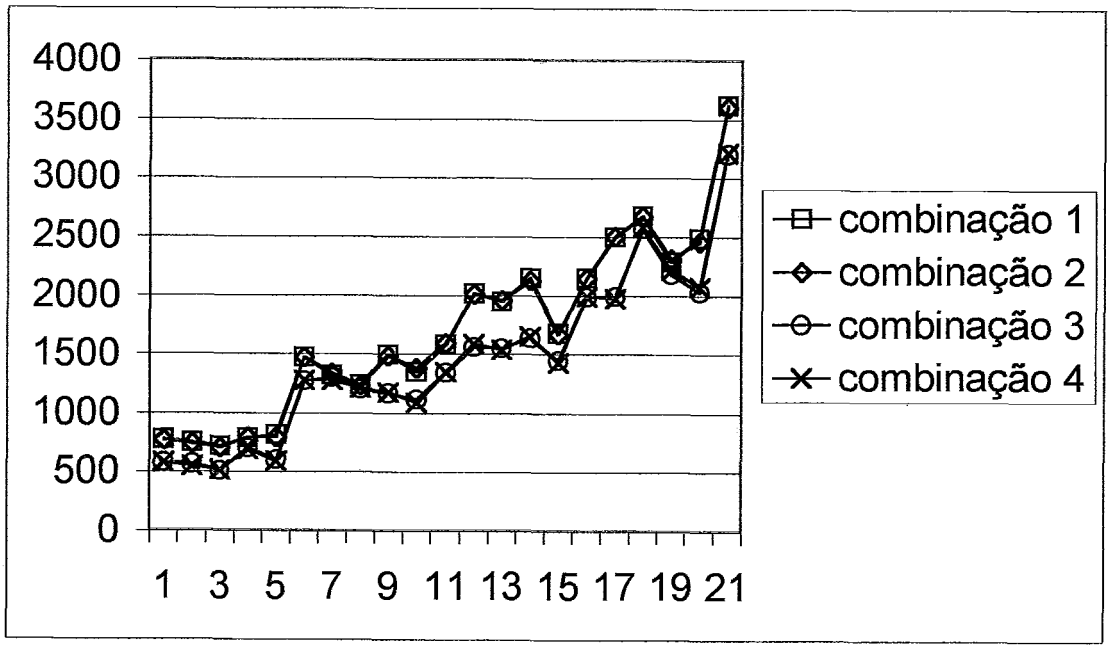

Figura 20 – Gráfico da evolução do número de bobinas intermediárias para cada combinação.

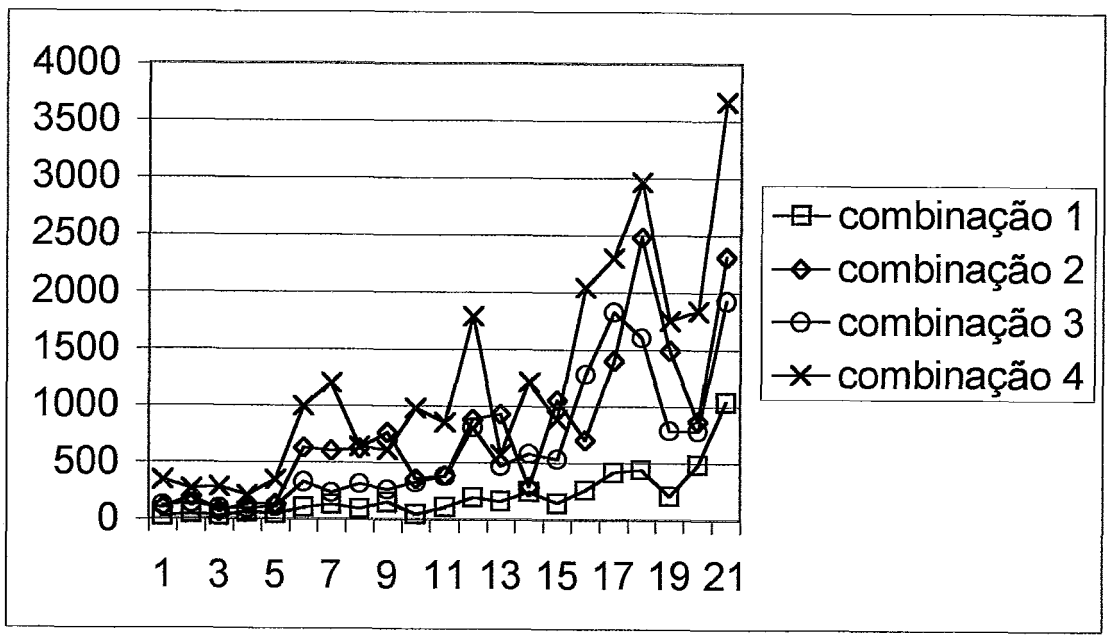

Figura 21 – Gráfico da evolução do tempo de execução para cada combinação.

No gráfico da figura 21 podemos observar que a combinação que requer menor tempo de execução na maioria dos exemplos é a combinação 1, porém, o número de bobinas intermediárias é bem superior se comparados com os obtidos pela combinação 4, que requer um tempo de execução mais elevado. Observe pelo gráfico da figura 20 que as combinações 1 e 2 são praticamente equivalentes quanto ao número de bobinas intermediárias, ocorrendo o mesmo com as combinações 3 e 4.

Os gráficos das figuras 20 e 21, assim como os resultados resumidos na tabela 8 apontam para a combinação 3 como a mais adequada para resolver exemplos do PCBA, pois, as perdas de material são baixas, o número de bobinas intermediárias é aceitável (comparativamente às demais combinações), e o tempo de execução é competitivo perante as outras combinações.

Para finalizar o capítulo, apresentamos os resultados obtidos pelo RollCut para um conjunto de 20 itens extraídos de dados reais, tabela 2 na página 45 do capítulo 2. Os agrupamentos dos itens são: $N_1=\{1, 2, 3\}$, $N_2=\{4\}$, $N_3=\{5\}$, $N_4=\{6, 7, 8\}$, $N_5=\{9, 10, 11, 12\}$, $N_6=\{13, 14, 15, 16, 17\}$ e $N_7=\{18, 19, 20\}$.

Fitas	Largura	Espessura Inicial	Espessura Final	Tipos de Aço	Demandas
tipo 1	200 mm	2,00 mm	1,10 mm	*SAE* 1008	50000 Kg
tipo 2	230 mm	2,00 mm	1,10 mm	*SAE* 1008 ou *SAE* 1010	55000 Kg
tipo 3	90 mm	2,00 mm	1,10 mm	*SAE* 1008	60000 Kg
tipo 4	115 mm	2,00 mm	1,20 mm	*SAE* 1008	57500 Kg
tipo 5	105 mm	2,00 mm	1,10 mm	*SAE* 1010	63000 Kg
tipo 6	105 mm	1,50 mm	0,90 mm	*SAE* 1010 ou *SAE* 1012	47000 Kg
tipo 7	200mm	1,50 mm	0,90 mm	*SAE* 1012	71000 Kg
tipo 8	120 mm	1,50 mm	0,90 mm	*SAE* 1010 ou *SAE* 1012	54000 Kg
tipo 9	90 mm	2,00 mm	2,00 mm	*SAE* 1008	53800 Kg
tipo 10	70 mm	2,00 mm	2,00 mm	*SAE* 1008	70200 Kg
tipo 11	65 mm	2,00 mm	2,00 mm	*SAE* 1008 ou *SAE* 1010	61000 Kg
tipo 12	105 mm	2,00 mm	2,00 mm	*SAE* 1008	50000 Kg
tipo 13	135 mm	1,00 mm	1,00 mm	*SAE* 1010 ou *SAE* 1012	55550 Kg
tipo 14	205 mm	1,00 mm	1,00 mm	*SAE* 1010	48600 Kg
tipo 15	125 mm	1,00 mm	1,00 mm	*SAE* 1010	49000 Kg
tipo 16	115 mm	1,00 mm	1,00 mm	*SAE* 1010	52500 Kg
tipo 17	180 mm	1,00 mm	1,00 mm	*SAE* 1010	63000 Kg
tipo 18	53 mm	1,50 mm	1,50 mm	*SAE* 1010 ou *SAE* 1012	60500 Kg
tipo 19	105 mm	1,50 mm	1,50 mm	*SAE* 1012	58200 Kg
tipo 20	90 mm	1,50 mm	1,50 mm	*SAE* 1010 ou *SAE* 1012	54500 Kg

Tabela 9 – Um exemplo do PCBA (extraído de dados reais).

Para este exemplo, também admitimos que o custo de bobinas intermediárias sujeitas à laminação é 50% maior que o de bobinas intermediárias isentas do processo e, utilizamos um estoque com 9 tipos de bobinas. Na tabela 10 estão resumidos os resultados obtidos.

Dados do Exemplo			**Combinação 1** COMPMT COMPRESTMT			**Combinação 2** COMPMT COMPREST			**Combinação 3** COMPEX COMPRESTMT			**Combinação 4** COMPEX COMPREST		
agrup	comp	itens	tempo	perda	bobint	tempo	perda	bobint	tempo	perda	bobint	tempo	perda	bobint
7	2121	20	0:11:320	0,77%	341	00:12:200	0,77%	342	00:33:230	0,25%	319	00:39:710	0,25%	325

Tabela 10 – Solução do exemplo da tabela 9.

Finalizamos assim a descrição dos estudos que efetuamos para resolver o Problema de Corte em Bobinas de Aço por meio da Técnica de Geração de Colunas, onde utilizamos o Problema da Mochila Compartimentada para gerar os padrões de corte. Pelos resultados obtidos o RollCut parece ter um bom desempenho, sobre tudo a combinação 3 que utiliza o COMPEX para gerar os padrões compartimentados iniciais, e o COMPRESTMT que gera os padrões compartimentados restritos.

No próximo capítulo faremos uma discussão sobre alguns pontos que ainda merecem atenção em estudos futuros.

Vivemos num hospício chamado Terra.

Capítulo 6
Conclusões e Investigações Futuras

Reunimos neste capítulo os principais tópicos que ainda necessitam atenção em estudos futuros. Inicialmente chamamos atenção para o fato de que não é única a definição dos agrupamentos das fitas (veja página 47 do capítulo 2), desta forma, seria interessante estudar como os possíveis conjuntos de agrupamentos podem influenciar nas soluções do PCBA.

Os métodos *branch-and-price* (página 16 do capítulo 1) tem sido muito explorados ultimamente, seria interestante desenvolver um algoritmo para o PCBA baseado nesta abordagem e comparar os resultados computacionais, sobre tudo com respeito ao tempo. Definir e testar funções objetivo alternativas para o PCBA também consiste uma tarefa importante e não muito evidente.

A estrutura de compartimentação encontrada na aplicação de corte de bobinas de aço sujeitas à laminação, como visto no capítulo 2, parece surgir em áreas ainda inexploradas. Como exemplo, considere o problema de programação de ligas metálicas numa fundição, onde num dia de trabalho N fornadas são preparadas. Cada fornada é associada uma única liga metálica que são fundidas para a produção de itens demandados.

O forno tem uma capacidade mínima (120 Kg) e uma capacidade máxima (360 Kg), indicando as restrições sobre os compartimentos. Uma vez definida uma liga para uma fornada (isto é, um agrupamento a um compartimento), o peso total em "Kg" dos itens a serem produzidos, restritos ao agrupamento definido pela liga, não pode violar as capacidades mínima e máxima do forno. Em termos da terminologia introduzida nesta tese, os itens associados à mesma liga pertencem ao mesmo agrupamento e cada fornada representa um compartimento, porém, ele difere do problema da mochila compartimentada definida no capítulo 4 desta tese, visto que, o número de compartimentos é inicialmente fixado (N fornadas num dia de trabalho).

Ressaltamos que compartimentos de capacidades menores que limite o máximo (fornadas que utilizam uma parte da capacidade do forno), não implicam no aumento do número de compartimentos (cada fornada tem um tempo fixo, ligeiramente menor quando a capacidade total do forno não esteja sendo utilizada, mas, não interfere no número de fornadas do dia de trabalho).

Tendo em vista sua importância prática, o Problema da Mochila Compartimentada Restrita (capítulo 4) necessita ser estudado com maior profundidade. O desenvolvimento de procedimentos mais eficientes para a resolução deste problema deve ser explorado.

Abordar o Problema da Mochila Compartimentada (capítulo 4) segundo a abordagem por grafos "e/ou" (MORABITO, 1992), (ARENALES, 1993) e (MORABITO e ARENALES, 1996) também pode se tornar bastante interestante e é um estudo a ser considerado. Além disto, uma possibilidade de estender o método de enumeração implícita (que segue a estratégia de busca em profundidade) proposto em (GILMORE e GOMORY, 1961, 1963) para o clássico Problema da Mochila seria uma tarefa bastante interestante.

Estender o Problema da Mochila Compartimentada para os casos bidimensional e tridimensional, onde podem surgir aplicações interestantes, por exemplo, quase sempre o acondicionamento de embalagens em furgões rodoviários não pode ser feito indiscriminadamente, pois, é preciso observar a distribuição das embalagens na carga. Restrições de agrupamento devem ser consideradas no sentido de evitar empacotamentos que necessitam trabalhar toda a carga para retirar parte dela.

Em problemas de corte bidimensionais, por exemplo cortes de placas de madeira numa indústria de móveis, certos itens devem passar por um processo técnico, cujo maquinário não suporta as dimensões originais das placas de madeira. Assim, há a necessidade de construir padrões compartimentos (figura 22), onde cada um deles corresponde a uma placa intermediária de dimensões menores que a da placa original.

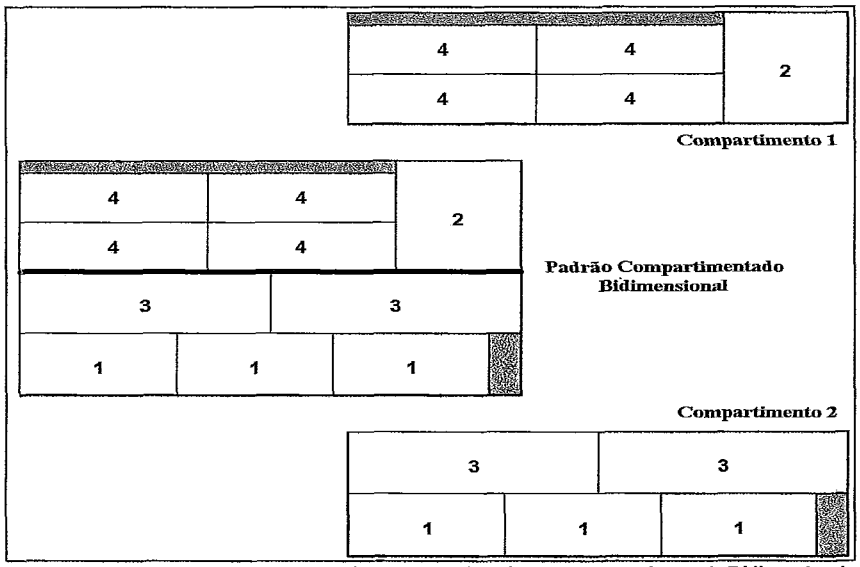

Figura 22 – Padrão de Corte Compartimentado Bidimensional.

Estas placas intermediárias (compartimentos) têm agora suas dimensões dentro das restrições requeridas para o processo técnico, por exemplo, algumas devem sofrer um processo especial de revestimento e outras não, ou numa outra situação, poderíamos supor que cada placa intermediária deverá ser revestida com materiais diferenciados.

Nesta tese abordamos o Problema de Corte em Bobinas de Aço, que é um problema de corte em duas fases. Apresentamos suas características técnicas e revisamos as abordagens encontradas na literatura, tratando deste específico problema. Em nossos estudos discutimos uma formulação matemática que utiliza a Técnica de Geração de Colunas (GILMORE e GOMORY, 1961, 1963).

Uma nova modalidade de mochila, que denominamos Mochila Compartimentada, aparece como o gerador de padrões (colunas) para o problema estudado. Em trabalhos anteriores (HAESSLER, 1979), (FERREIRA *et al.*, 1990), (PEREIRA, 1993), (CARVALHO, 1991) e (CARVALHO e RODRIGUES, 1994, 1995) este problema aparece de forma implícita, e MARQUES (2000) e MARQUES e ARENALES (1999, 2001) estudam o caso restrito. Em (HOTO, 1996) e (HOTO e ARENALES, 1996) são apresentados os primeiros elementos matemáticos que nos permitiram definir e escrever concisamente o Problema da Mochila Compartimentada (capítulo 4).

Desenvolvemos o algoritmo COMPEX (página 72 do capítulo 4) que resolve o caso irrestrito da compartimentação de uma mochila. Prosseguindo em nossos estudos propomos uma heurística que é baseada em limitantes superiores, obtidos pelos limitantes de Martello-Toth para mochilas irrestritas (MARTELLO e TOTH, 1990), disto resultou o algoritmo COMPMT (página 74 do capítulo 4) que demonstrou ser eficaz perante o algoritmo COMPEX (tabela 6 na página 77 do capítulo 4).

Ainda estudamos o caso restrito da Mochila Compartimentada (página 88 do capítulo 4), e desenvolvemos duas heurísticas que denominamos COMPREST e COMPRESTMT.

O Problema da Mochila Compartimentada 0-1 também foi apresentado e um caso particular deste problema foi estudado (página 79 do capítulo 4). Calculamos limitantes superiores que foram investigados, fornecendo critérios de poda na elaboração de uma árvore, onde cada ramo define uma solução viável do problema.

Por fim, a caracterização do Problema da Mochila Compartimentada apresentada nesta tese, e os algoritmos desenvolvidos, permitiram estender os trabalhos de CARVALHO (1991) e CARVALHO e RODRIGUES (1994, 1995) que tratam do Problema de Corte de Bobinas de Aço, além é claro, de contribuírem em aspectos de investigação teórica, visto que, outros pesquisadores têm apresentado interesse pelo problema.

Quem pensa muito, geralmente executa de menos,
quem executa de mais quase sempre erra muito.

Apêndice A

Tipologia dos Problemas de Corte e Empacotamento

1 - Introdução

Dada a enorme variedade de problemas de corte e empacotamento que surgem na prática, bem como o crescente interesse de pesquisadores pelo assunto, Harald Dyckhoff sugeriu em 1990 uma maneira de sistematizá-los. Posteriormente em 1992 ele e Ute Finke apresentaram um livro que reúne uma extensa bibliografia de diversos problemas da área, além de relacionar critérios associados à estrutura lógica e real dos problemas de corte e empacotamento, cujo objetivo não é o de simplesmente classificar os problemas, mas, estabelecer relações entre eles.

Na primeira parte desse apêndice faremos uma transcrição sucinta das idéias contidas no texto de DYCKHOFF e FINKE (1992), suficientes para determinar o tipo de um problema.

Um aspecto peculiar que ocorre em corte e empacotamento é o fato de que alguns problemas sem relação aparente com o contexto podem ser modelados como problemas dessa natureza, onde os objetos e itens são entidades abstratas. Por exemplo, a programação de veículos ou determinação de rotas, o balanceamento de uma linha de produção, a alocação de tarefas, etc. A figura 23 fornece uma idéia dessa peculiaridade.

Para determinarmos o tipo de um problema de corte e empacotamento aspectos básicos devem ser focados com respeito à estrutura lógica: dimensionalidade, tipo de seleção dos objetos/itens, características dos objetos/itens, restrições associadas aos padrões, objetivos, status da informação, variabilidade dos dados e métodos de solução. Finalmente são analisadas propriedades baseadas na realidade do problema.

O esquema que permite relacionar um problema com outro será baseado em quatro características básicas tidas como fundamentais: dimensionalidade, seleção de objetos/itens, características dos objetos/itens (sortimento de objetos e sortimento de itens). Cada problema de corte e empacotamento terá uma quádrupla associada a ele: **dimensionalidade/ seleção/ sortimento objetos/ sortimento itens**.

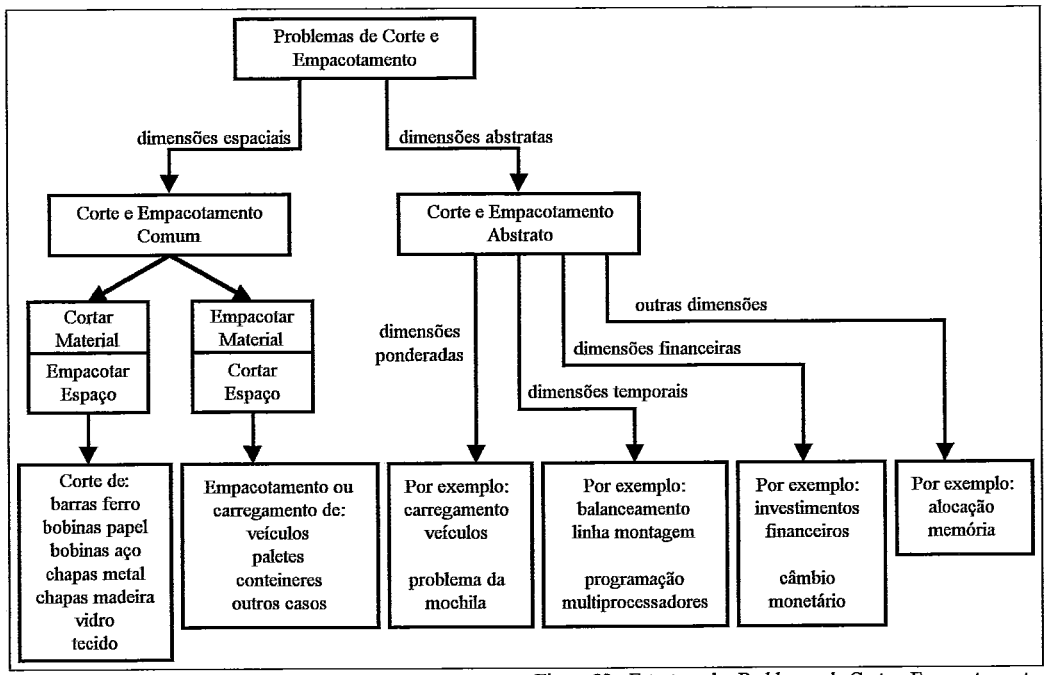

Figura 23 - Estrutura dos Problemas de Corte e Empacotamento.
Figura retirada do livro Cutting and Packing in Production and Distribution.

Vejamos agora cada um dos aspectos relacionados:

2 - Dimensionalidade

Esse aspecto está relacionado com o número de dimensões relevantes na definição de um padrão. Por exemplo, no caso do corte de barras de ferro (construção civil) apenas o comprimento da barra é relevante na definição de um padrão associado a ela, o problema é dito unidimensional.

No caso do corte de chapas de madeira (confecção de móveis) a largura e o comprimento da chapa são relevantes na definição de um padrão, o problema é dito bidimensional. O acondicionamento de caixas em um contêiner necessita da largura, comprimento e altura na definição de um padrão de empacotamento, o problema é dito tridimensional. Problemas em que são relevantes mais de três dimensões são ditos multidimensionais, é o caso do investimento no mercado financeiro em vários períodos.

Surge aqui uma questão introduzida por HAESSLER (1978). Seja o exemplo do problema de corte em bobinas de aço onde uma dimensão não é suficiente na definição do padrão, entretanto, duas dimensões vão além da necessidade, na figura 24 podemos perceber que para um padrão de corte estar definido a largura L da bobina e a variável

y_j (comprimento a ser cortado pelo padrão j) são necessárias, note porém que L está fixa, enquanto, y_j é variável. Esse problema não é unidimensional, nem tão pouco bidimensional, mas, duas dimensões são relevantes.

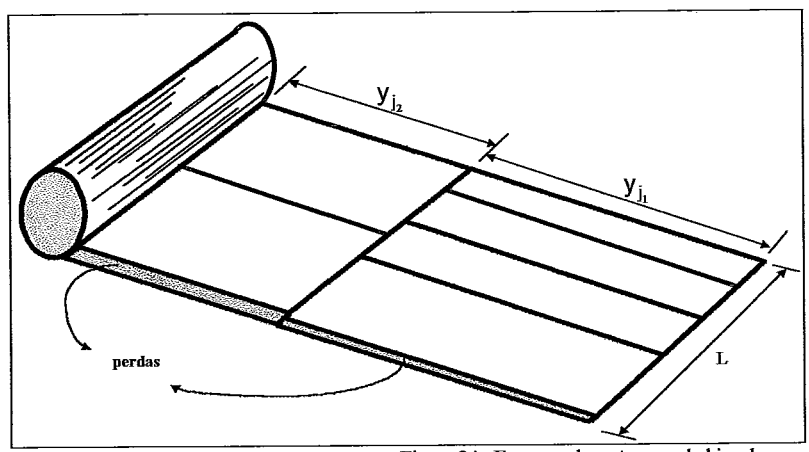

Figura 24 - Esquema de corte numa bobina de aço.

Problemas em que n+1 dimensões são relevantes e uma delas é variável serão denominados $n\frac{1}{2}$- dimensionais, no caso do corte em bobinas de aço trata-se de um problema $1\frac{1}{2}$- dimensional. Resumindo temos:

1) Problemas unidimensionais (uma dimensão relevante e fixa), simbolizados pelo algarismo **1** na quádrupla;
2) Problemas bidimensionais (duas dimensões relevantes e fixas), simbolizados pelo algarismo **2** na quádrupla;
3) Problemas tridimensionais (três dimensões relevantes e fixas), simbolizados pelo algarismo **3** na quádrupla;
4) Problemas n-dimensionais (n > 3 dimensões relevantes e fixas), simbolizados por **n** na quádrupla;
5) Problemas $n\frac{1}{2}$- dimensionais (n+1 dimensões relevantes, n fixas e uma variável), simbolizados por $\mathbf{n}\frac{1}{2}$ na quádrupla.

3 - Tipos de Seleção dos Objetos e Itens

São os objetos e os itens os responsáveis indiretos pela natureza combinatorial dos problemas de corte e empacotamento, visto que os itens são combinados entre si respeitando condições associadas aos objetos. Há quatro possibilidades para a seleção:

Tipo I - (não há seleção) o conjunto de itens é todo designado ao conjunto de objetos;

Tipo II - (seleção de itens) um subconjunto de itens é designado ao conjunto de objetos, esse tipo de seleção é simbolizada pela letra **B** na quádrupla;

Tipo III - (seleção de objetos) o conjunto de itens é todo designado a um subconjunto de objetos, esse tipo de seleção é simbolizada pela letra **V** na quádrupla;

Tipo IV - (seleção de itens e objetos) um subconjunto de itens é designado a um subconjunto de objetos.

4 - Características dos Objetos e Itens

4.1 - Tipos de Medida

Os problemas de corte e empacotamento apresentam variáveis associadas aos objetos e itens, aqueles cujas dimensões são todas fixas em geral apresentam variáveis discretas que medem o número de itens num padrão e a freqüência deste.

Já os problemas que apresentam uma dimensão variável possuem variáveis contínuas como no caso do corte de bobinas de aço.

4.2 - Aparência

Ela está associada ao formato, tamanho e orientação de um item em relação a um objeto. Dois objetos ou itens apresentam mesmo formato se após um redimensionamento eles são côngruos. Objetos ou itens, cujos três fatores coincidem são considerados de aparências idênticas. Caso apresentem mesmo formato e tamanhos diferentes (independente da orientação) são considerados de aparências distintas.

O formato dos objetos ou itens em problemas com mais de duas dimensões, são distinguidos em retangulares (retângulos ou blocos) e não-retangulares (por exemplo, círculos, triângulos, formas irregulares, etc).

O tamanho dos itens relativamente aos objetos é um fator que pode introduzir dificuldades consideráveis nos problemas. A orientação de itens relativamente a objetos depende de cada problema, de modo que ela pode ser fixa ou não.

4.3 - Sortimento

Esse atributo está associado ao tipo e variabilidade das aparências dos objetos e itens. Duas categorias são destacadas: homogênea, onde os objetos ou itens possuem aparências congruentes (formato e tamanho iguais) com orientações diferentes; e heterogênea, onde os objetos e itens possuem formatos e possivelmente tamanhos diferentes.

Para o sortimento dos objetos utilizamos a seguinte simbologia na quádrupla: a letra **O** indica apenas um objeto, a letra **I** indica objetos de aparências idênticas e a letra **D** objetos de aparências distintas. Quanto ao sortimento dos itens: a letra **F** indica poucos itens, a letra **M** indica muitos itens de muitas aparências distintas, a letra **R** indica muitos itens de aparências distintas em relativa quantidade, e finalmente a letra **C** indica itens congruentes.

4.4 - Disponibilidade

Está associada à quantidade (que pode ser limitada ou não), ordem (por exemplo, no carregamento de contêineres uma ordem sob os itens deve ser considerada para o posterior descarregamento em destinos distintos) e datas de utilização (esse fator está ligado à expedição) dos objetos e itens.

5 - Restrições sob os Padrões

Essencialmente elas estão associadas com características geométricas, operacionais e processos de alocação.

As características geométricas estão ligadas às distâncias entre itens, limitação quanto à combinação entre as aparências dos itens, limitação quanto ao número de aparências dos itens e limitação quanto ao número de itens.

As características operacionais estão ligadas ao número de estágios (número de ciclos de operações independentes no corte ou empacotamento), sequenciamento e freqüência (limitação no número de padrões distintos).

O processo de alocação pode ser estático, onde os objetos e os itens estão disponíveis no mesmo período de tempo. Nesse caso a alocação pode ser feita *on-line* (nem todos os objetos e itens são conhecidos com antecedência), ou *off-line* (todos os objetos e itens são conhecidos antecipadamente). Um outro processo de alocação é o dinâmico, onde os objetos e itens não estão disponíveis no mesmo período de tempo, neste caso os itens podem ou não ser realocados.

6 - Objetivos

Define a meta que deve ser atingida, por exemplo, minimizar o número de objetos a serem utilizados, minimizar a perda absoluta ou relativa de material, minimizar custos de produção, minimizar custos de material, minimizar custos devidos às trocas de padrões, minimizar custos de armazenamento, maximizar lucros, etc. A combinação de vários objetivos é apreciável.

7 - Status e Variabilidade dos Dados

Os dados em geral são determinísticos, porém, em alguns problemas eles podem ser estocásticos. Além disso, eles podem ser exatos ou admitir tolerâncias.

8 - Métodos de Solução

Os métodos de solução são baseados em heurísticas, por exemplo, a FFD (*First-Fit Decreasing*) e a BFD (*Best-Fit Decreasing*). Outros procedimentos são adotados, tais como: heurísticas de otimização local, programação matemática, heurísticas de decomposição e de particionamento do problema, heurísticas que restringem o espaço das soluções, etc.

Na tabela 11 apresentamos um resumo da nomenclatura sugerida por Dyckhoff e Finke que permite relacionar um problema de corte e empacotamento com outro, por meio da quádrupla: **dimensionalidade/ seleção/ sortimento objetos/ sortimento itens**.

Dimensionalidade	**Sortimento Objetos**
(1) unidimensional; **(2)** bidimensional; **(3)** tridimensional; **(n)** n-dimensional; $(\mathbf{n}\frac{1}{2})\,n\frac{1}{2}$ - dimensional.	**(O)** um objeto; **(I)** objetos de aparências idênticas; **(D)** objetos de aparências distintas.
Seleção	**Sortimento Itens**
(B) um subconjunto de itens é designado ao conjunto de objetos; **(V)** o conjunto de itens é todo designado a um subconjunto de objetos.	**(F)** poucos itens; **(M)** muitos itens de muitas aparências distintas; **(R)** muitos itens de aparências distintas em relativa quantidade; **(C)** itens congruentes.

Tabela 11 - Nomenclatura da Tipologia de Dyckhoff-Finke.

As propriedades baseadas na realidade são:

1) Identificação dos objetos/itens e setor industrial (por exemplo, na descrição do problema a ser abordado nesse trabalho identificamos as bobinas de aço como objetos e as fitas como itens, de modo que a empresa pertence ao setor do aço);
2) Contexto do planejamento (por exemplo, se será tratado apenas o contexto corte e empacotamento, ou outros atributos como: planejamento de estoque, programação de períodos, planejamento de transporte, equilíbrio e estabilidade de cargas, etc);
3) Funcionalidade do Software (se ele será desenvolvido para fins comerciais, ou para simulações e testes).

Na tabela 12 são listados alguns problemas clássicos da literatura com as respectivas tipologias.

Problema	**Tipo**
Problema da mochila clássico	1/ B/ O
Carregamento de palete	2/ B/ O/ C
Problema da mochila multidimensional	/ B/ O/ C
Problema do *Bin-Packing* dual	1/ B/ O/ M
Carregamento de veículos	1/ V/ I/ F ou 1/ V/ I/ M
Carregamento de contêiner	3/ V/ I/ ou 3/ B/ O/
Problema do *Bin-Packing* clássico	1/ V/ I/ M
Problema do *Cutting-Stock* clássico	1/ V/ I/ R
Problema do *Bin-Packing* bidimensional	2/ V/ D/ M
Problema do *Cutting-Stock* bidimensional	2/ V/ I/ R
Problema geral do *Cutting-Stock*	1/ / / ou 2/ / / ou 3/ / /
Problema do balanceamento de uma linha de montagem	1/ V/ I/ M
Problema de alocação de tarefas	1/ V/ I/ M
Problema de alocação de memória	1/ V/ I/ M
Problema da troca de moeda	1/ B/ O/ R
Problema do investimento financeiro em vários períodos	N/ B/ O/

Tabela 12 - Alguns problemas encontrados na literatura com seu respectivo tipo.
Tabela retirada do artigo: A typology of cutting and packing problems. European Journal of Operational Research, 44, 145-159.

Apêndice B

Método do Simplex

Decomposição de Dantzig-Wolfe

Geração de Colunas de Gilmore-Gomory

O Método do Primal – Simplex

1 – Introdução

Muitos problemas de ordem prática podem ser traduzidos matematicamente sob a forma de uma função linear e um sistema de equações lineares. Neste sentido, seria de vital importância obter um método capaz de computar soluções para problemas desta natureza.

O Método Simplex, introduzido em 1947 por DANTZIG (1963), tem por finalidade resolver um Problema de Programação Linear (PPL) que sempre pode ser escrito da seguinte forma:

(PPL)

$$\text{maximizar} \quad z = \sum_{j=1}^{n} c_j x_j$$

sujeito a:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \,,\; i = 1, \ldots, m$$

$$x_j \geq 0 \,,\; j = 1, \ldots, n$$

Ou ainda na forma matricial:

(PPL)

$$\text{maximizar} \quad z = cx$$

sujeito a:

$$Ax = b$$

$$x \geq 0$$

Onde $c = (c_j)_{1xn}$, $x = (x_j)_{nx1}$, $b = (b_i)_{mx1}$ e $A = (a_{ij})_{mxn} \in \mathbb{R}^{mxn}$ com posto m, ou seja, A possui m colunas linearmente independentes, ou ainda, A possui uma submatriz inversível de ordem m.

A idéia do Simplex não é muito difícil de se compreender, para isso, desenvolveremos alguns cálculos simples que necessitam uma familiaridade com notações matriciais. Considere B a submatriz inversível de ordem m obtida de A, e seja N a matriz formada pelas colunas restantes de A. Observe que as colunas de A que formam B podem não ser as m primeiras, mas sem perda de generalidade, iremos supor A = (B N).

De maneira análoga particionamos os vetores $c = (c_B \; c_N)$ e $x^T = (x_B^T \; x_N^T)$, onde c_B e x_B possuem m componentes associadas à matriz B. Tendo em vista estas notações, nosso último PPL pode ser rescrito como:

(PPL)

$$\text{maximizar} \quad z = c_B x_B + c_N x_N$$

sujeito a:

$$Bx_B + Nx_N = b$$

$$x_B, x_N \geq 0$$

Podemos ainda escrever x_B em função de x_N, isto é, $x_B = B^{-1}b - B^{-1}Nx_N$.

O vetor x_B é denominado vetor de varáveis básicas, enquanto x_N é denominado vetor de variáveis não básicas. Assim, definimos I_B como conjunto dos índices básicos, e I_N como o conjunto dos índices não básicos. Observe que $I_B \cap I_N = \varnothing$ e $I_B \cup I_N = \{1, \ldots, n\}$. Quando $x_N = 0$ denominamos $\bar{x} = (\bar{x}_B^T \; 0)$ de solução básica primal-viável, e dizemos que a partição efetuada é primal-viável, onde $\bar{x}_B^T = B^{-1}b$.

Novamente podemos reescrever o PPL, agora em função das varáveis não básicas:

(PPL)

maximizar $\quad z = c_B B^{-1}b - (c_B B^{-1}N - c_N)x_N$

sujeito a:

$$Bx_B + Nx_N = b$$

$$x_B, x_N \geq 0$$

Considere ainda as seguintes notações:

$\pi = c_B B^{-1}$, denominado vetor dos multiplicadores Simplex

$z_j = \pi a_j$ e $y_j = B^{-1}a_j$, $j \in I_B \cup I_N$, onde a_j é uma coluna de A

$$\bar{z} = c_B B^{-1}b = \pi b = c_B \bar{x}_B$$

Mais uma vez reescreveremos nosso PPL:

(PPL)

maximizar $\quad z = \bar{z} + \sum_{j \in I_N} (c_j - z_j)$

sujeito a:

$$x_B = \bar{x}_B - \sum_{j \in I_N} y_j x_j$$

$$x_B, x_j \geq 0, \; j \in I_N$$

No último formato que apresentamos para o PPL original, é importante observar que para cada $j \in I_N$ se $\bar{x}_B \geq 0$ e $c_j - z_j \leq 0$, o valor do objetivo z não poderá ser melhorado. Observe ainda que $(\bar{x}_B^T \; 0)$ é uma solução viável do PPL, portanto, será uma solução ótima.

Por outro lado, durante o transcorrer do Algoritmo Simplex iremos nos deparar com soluções viáveis $\hat{x}$ não básicas (existirá ao menos um $j \in I_N$, tal que $\hat{x}_j > 0$). Neste caso, como poderíamos proceder para obtermos uma solução básica sem diminuir o valor do objetivo?

Responder a esta indagação corresponde a descrever o Algoritmo Simplex, cujo objetivo é passar de $\hat{x}$ até uma solução ótima. Note que as restrições $x_B = \bar{x}_B - \sum_{j \in I_N} y_j x_j$ podem ser escritas como $x_{B(i)} = \bar{x}_{B(i)} - \sum_{j \in I_N} y_{ij} x_j$, $i = 1, \ldots, m$.

Vamos destacar uma variável x_k, $k \in I_N$, e fazer seu valor variar, enquanto as demais variáveis não básicas permanecerão com seus valores inalterados, isto é, $x_j = \hat{x}_j$ para $j \in I_N - \{k\}$. Assim, teremos $x_{B(i)} = \bar{x}_{B(i)} - \sum_{j \in I_N - \{k\}} y_{ij} \hat{x}_j - y_{ik} x_k$, e como $x_{B(i)} \geq 0$, segue que $y_{ik} x_k \leq \bar{x}_{B(i)} - \sum_{j \in I_N - \{k\}} y_{ij} \hat{x}_j$ para $i = 1, \ldots, m$. Agora considere os seguintes conjuntos de índices:

1. $L_0 = \{i \mid y_{ik} = 0\}$, neste caso basta tomar x_k não negativo;

2. $L_1 = \{i \mid y_{ik} > 0\}$, neste caso $x_k \leq \dfrac{1}{y_{ik}} \left(\bar{x}_{B(i)} - \sum_{j \in I_N - \{k\}} y_{ij} \hat{x}_j \right)$;

3. $L_2 = \{i \mid y_{ik} < 0\}$, neste caso $x_k \geq \dfrac{1}{y_{ik}} \left(\bar{x}_{B(i)} - \sum_{j \in I_N - \{k\}} y_{ij} \hat{x}_j \right)$;

Sejam agora os seguintes parâmetros:

$$\alpha_k = \frac{1}{y_{sk}}\left(\bar{x}_{B(s)} - \sum_{j\in I_N-\{k\}} y_{sj}\hat{x}_j\right) = \min_{i\in L_1}\left\{\frac{1}{y_{ik}}\left(\bar{x}_{B(i)} - \sum_{j\in I_N-\{k\}} y_{ij}\hat{x}_j\right)\right\}$$

$$\beta_k = \frac{1}{y_{rk}}\left(\bar{x}_{B(r)} - \sum_{j\in I_N-\{k\}} y_{rj}\hat{x}_j\right) = \max_{i\in L_2}\left\{\frac{1}{y_{ik}}\left(\bar{x}_{B(i)} - \sum_{j\in I_N-\{k\}} y_{ij}\hat{x}_j\right)\right\}$$

$$\gamma_k = \max\{0, \beta_k\}$$

Note que $\gamma_k \le x_k \le \alpha_k$, e que quando $L_1 = \varnothing$ temos $\alpha_k \to +\infty$, ao passo que quando $L_2 = \varnothing$ temos $\beta_k \to -\infty$.

Observe que as colunas $a_{B(1)}, \ldots, a_{B(m)}$ de B são linearmente independentes, existem λ_i reais, tais que $a_k = \sum_{i=1}^{m} \lambda_i a_{B(i)}$. Seja $s \in M = \{1, \ldots, m\}$, tal que $\lambda_s \neq 0$, de forma que, podemos escrever $a_{B(s)} = \frac{1}{\lambda_s}\left(a_k - \sum_{i\in M-\{s\}} \lambda_i a_{B(i)}\right)$ e concluir que a matriz básica B pode ser substituída por $B' = \left(a_{B(1)} \ldots a_{B(s-1)}\ a_k\ a_{B(s+1)} \ldots a_{B(m)}\right)$ inversível. Ainda vale destacar que se $v^T = (\lambda_1, \ldots, \lambda_m)$, teremos $a_k = Bv$, ou seja, $v = B^{-1}a_k$.

Tendo em vista estas observações, concluímos que se $y_{sk} \neq 0$, então podemos substituir a corrente matriz B pela matriz B'.

Já estamos aptos a escrever um procedimento do Simplex:

Algoritmo Primal-Simplex para Maximização

Fase 1

Encontre uma solução básica primal-viável, $x_B = B^{-1}b$, para o PPL, onde A = (B, N).

Fase 2
Faça PARE = falso.
Enquanto PARE = falso faça:

1. Calcule o vetor linha $\pi = c_B B^{-1} = (\pi_1, \ldots, \pi_n, \pi_{n+1}, \ldots, \pi_{n+r}, \ldots, \pi_{n+m})$ dos **multiplicadores Simplex**.

3. Determine uma **coluna** de índice k de N **para entrar na base** (na matriz básica B).

   Isso pode ser feito por uma das duas maneiras:

   1.a) Escolha aquela de **maior custo relativo**:
   $c_k - \pi a_k =$ máximo $\{ c_j - \pi a_j \mid a_j$ é uma coluna arbitrária de N$\}$

   2.a) Escolha a_k pela Regra de Bland (BLAND, 1977).

   Teste a Otimalidade (maximização):
   Se $c_k - \pi a_k \leq 0$ então PARE = verdade. *{a corrente solução é ótima}*
   Senão vá para o passo 4.

4. Determine a **Direção Simplex** $y_k = B^{-1} a_k$.
   Se $\alpha_k \to +\infty$ $(y_k < 0)$ então **PARE = verdade**. *{o problema não tem solução ótima finita}*
   Senão vá para o passo 5.

5. Determine a **coluna** de índice ℓ de B **para sair da base** (a_ℓ será substituída por a_k).
   Isso é feito da seguinte forma: $\dfrac{x_{B(\ell)}}{y_\ell} =$ mínimo $\left\{ \dfrac{x_{B(i)}}{y_i} \mid y_i > 0, i = 1, \ldots, n \right\}$.
   Faça $x_k = \alpha_k$ e $x_{B(\ell)} = 0$.

6. **Atualizações:** da matriz básica B (coloque a_k no lugar de a_ℓ), e de B^{-1}.

A Regra de Bland (BLAND, 1977), mencionada no passo 3, evita um fenômeno conhecido como ciclagem. Para explicarmos esta regra, considere o (PPL) no seu formato padrão:

$$\begin{aligned} &\text{maximizar} \quad z = cx \\ &\text{sujeito a:} \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

No modelo anterior $c=(c_j)_{1xn}$, $x=(x_j)_{nx1}$, $b=(b_i)_{mx1}$ e $A=(a_{ij})_{mxn} \in \mathbb{R}^{mxn}$ têm posto m. O número total de soluções básicas do (PPL) tem como cota superior $\frac{n!}{m!(n-m)!}$. Observe que o Método do Simplex parte de uma solução básica primal viável e passa para outra, iterativamente, até que a condição de otimalidade seja satisfeita, ou concluindo que a solução é ilimitada.

De uma iteração para outra do Simplex, a variável x_ℓ poderá entrar na base com valor **nulo**, neste caso a solução básica é denominada **degenerada** e a função objetivo não terá seu valor alterado.

Quando o Método do Simplex é aplicado a um problema em que não existem soluções básicas degeneradas, a função objetivo sempre será otimizada de uma iteração para outra, logo, o Simplex convergirá.

Por outro lado, quando um problema apresenta soluções básicas degeneradas poderemos observar, às vezes, que após algumas iterações tenhamos uma solução básica já visitada anteriormente, sem que o objetivo seja alterado. Este fenômeno é conhecido como ciclagem e, pode ser evitado por uma regra enunciada por BLAND (1977):

Regra de Bland (maximização)

Critério de entrada a_k entra na base se $c_k - \pi a_k > 0$ e k é o menor índice entre todos os $j \in I_N$, tais que $c_j - \pi a_j > 0$;

Critério de saída a_ℓ sai da base se $\frac{x_{B(\ell)}}{y_{\ell k}} = \theta =$ mínimo $\left\{ \frac{x_{B(i)}}{y_{ik}} \mid y_{ik} > 0, i = 1, \ldots, n \right\}$, onde $B(\ell)$ é o menor índice para os quais $\frac{x_{B(i)}}{y_{ik}} = \theta$;

Ao aplicar a Regra de Bland no Procedimento do Simplex, nunca haverá o fenômeno de ciclagem. Para maiores detalhes recomendamos CHVÀTAL (1980, pag 37) e MACULAN (1998, pag 44).

2 – Como obter uma Solução Básica Viável

Quando a solução básica viável inicial não for evidente, podemos acrescentar uma variável artificial $g_i \geq 0$, $i = 1, \ldots, m$, em cada restrição do (PPL), de modo que, teremos o seguinte Problema Artificial:

(Problema Artificial)

maximizar $z_{art} = \sum_{i=1}^{m} g_i$

sujeito a:

$$(\sum_{j=1}^{n} a_{ij} x_j) + g_i = b_i, \; i = 1, \ldots, m$$

$$x_j \geq 0, \; j = 1, \ldots, n$$

$$g_i \geq 0, \; i = 1, \ldots, m$$

Observe que $x_j = 0$, $j = 1, \ldots, n$, e $g_i = b_i \geq 0$, $i = 1, \ldots, m$, fornece uma solução básica viável para o Problema Artificial (PA) que pode ser resolvido pelo Simplex, lembre que min z = - max (- z).

Caso $z_{art} > 0$ o conjunto de restrições do (PPL) é vazio. Se por outro lado $z_{art} = 0$, e a base final do (PA) não contiver nenhuma coluna associada às variáveis artificiais g_i, $i = 1, \ldots, m$, ela será uma base primal-viável do (PPL). Mesmo que a solução básica do (PA) seja degenerada, pode-se iniciar o Simplex com esta base para resolver o (PPL), porém, não deve-se permitir que as variáveis g_i associadas à base tenham valores distintos de zero.

3 – Cálculo da Inversa de B

Observe que em cada etapa do Simplex é preciso determinar $x_B = B^{-1}b$, $\pi = c_B B^{-1}$ e $y_k = B^{-1}a_k$, onde a_k é a coluna escolhida para entrar na base. Na realidade temos três sistemas de equações lineares para resolver que determinam x_B, π e y_k.

Caso tivéssemos de resolver estes sistemas apenas uma vez, poderíamos utilizar os métodos numéricos de resolução de sistemas de equações lineares simultâneas sem a inversão explícita de B, entretanto, em cada iteração do Simplex necessitamos resolver estes três sistemas.

Nesta seção, descreveremos um método de atualização da inversa de B, isto é, em cada iteração do Simplex iremos calcular a inversa de B a partir da inversa da iteração precedente. Seja $B_k = (a_1 \dots a_{\ell-1}\ a_k\ a_{\ell+1} \dots a_m)$ a corrente matriz básica obtida da matriz básica $B_\ell = (a_1 \dots a_{\ell-1}\ a_\ell\ a_{\ell+1} \dots a_m)$, da iteração anterior do Simplex, substituindo a coluna a_ℓ pela coluna a_k. Note que a matriz B_ℓ^{-1} é conhecida e estamos interestados em obter B_k^{-1}, assim considere:

$$B_\ell^{-1}B_k = (B_\ell^{-1}a_1 \dots B_\ell^{-1}a_{\ell-1}\ \ B_\ell^{-1}a_k\ \ B_\ell^{-1}a_{\ell+1} \dots B_\ell^{-1}a_m)$$

Observe que $e_j = B_\ell^{-1}a_j = (0 \dots 1 \dots 0)$, $j = 1, \dots, m$, $j \neq k$, onde o valor unitário ocupa a j-ésima posição. Seja $v = B_\ell^{-1}a_k = (v_1 \dots v_{\ell-1}\ \ v_\ell\ \ v_{\ell+1} \dots v_m)$, então:

$$E_\ell = B_\ell^{-1}B_k = (e_1 \dots e_{\ell-1}\ \ v\ \ e_{\ell+1} \dots e_m) \Leftrightarrow B_k = B_\ell E_\ell \Leftrightarrow B_k^{-1} = E_\ell^{-1}B_\ell^{-1}, \text{ onde}$$

$$E_\ell = \begin{bmatrix} 1 & \dots & 0 & v_1 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & v_{\ell-1} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & v_\ell & 0 & \cdots & 0 \\ 0 & \dots & 0 & v_{\ell+1} & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & v_m & 0 & \cdots & 1 \end{bmatrix}, \text{ cujo determinante é } v_\ell$$

Basta que $v_\ell \neq 0$ para que exista E_ℓ^{-1} e, por conseqüência, B_k^{-1}. Observe ainda que:

$$E_\ell^{-1} = \begin{bmatrix} 1 & \dots & 0 & -v_1/v_\ell & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & -v_{\ell-1}/v_\ell & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1/v_\ell & 0 & \cdots & 0 \\ 0 & \dots & 0 & -v_{\ell+1}/v_\ell & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & -v_m/v_\ell & 0 & \cdots & 1 \end{bmatrix}$$

Assim, a inversa da corrente matriz básica é obtida por meio de atualizações, e quando esta atualização fornecer uma matriz inversa com grande acúmulo de erros, recomendamos utilizar um método numérico que calcule a matriz inversa.

A Decomposição de Dantzig-Wolfe

1 – Introdução

A decomposição DANTZIG e WOLFE (1959) consiste em escrever um problema de programação linear por meio de um problema principal e de problemas secundários, valendo-se da teoria poliedral. Iterativamente estes problemas secundários avaliam a melhor proposta a ser feita ao problema principal por meio de geração de colunas, que são vértices e raios extremos convenientemente definidos. Considere o seguinte teorema:

Teorema da Representação de um Conjunto Poliédrico Convexo

Seja o conjunto poliédrico convexo $X = \{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\}$, onde $A \in \mathbb{R}^{m \times n}$ tem posto igual a m. Um ponto $x \in X$ se, e somente se existem λ_j, $j = 1, \ldots, p$, tais que $\sum_{j=1}^{p} \lambda_j = 1$ e $\mu_i \geq 0$, $i = 1, \ldots, q$, de modo que: $x = \sum_{j=1}^{p} \lambda_j v^j + \sum_{i=1}^{q} \mu_i r^i$, onde v^j é um vértice extremo de X e r^i é um raio extremo de X.

A seguir, considere o seguinte Problema de Programação Linear:

$$\begin{aligned} &\text{minimizar} \quad z = cx \\ &\text{sujeito a:} \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

Particionaremos A e b da seguinte maneira, $A=\begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ e $b=\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$, onde $A_1 \in \mathbb{R}^{m_1 x n}$, $A_2 \in \mathbb{R}^{m_2 x n}$, $b_1 \in \mathbb{R}^{m_1}$ e $b_2 \in \mathbb{R}^{m_2}$. Assim, teremos:

$$\text{minimizar} \quad z = cx$$

sujeito a:

$$A_1 x = b_1$$

$$A_2 x = b_2$$

$$x \geq 0$$

Considere o conjunto $X = \{x \in \mathbb{R}^n \mid A_2 x = b_2, x \geq 0\} \neq \varnothing$, e sejam $V(X) = \{v^1, \ldots, v^p\}$ o conjunto de seus vértices extremos e $R(X) = \{r^1, \ldots, r^q\}$ o conjunto de seus raios extremos. De acordo com o Teorema da Representação de um Conjunto Poliédrico Convexo, passamos a ter o seguinte problema:

Problema Mestre

$$\text{minimizar} \quad z = \sum_{j=1}^{p} (cv^j)\lambda_j + \sum_{i=1}^{q} (cr^i)\mu_i$$

sujeito a:

$$\sum_{j=1}^{p} (A_1 v^j)\lambda_j + \sum_{i=1}^{q} (A_1 r^i)\mu_i = b_1$$

$$\sum_{j=1}^{p} \lambda_j = 1$$

$$\lambda_j, \mu_i \geq 0, \; j = 1, \ldots, p, \; i = 1, \ldots, q$$

A matriz dos coeficientes das variáveis λ_j e μ_i pode ser esquematizada como:

$$M = \begin{pmatrix} A_1 v^1 & \cdots & A_1 v^p & A_1 r^1 & \cdots & A_1 r^q \\ 1 & \cdots & 1 & 0 & \cdots & 0 \end{pmatrix}$$

Seja uma matriz B inversível formada por $m_1 + 1$ colunas de M, e associada a uma solução básica primal-viável. Esta matriz terá colunas do tipo $\begin{pmatrix} A_1 v^j \\ 1 \end{pmatrix}$ ou do tipo $\begin{pmatrix} A_1 r^i \\ 0 \end{pmatrix}$, de modo que escreveremos B como:

$$B = \begin{pmatrix} A_1 v^1 & \cdots & A_1 v^{p_1} & A_1 r^1 & \cdots & A_1 r^{q_1} \\ 1 & \cdots & 1 & 0 & \cdots & 0 \end{pmatrix}, \text{ onde } p_1 + q_1 = m_1 + 1$$

A seguir, verificaremos se a base B está associada a uma solução dual-viável do Problema Mestre.

2 – Os Problemas Secundários da Decomposição de Dantzig-Wolfe

Suponha que $\pi = (\pi_1 \ \pi_0)$, onde π_0 é um número real, e observe que se $cv^j - \pi \begin{pmatrix} A_1 v^j \\ 1 \end{pmatrix} = (c - \pi_1 A_1) v^j - \pi_0 \geq 0$ e $cr^i - \pi \begin{pmatrix} A_1 r^i \\ 1 \end{pmatrix} = (c - \pi_1 A_1) r^i \geq 0$, para todo j e i pertencentes ao conjunto dos índices da partição não básica de M, então a matriz B definirá uma solução básica ótima do Problema Mestre.

Ressaltamos que o número de vértices e raios extremos de X pode ser extremamente grande (milhões ou até bilhões), impossibilitando o cálculo de todos os custos relativos. Lembrando que estamos minimizando, poderíamos pensar em calcular o mínimo dos custos relativos por meio do seguinte problema de programação linear:

$$\begin{aligned} &\text{minimizar} \quad z_1 = (c - \pi_1 A_1) v - \pi_0 \\ &\text{sujeito a:} \\ &A_2 v = b_2 \\ &v \geq 0 \end{aligned}$$

Ou equivalentemente,

Problema Secundário dos Vértices Extremos

$$z_1 = \max_{v} (\pi_1 A_1 - c)v + \pi_0$$
sujeito a:
$$A_2 v = b_2$$
$$v \geq 0$$

Do mesmo modo, é preciso avaliar os custos relativos associados aos raios extremos, o que pode ser feito resolvendo o seguinte problema:

Problema Secundário dos Raios Extremos

$$z_2 = \max_{r} (\pi_1 A_1 - c)r$$
sujeito a:
$$A_2 r = b_2$$
$$r \geq 0$$

Se ambos os máximos z_1 e z_2 forem não negativos, então a matriz B definirá uma solução básica ótima do Problema Mestre.

Paralelamente a DANTZIG e WOLFE (1959), surgem os trabalhos os trabalhos de GILMORE e GOMORY (1961, 1963) em corte e empacotamento que tratam da Técnica de Geração de Colunas, a qual descrevemos a seguir.

A Técnica de Geração de Colunas de Gilmore-Gomory

1 – Introdução

GILMORE e GOMORY (1961, 1963) foram pioneiros ao descreverem a Técnica de Geração de Colunas que faz uso da particular estrutura de um problema de corte, onde as colunas da matriz de restrição obedecem a uma lei de formação bem definida.

Vamos considerar o Problema de Corte de Estoque descrito na página 13 do capítulo 1, onde poucos tipos de itens devem ser produzidos, porém, a quantidade a ser produzida é alta. O objetivo é selecionar objetos em estoque para a produção dos itens, de modo a minimizar o custo total destes objetos.

Dados do problema:

n: número total de itens;

ℓ_i: largura do item i, $i = 1, \ldots, n$;

d_i: demanda do item i, $i = 1, \ldots, n$;

L: largura do objeto;

Variáveis do problema:

x_j: número de vezes que o padrão j é usado;

a_{ij}: número de itens do tipo i no padrão j;

O problema pode ser modelado em duas etapas:

1.a etapa Encontre todas as p colunas que definem os padrões de corte, segundo as restrições:

$$\sum_{i=1}^{n} \ell_i a_{ij} \leq L, \; j = 1, \ldots, p$$
$$a_{ij} \geq 0, \text{inteiro}, \; i = 1, \ldots, n, \; j = 1, \ldots, p$$

2.a etapa Resolva o seguinte Problema de Otimização Linear Inteiro, onde $a_j = (a_{1j}, \ldots, a_{nj})^T$, $j = 1, \ldots, p$:

$$\text{minimizar} \quad \sum_{j=1}^{p} c_j x_j$$
sujeito a:
$$a_1 x_1 + \cdots + a_p x_p = d$$
$$x_j \geq 0, \text{inteiro}, \; j = 1, \ldots, p$$

O parâmetro c_j da função objetivo, pode representar a perda de material produzida pelo padrão de corte j, mas, ele também pode ser considerado unitário, e neste caso o objetivo passa a ser o de minimizar o número de objetos necessários para o cumprimento das demandas dos itens.

Seja a matriz $A = (a_1 \ldots a_p)$ e uma partição básica A = (B, N). Considere, o vetor $\pi = c_B B^{-1} = (\pi_1, \ldots, \pi_n)$. Se $c_j - \pi a_j \geq 0$ para cada coluna a_j de N, então a matriz B definirá uma solução básica ótima do Problema de Programação Linear Relaxado da segunda etapa. Vale destacar que c_j depende da coluna a_j, digamos que $c_j = L - \sum_{i=1}^{n} \ell_i a_{ij}$ representa a perda de material produzida pelo padrão de corte j, assim $c_j - \pi a_j = L + \sum_{i=1}^{n} (\ell_i - \pi_i) a_{ij}$.

ressaltamos que o valor de p pode atingir a ordem de vários milhões e até bilhões de padrões (colunas), isso inviabiliza o uso do Método do Simplex descrito neste apêndice, pois, o número de colunas da matriz N a serem examinadas é muito alto. Para resolver o problema GILMORE e GOMORY (1961, 1963) consideram a relaxação contínua do Problema Linear Inteiro da segunda etapa. A seguir, observam que cada coluna do problema relaxado, assim como do problema linear inteiro, obedecem a uma lei de formação bem definida matematicamente. Os autores propõem resolver o seguinte problema para testar a otimalidade do Simplex:

Gerador de Colunas

$$\text{maximizar} \quad z = L - \sum_{i=1}^{n} (\ell_i - \pi_i) a_{ij}$$

sujeito a:

$$\sum_{i=1}^{n} \ell_i a_{ij} \leq L$$

$$a_{ij} \geq 0, \text{ inteiro, } i = 1, \ldots, n$$

Observe que o Gerador de Colunas para o Problema de Corte de Estoque é um Problema da Mochila Irrestrita. A seguir apresentamos o Método do Simplex com a Técnica de Geração de Colunas de Gilmore-Gomory.

Algoritmo Primal-Simplex com Geração de Colunas para Minimização

Fase 1
Considere a partição básica primal-viável de A = (B, N), onde:

$$B = \begin{bmatrix} \left\lfloor \frac{L}{\ell_1} \right\rfloor & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \left\lfloor \frac{L}{\ell_n} \right\rfloor \end{bmatrix}, \text{ onde } \lfloor . \rfloor \text{ é a função maior inteiro (piso)}$$

Fase 2
Faça PARE = falso.
Enquanto PARE = falso faça:

1. Calcule o vetor linha $\pi = c_B B^{-1} = (\pi_1, \ldots, \pi_n, \pi_{n+1}, \ldots, \pi_{n+r}, \ldots, \pi_{n+m})$ dos multiplicadores Simplex.

2. Calcule a solução básica $x_B = B^{-1}d$.

3. Gere uma coluna de índice k de N para entrar na base (na matriz básica B).
   Isto é feito resolvendo o seguinte problema da mochila:

   $$\text{maximizar} \quad z = \sum_{i=1}^{n} (\ell_i - \pi_i) a_{ik}$$

   sujeito a:

   $$\sum_{i=1}^{n} \ell_i a_{ik} \leq L$$

   $$a_{ik} \geq 0, \text{ inteiro}, i = 1, \ldots, n$$

   Teste a Otimalidade (minimização):
   Se $z \geq 0$ então PARE = verdade. *{a corrente solução é ótima}*
   Senão vá para o passo 4.

4. Determine a Direção Simplex $y_k = B^{-1}a_k$.
   Se $y_k < 0$ então PARE = verdade. *{o problema não tem solução ótima finita}*
   Senão vá para o passo 5.

5. Determine a coluna de índice ℓ de B para sair da base (a_ℓ será substituída por a_k).
   Isso é feito da seguinte forma: $\frac{x_{B(\ell)}}{y_\ell} = \text{mínimo} \left\{ \frac{x_{B(i)}}{y_i} \mid y_i > 0, i = 1, \ldots, n \right\}$.

6. Atualize a matriz básica B (coloque a_k no lugar de a_ℓ), e atualize B^{-1}.

Observe que a largura L foi excluída da função objetivo do Gerador de Colunas do passo 3, pois, trata-se de uma constante. O algoritmo Simplex com Geração de Colunas que descrevemos aqui pode apresentar variações, pois, ele depende fundamentalmente do Problema Gerador de Colunas que varia segundo a definição dos padrões de corte.

A Técnica de Geração de Colunas pode ser utilizada em problemas mais elaborados, como Problemas de Corte Bidimensionais (veja página 23 do capítulo 1) e na construção de árvores que envolvem a resolução de um problema de geração de colunas em cada nó (veja página 16 do capítulo 1).

Apêndice C
Classes Algorítmicas

1 – Introdução

Nesta parte do apêndice faremos um resumo sobre complexidade que pode ser mais bem estudada em (GAREY e JOHNSON, 1979) ou (CAMPELLO e MACULAN, 1994).

Problemas de natureza algorítmica que não admitem solução por algoritmo são denominados **não computáveis**. Problemas para os quais existem algoritmos polinomiais são denominados **tratáveis**, enquanto, aqueles que comprovadamente não podem ser resolvidos por algoritmos polinomiais são denominados **intratáveis**.

Os problemas comprovadamente tratáveis, isto é, com algoritmos polinomiais conhecidos, constituem uma classe que é comumente denominada de **P**, já os que podem ser resolvidos por algoritmos enumerativos, cuja busca no espaço de soluções é feita em árvore com profundidade limitada por funções polinomiais no tamanho da entrada do problema, e com largura eventualmente exponencial, compõem uma classe normalmente designada por **NP**. Observe que é intuitivo $P \subseteq NP$.

Para melhor entender estas duas classes é conveniente classificar os problemas algorítmicos nas seguintes categorias:

- Problemas de Decisão;
- Problemas de Localização;
- Problemas de Otimização;

2 – Problemas de Decisão

Um problema de decisão tem por objetivo responder **sim (1)** ou **não (0)** à uma determinada pergunta.

Por exemplo, dado um grafo G e um inteiro $k > 0$, existe ou não uma clique em G de tamanho maior ou igual a k? Pois bem, caso exista um subgrafo completo maximal em G com k ou mais vértices a resposta é **sim**, do contrário é **não**.

Uma classe de problemas ainda mais geral será considerada, ela é formada pelos problemas de pesquisa. A formulação de um problema de pesquisa Π consiste de um conjunto finito D_Π de exemplos, de modo que, para cada exemplo I de D_Π existe um conjunto finito $S_\Pi(I)$ de soluções do exemplo I. Um algoritmo resolve um problema de pesquisa Π se, para qualquer exemplo $I \in D_\Pi$ o algoritmo retorna **não** como resposta quando $S_\Pi(I)$ é vazio, do contrário retorna uma solução de $S_\Pi(I)$.

Note que um problema de decisão pode ser formulado como um problema de pesquisa, basta definir $S_\Pi(I) = \{\text{sim}\}$ se o exemplo I apresenta resposta **sim**, e $S_\Pi(I) = \varnothing$ no caso contrário.

3 – Problemas de Localização

Em problemas de localização o objetivo consiste em encontrar, quando existe, uma determinada estrutura satisfazendo requisitos previamente especificados por uma questão.

Por exemplo, dado um grafo G e um inteiro $k > 0$, encontre, caso exista, uma clique em G de tamanho maior ou igual a k. Perceba que no caso anterior o problema consiste em decidir se existe ou não a clique, aqui o problema consiste em determiná-la.

4 – Problemas de Otimização

Um problema de otimização tem por objetivo obter determinada estrutura concomitante a algum critério de otimização previamente estabelecido. Por exemplo, nos casos anteriores, além de decidir e determinar uma clique de tamanho maior ou igual a k, ela deve ter tamanho mínimo.

Neste ponto é razoável compreender que o problema de decisão apresenta dificuldade não maior que o de localização, e este, não maior que o de otimização. Em geral utiliza-se o problema de decisão para obter alguma indicação a respeito da intratabilidade.

5 – Algoritmos Não Determinísticos

A *princípio* determinísticos são os algoritmos em que toda a operação é definida de forma única, conceitualmente esta imposição pode ser relaxada, permitindo que um determinado algoritmo implemente operações cujo resultado não é único, porém, limitado a um certo conjunto finito de possibilidades.

Para definir um algoritmo A não determinístico três funções primitivas são utilizadas:

Escolha (A): escolhe um dos elementos de A;

Fracasso: sinaliza término com fracasso;

Sucesso: sinaliza término com sucesso;

Para um algoritmo não determinístico, sempre que houver uma seqüência de escolhas capaz de conduzir ao término com sucesso, isso será efetuado no menor número possível de iterações. Se por outro lado, não existe tal seqüência, o algoritmo sinaliza o término com fracasso. Além do mais, um algoritmo não determinístico é de ordem O(f(n)) se para qualquer exemplo de tamanho $n \geq n_0$, resultando em término com sucesso, o tempo necessário é no máximo c.f(n) para alguma escolha de constantes c e n_0. Do contrário, isto é, para exemplos levando ao fracasso, o algoritmo é de ordem O(1).

Problemas de decisão que admitem algoritmo não determinístico polinomial constituem a classe **NP**, já aqueles para os quais existem algoritmos determinísticos polinomiais formam a classe **P**.

Uma grande questão da área de complexidade é determinar se $P = NP$ ou $P \neq NP$, um importante resultado é devido a COOK (1983), laureado em 1982 com o Prêmio Turing. Em 1971, Cook formulou e respondeu a seguinte questão:

Questão de Cook

"Existirá algum problema NP (classe não polinomial) tal que,
sendo provada sua pertinência a P (classe polinomial),
implicaria $P = NP$, ou mais resumidamente $NP \subseteq P$?"

Para responder a esta questão considere o seguinte problema:

Problema da SAT - Satisfiability Problem

Consiste em determinar se uma expressão *booleana* na sua forma normal conjuntiva é verdadeira para alguma atribuição das varáveis lógicas que a definem, neste caso, **verdadeiro** ou **falso**. Uma expressão lógica $E(x_1, \ldots, x_n)$ está na sua forma normal conjuntiva quando é formada por uma conjunção de cláusulas $C_1, \ldots, C_k$, e cada uma delas é formada por uma disjunção de literais associados às variáveis *booleanas* $x_1, \ldots, x_n$.

A resposta de Cook para a questão por ele mesmo levantada é a seguinte:

Resposta de Cook

SAT pertencerá à classe **P** se, e somente se **P = NP**

6 – Problemas NP-árduos e NP-completos

Um problema de pesquisa Π_0 é reduzido a um problema Π se, para qualquer exemplo I_0 de Π_0, um exemplo I de Π pode ser construído por meio de uma função de redução $f(I_0) = I$ computável em tempo polinomial. O problema Π_0 pode ser encarado como um caso particular de Π, que será tão difícil quanto for Π_0. Assim, existindo um algoritmo polinomial que resolva Π, o mesmo resolverá Π_0 em tempo polinomial.

Um problema de pesquisa Π é denominado **NP-árduo** se Π_0 é reduzido a Π, qualquer que seja Π_0 em **NP**.

A resposta encontrada por Cook estabelece que todos os problemas de **NP** podem ser reduzidos polinomialmente ao problema da **SAT**, e se este possuir um algoritmo eficiente, todo problema em **NP** será polinomial. Temos aqui um problema de lógica que seria o mais difícil na classe **NP**, e neste sentido Cook o denominou **NP-completo**.

Um problema de pesquisa Π em NP, tal que Π é NP-árduo denomina-se **NP-completo**.

Das definições decorre que os problemas NP-completos são os mais difíceis na classe NP. A classe NP contém diversos problemas combinatórios notoriamente difíceis, para os quais não foram encontrados algoritmos polinomiais, assim é pouco provável que **P** = **NP**, porém, esta questão permanece em aberto.

Em 1972 Karp mostrou que 24 problemas importantes da matemática discreta são reduzidos à **SAT**, sendo assim tão difíceis quanto este e portanto NP-completos. Pelos seus trabalhos na Teoria de Complexidade (KARP, 1975, 1986), Karp também foi laureado com o Prêmio Turing em 1985.

Apêndice D

Interface do Aplicativo RollCut

Na seção 1 deste apêndice iremos descrever a interface do aplicativo RollCut que foi desenvolvido durante esta tese, cujo objetivo é resolver exemplos do Problema de Corte de Bobinas de Aço. Na seção 2 descreveremos os fluxos de dados entre os procedimentos que compõem o aplicativo.

1 – A Interface do Aplicativo

O aplicativo RollCut foi implementado em Delphi 4, recomendamos a seguinte configuração mínima para um bom desempenho do aplicativo: Windows 98 ou superior, processador de 200 Mhz ou superior, 32 Mb de RAM ou mais e 2 Mb de espaço disponível em disco.

O RollCut é constituído de um único formulário ilustrado na figura 25, onde estão descritos seus botões de comando. O aplicativo possui um editor de texto integrado onde podem ser visualizados os arquivos de estoque e de clientes (figura 26), e um campo de *Status* que fornece informações em tempo de execução.

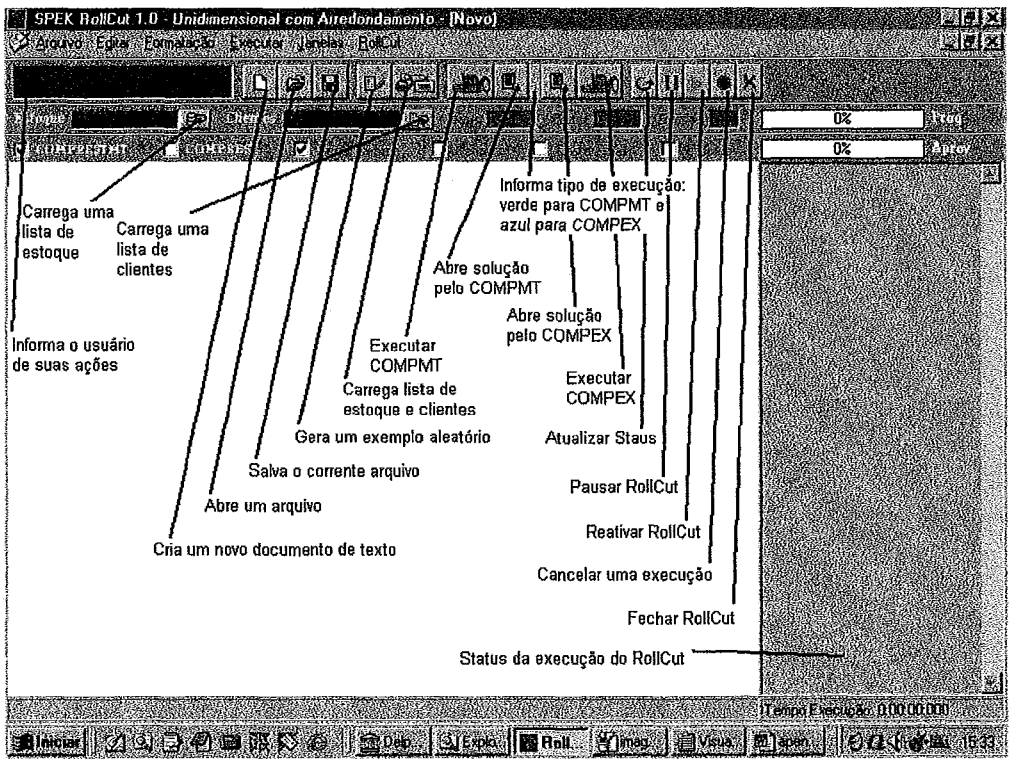

Figura 25 – Tela de entrada do RollCut.

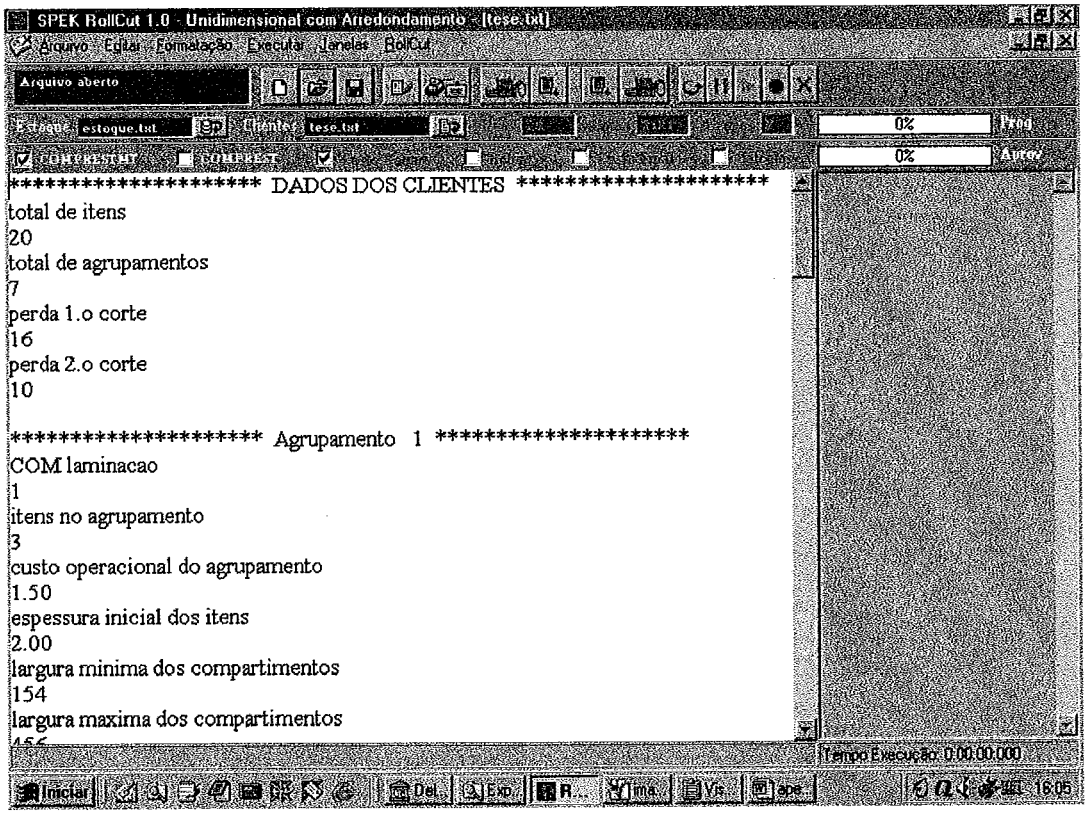

Figura 26 – Editor de texto do RollCut.

Nos campos **Estoque** e **Clientes**, o aplicativo RollCut informa o nome dos arquivos de estoque e clientes carregados, caso a lista de estoque escolhida não seja suficiente para atender a lista de clientes, o aplicativo pede que uma nova lista seja carregada, não permitindo a execução enquanto esta tarefa não for completada (figura 27). O editor de texto integrado ao RollCut também permite visualizar os relatórios com o resultado das soluções. Estes relatórios podem ser obtidos pelo simples acionamento dos respectivos botões (figura 28).

Uma vez carregada uma lista de clientes, no campo **Itens** é informado o número total de itens a serem processados, no campo **Comp** o número total de compartimentos a serem examinados, e no campo **Grupos** o número de agrupamentos. Este campo também é usado para informar o número de agrupamentos de um exemplo aleatório a ser gerado.

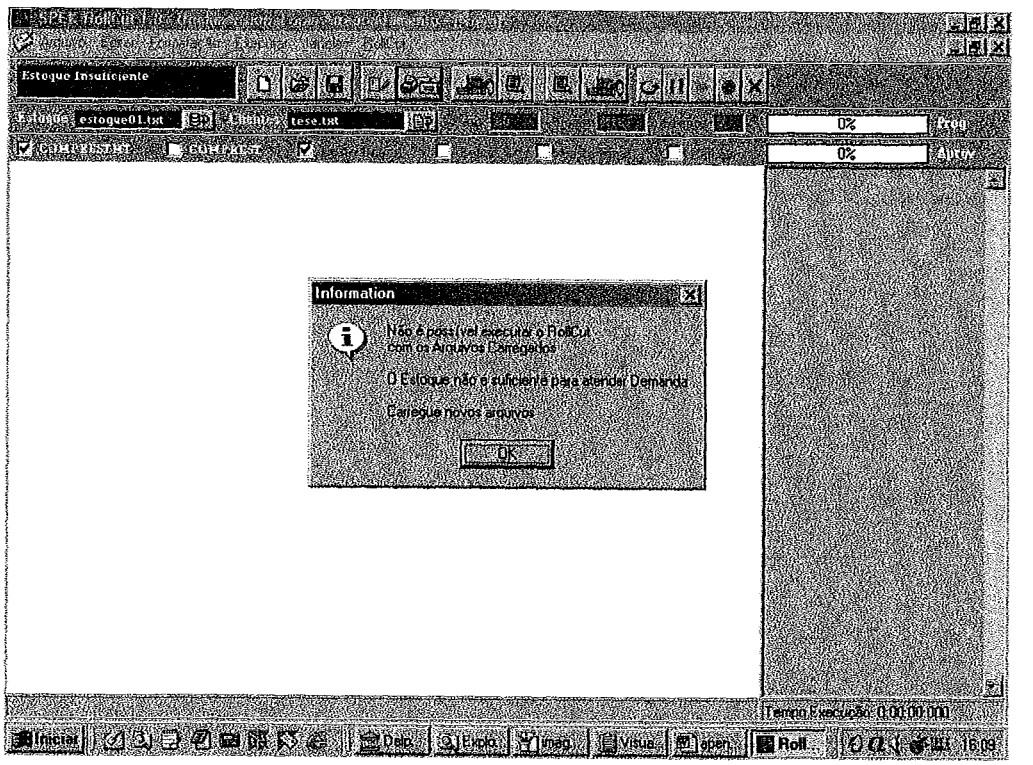

Figura 27 – Arquivos carregados são *incompatíveis*.

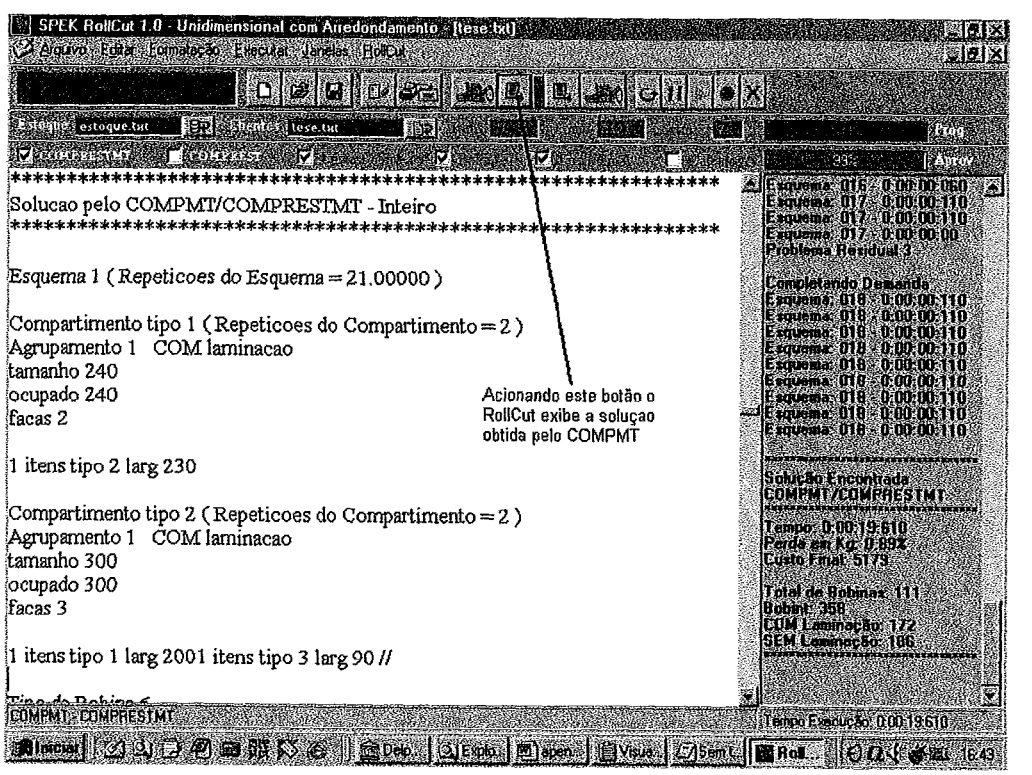

Figura 28 – Arquivo de solução pelo COMPMT.

Para gerar padrões compartimentados restritos existem duas caixas de opções: **COMPRESTMT** e **COMPREST**. A primeira informa ao aplicativo que o algoritmo COMPRESTMT deve ser usado, a segunda informa ao aplicativo que o algoritmo COMPREST deve ser usado. As duas caixas não devem ser marcadas ao mesmo tempo, mas, se forem marcadas simultaneamente, o aplicativo pede para que apenas uma seja escolhida e impede a execução enquanto esta tarefa não for completada (figura 29).

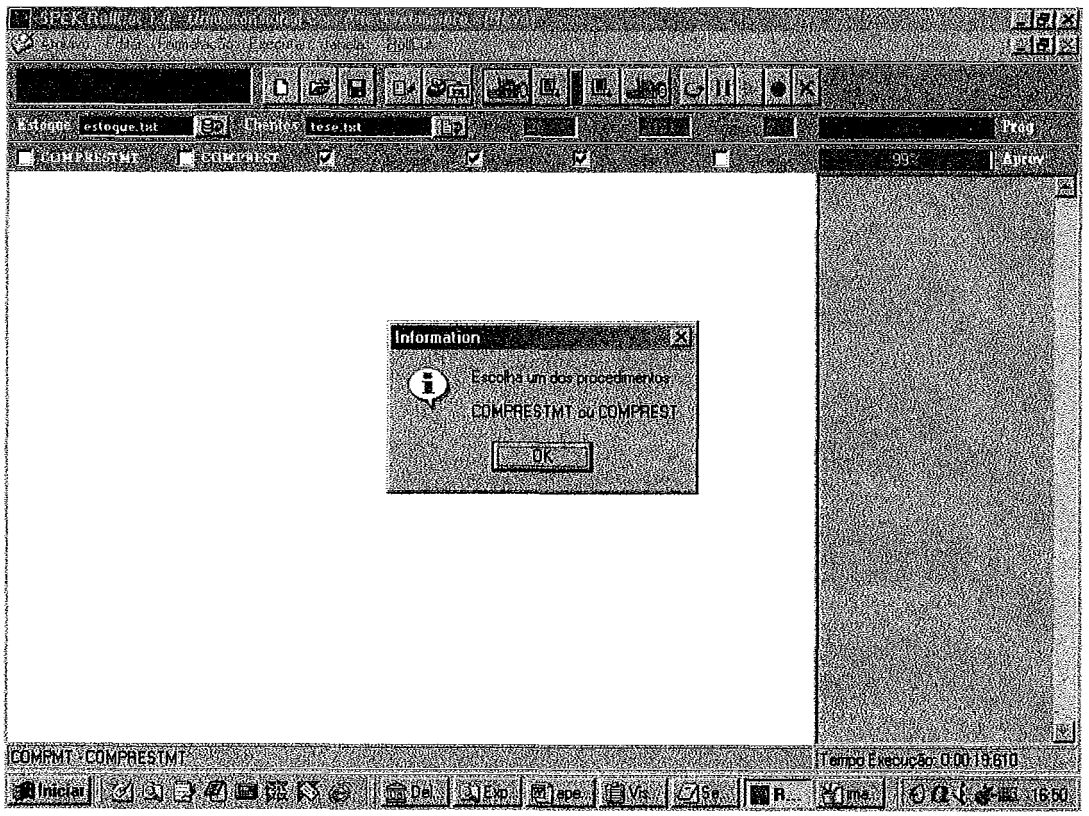

Figura 29 – Escolha simultânea dos procedimentos COMPRESTMT e COMPREST.

Existem ainda mais quatro opções: **Custo Bobint** (habilita o aplicativo a considerar custos pela utilização de uma bobina intermediária), **Bobint** (habilita o aplicativo a informar o número de bobinas intermediárias envolvidas na solução), **Custo Final** (habilita o aplicativo a calcular o custo associado a perdas de material e a bobinas intermediárias) e **Cálculos** (habilita o aplicativo a registrar em arquivos predeterminados as matrizes básicas e inversas, os multiplicadores Simplex e as colunas geradas). Na figura 30 ilustramos cada uma das opções do menu principal do RollCut, com seus respectivos comandos.

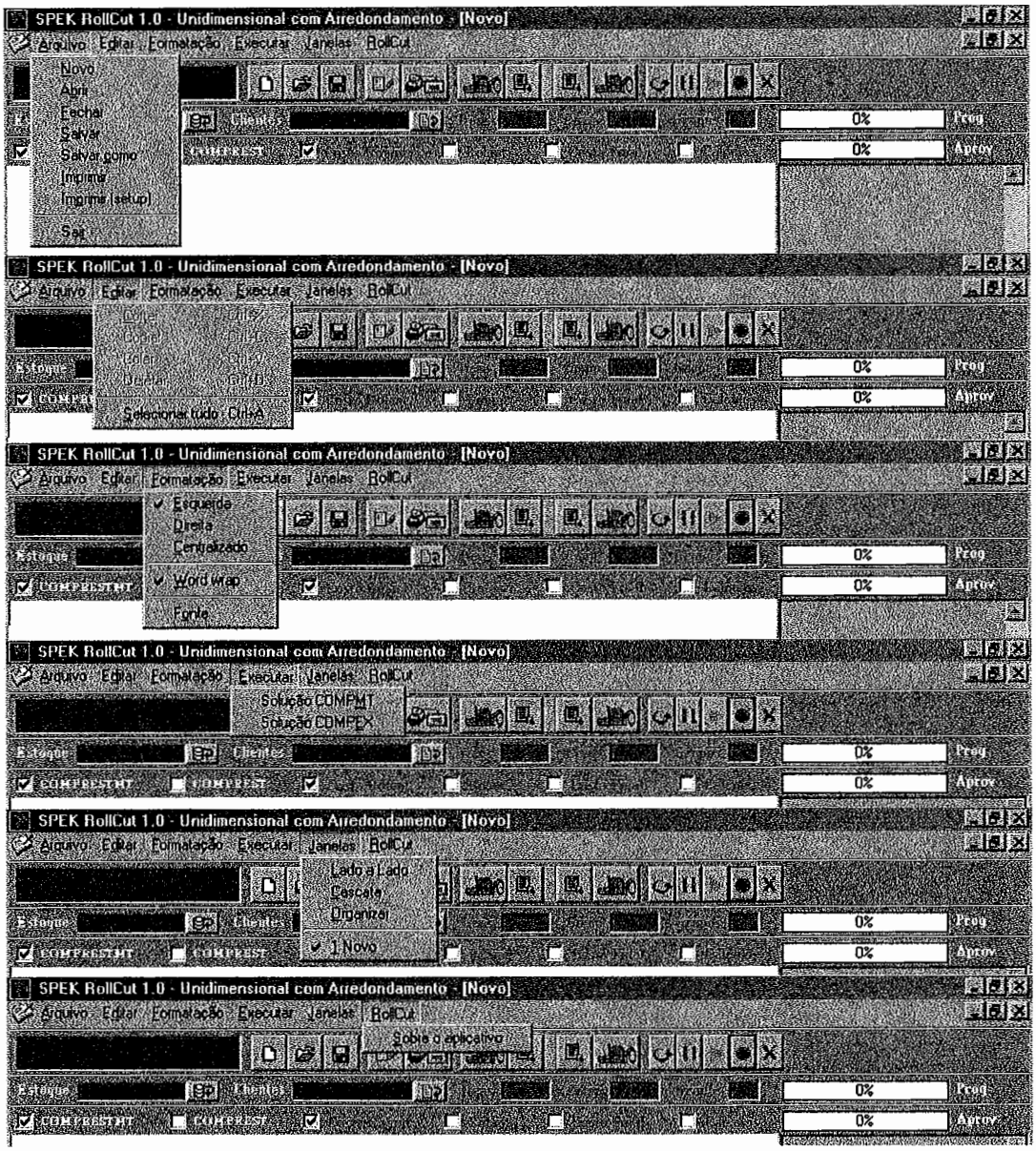

Figura 30 – Comandos do menu principal do RollCut.

Durante a execução do RollCut a barra de progresso **Prog** indica o progresso de execução do aplicativo, e a barra de progresso **Aprov** indica o aproveitamento de aço associado aos padrões de corte gerados (figura 31).

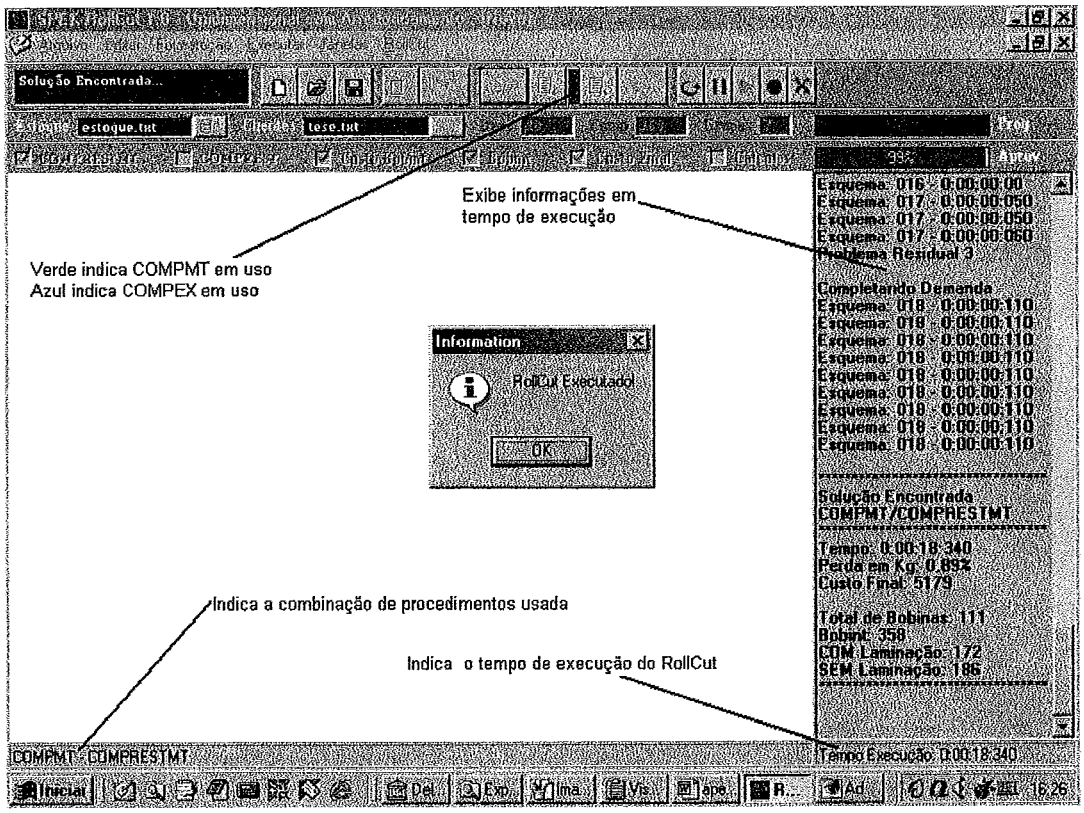

Figura 31 – Execução do RollCut.

Observe na figura 31 que alguns comandos do RollCut ficam desativados durante sua execução, são eles: botão para gerar um exemplo aleatório, botão para carregar arquivos de estoque e de clientes (os botões individuais também ficam desativados), botão para executar o COMPMT, botão para informar a solução pelo COMPMT, botão para executar o COMPEX, botão para informar a solução pelo COMPEX, todas as caixas de opções. Ao término da execução, ou com um cancelamento da mesma, o aplicativo retorna ao estado original.

2 – Fluxo de Dados entre os Procedimentos principais do RollCut

A seguir, apresentamos Diagramas de Fluxos de Dados do RollCut. Os objetos que utilizaremos nos DFD's com suas respectivas funcionalidades são apresentados na figura 32.

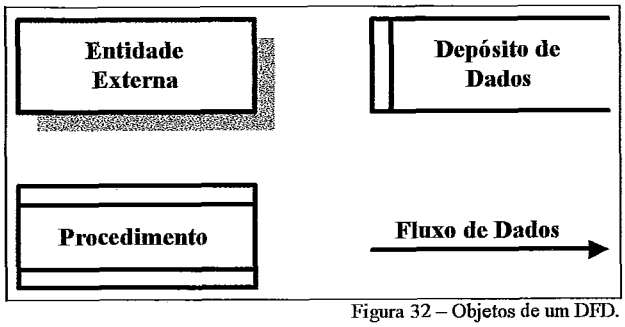

Figura 32 – Objetos de um DFD.

Na figura 33 apresentamos o DFD nível 1 do aplicativo RollCut.

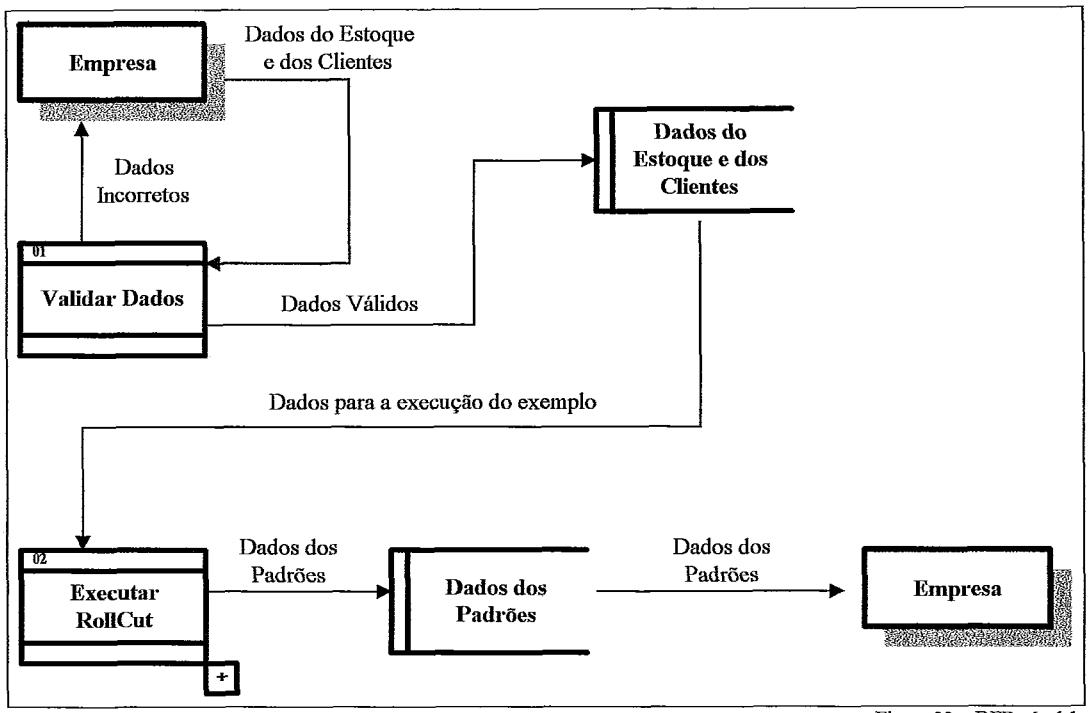

Figura 33 – DFD nível 1.

Na figura 34 apresentamos a "explosão" do procedimento "Executar RollCut".

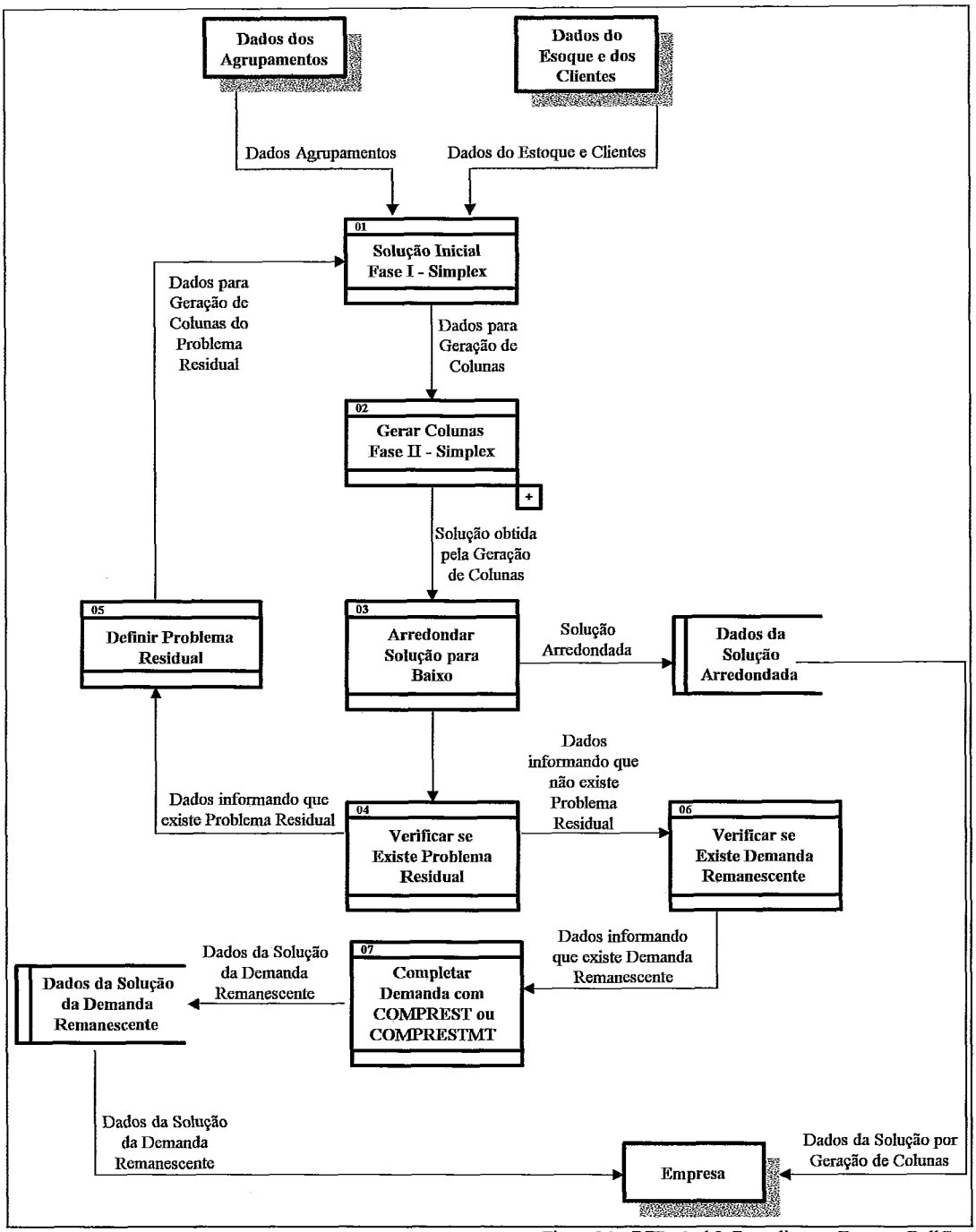

Figura 34 – DFD nível 2, Procedimento Executar RollCut.

Por fim, apresentamos na figura 35 a "explosão" do procedimento "Gerar Colunas".

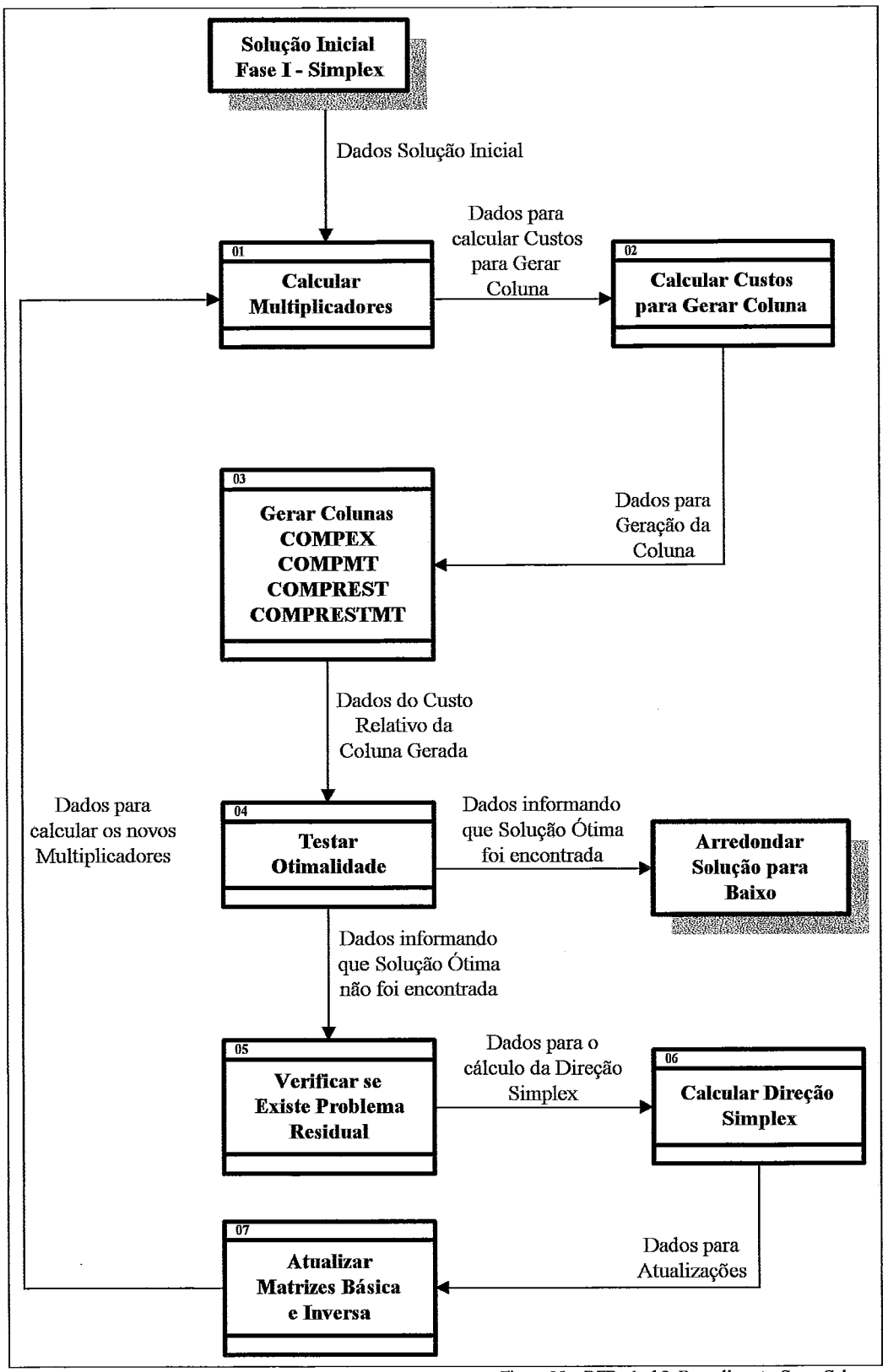

Figura 35 – DFD nível 3, Procedimento Gerar Colunas.

Apêndice E

Pseudocódigos de alguns Problemas da Mochila

1 – Algoritmo Guloso de Martello-Toth para Mochila 0-1

```
caux:=cap;
 z:=0;
 jestrela:=1;
 for j:=1 to maux do begin
   if p_aux[j] > caux
   then col[j]:=0
   else begin
     col[j]:=1;
     caux:=caux - p_aux[j];
     z:=z + u_aux[j];
   end;
   if u_aux[j] > u_aux[jestrela] then jestrela:=j;
 end;
 if ( z < u_aux[jestrela] ) and ( p_aux[jestrela] <= cap ) then begin
   z:=u_aux[jestrela];
   for j:=1 to maux do col[j]:=0;
   col[jestrela]:=1;
 end;
```

2 – Algoritmo Guloso de Martello-Toth para Mochila Restrita

```
caux:=cap;
 z:=0;
 jestrela:=1;
 for j:=1 to maux do begin
   col[j]:=trunc(caux/p_aux[j]);
   if col[j] > bitem[j] then col[j]:=bitem[j];
   caux:=caux - p_aux[j]*col[j];
   z:=z + u_aux[j]*col[j];
   if bitem[j]*u_aux[j] > bitem[jestrela]*u_aux[jestrela] then jestrela:=j;
 end;
if ( z < (bitem[jestrela]*u_aux[jestrela]) ) and ( (bitem[jestrela]*p_aux[jestrela]) <= cap ) then
begin
   z:=bitem[jestrela]*u_aux[jestrela];
   for j:=1 to maux do col[j]:=0;
   col[jestrela]:=bitem[jestrela];
 end;
```

3 – Algoritmo de Gilmore-Gomory para Mochila Irrestrita

```
{Passo 1 - Inicializacao}
IterGG:=1;
passo3:=true;
passo4:=true;
liminf:=0;
limsup:=0;
for i:=1 to Nsup do begin
  col_aux[i]:=0;
  col[i]:=0;
end;

{Passo 2 - Calcula uma solucao inicial}
col_aux[1]:=trunc(cap/p_aux[1]);
sobra:=cap - (p_aux[1]*col_aux[1]);
for i:=2 to maux do begin
  col_aux[i]:=trunc(sobra/p_aux[i]);
  sobra:=sobra - (p_aux[i]*col_aux[i]);
end;

{Passo 3 - Avalia a solucao corrente}
repeat
  IterGG:=IterGG + 1;
  if IterGG > LimiteMochila then Exit;{PARE, excedeu o limite maximo de iteracoes}
  zaux:=0;
  for i:=1 to maux do zaux:=zaux + (u_aux[i]*col_aux[i]);
  if liminf < zaux then begin
    liminf:=zaux;
    for i:=1 to maux do col[i]:=col_aux[i];
    z:=zaux;
  end;

{Passo 4 - Verifica se solucao atual e otima}
  repeat
    sobra:=0;
    s:=0;
    i:=1;
    while i <= maux do begin
      if col_aux[i] > 0 then s:=i;
      i:=i + 1;
    end;
    if s = 0 then begin
      passo3:=false;
      passo4:=false;
    end
    else begin
      if s = maux then begin
        u_aux[maux+1]:=0;
        p_aux[maux+1]:=1;
      end;
      limsup:=0;
```

```
    for i:=1 to s do begin
     if i <> s then begin
      limsup:=limsup + (u_aux[i]*col_aux[i]);
      sobra:=sobra + (p_aux[i]*col_aux[i]);
     end
     else begin
      limsup:=limsup + (u_aux[s]*(col_aux[s] - 1));
         sobra:=sobra + (p_aux[s]*(col_aux[s] - 1));
        end;
    end;
    sobra:=cap - sobra;
    limsup:=limsup + ((u_aux[s+1]/p_aux[s+1])*sobra);
   end;

{Passo 5 - Se nao for solucao otima faz a volta na arvore de decisoes}
   if (passo3 = true) or (passo4 = true) then begin
    if limsup <= liminf then begin
     col_aux[s]:=0;
     passo4:=true;
    end
    else begin
     col_aux[s]:=col_aux[s] - 1;
     for j:=s to (maux - 1) do begin
      sobra:=0;
      for i:=1 to j do sobra:=sobra + (p_aux[i]*col_aux[i]);
      sobra:=cap - sobra;
      col_aux[j+1]:=trunc(sobra/p_aux[j+1]);
     end;
     passo3:=true;
     passo4:=false;
    end;
   end;
  until (passo4 = false);
 until (passo3 = false);
```

4 – Algoritmo de Martello-Toth para Mochila Irrestrita

```
{Inicializar}
  1:
  for i:=1 to Nsup do begin
    col_aux[i]:=0;
    col[i]:=0;
    mMT[i]:=0;
  end;
  z:=0;
  zaux:=0;
  caux:=cap;
  u_aux[maux+1]:=0;
  p_aux[maux+1]:=big;
  IterMTU:=1;
  for s:=1 to maux do col_aux[s]:=0;
```

```
  {calculo bound MT}
  if maux >= 3 then begin
    cbarra:=cap - p_aux[1]*trunc(cap/p_aux[1]);
    clinha:=cbarra - p_aux[2]*trunc(cbarra/p_aux[2]);
    cchapeu:=(p_aux[2] - clinha)/p_aux[1];
    if cchapeu < 0 then cchapeu:=-trunc(abs(cchapeu))
    else begin
      if cchapeu > 0 then begin
        if (cchapeu - trunc(cchapeu)) > 0 then cchapeu:=trunc(cchapeu) + 1
        else cchapeu:=trunc(cchapeu);
      end;
    end;
    zlinha:=u_aux[1]*trunc(cap/p_aux[1]) + u_aux[2]*trunc(cbarra/p_aux[2]);
    aux:=(clinha + cchapeu*p_aux[1])*(u_aux[2]/p_aux[2]) - cchapeu*u_aux[1];
    if aux > 0 then aux:=trunc(aux)
    else begin
      if aux < 0 then begin
        if ( abs(aux) - trunc(abs(aux)) ) < 0 then aux:=-trunc(abs(aux)) - 1
        else aux:=trunc(aux);
      end;
    end;
    U0:=trunc(zlinha) + trunc(clinha*(u_aux[3]/p_aux[3]));
    U1:=trunc(zlinha) + trunc(aux);
    U3:=U0;
    if U1 > U3 then U3:=U1;
  end
  else begin
    if maux = 2 then begin
      cbarra:=cap - p_aux[1]*trunc(cap/p_aux[1]);
      U3:=u_aux[1]*trunc(cap/p_aux[1]) + trunc(cbarra*(u_aux[2]/p_aux[2]));
    end
    else begin
      U3:=trunc(cap*(u_aux[1]/p_aux[1]));
    end;
  end;
  {fim calculo bound}
  for s:=maux downto 1 do begin
    mMT[s]:=p_aux[s+1];
    for i:=s+2 to maux do if p_aux[i] < mMT[s] then mMT[s]:=p_aux[i];
  end;
  j:=1;
{fim inicializar}

{Construir nova solucao}
  2:
  IterMTU:=IterMTU + 1;
  if IterMTU > LimiteMochila then Exit;{PARE, excedeu o limite maximo de iteracoes}
  while p_aux[j] > caux do begin
    if z >= zaux + trunc(caux*(u_aux[j+1]/p_aux[j+1])) then goto 5
    else j:=j+1;
  end;
```

```
  yaux:=trunc(caux/p_aux[j]);
  vaux:=trunc( (caux - (yaux*p_aux[j]))*(u_aux[j+1]/p_aux[j+1]) );
  if z >= zaux + yaux*u_aux[j] + vaux then goto 5;
  if vaux=0 then goto 4;
{fim construir nova solucao}

{Salvar corrente solucao}
  3:
  caux:=caux - yaux*p_aux[j];
  zaux:=zaux + yaux*u_aux[j];
  col_aux[j]:=yaux;
  j:=j+1;
  if caux >= mMT[j-1] then goto 2;
  if z >= zaux then goto 5;
  yaux:=0;
{fim salvar corrente solucao}

{Atualizar solucao}
  4:
  z:=zaux + yaux*u_aux[j];
  for s:=1 to j-1 do col[s]:=col_aux[s];
  col[j]:=yaux;
  for s:=j+1 to maux do col[s]:=0;
  if z = U3 then Exit;{PARE, solucao encontrada}
{fim atualizar solucao}

{Backtrack}
  5:
  i:=0;
  for s:=1 to j-1 do if (col_aux[s] > 0) then i:=s;
  if i=0 then Exit;{PARE, solucao encontrada}
  caux:=caux + p_aux[i];
  zaux:=zaux - u_aux[i];
  col_aux[i]:=col_aux[i] - 1;
  if z >= zaux + trunc(caux*(u_aux[i+1]/p_aux[i+1])) then begin
     {Remover todos os itens tipo i}
     caux:=caux + p_aux[i]*col_aux[i];
     zaux:=zaux - u_aux[i]*col_aux[i];
     col_aux[i]:=0;
     j:=i;
     goto 5;
  end;
  j:=i+1;
  if caux - p_aux[i] >= mMT[i] then goto 2;
  h:=i;
{fim backtrack}

{Trocar item tipo i por tipo h}
    6:
    h:=h+1;
    if z >= zaux + trunc(caux*(u_aux[h]/p_aux[h])) then goto 5;
    if p_aux[h] = p_aux[i] then goto 6;
```

```
    if p_aux[h] > p_aux[i] then begin
      if (p_aux[h] > caux) or (z >= zaux + u_aux[h]) then goto 6;
      z:=zaux + u_aux[h];
      for s:=1 to maux do begin
        col[s]:=col_aux[s];
      end;
      col[h]:=1;
      if z = U3 then Exit;{PARE, solucao encontrada}
      i:=h;
      goto 6;
    end
    else begin
      if caux - p_aux[h] < mMT[h-1] then goto 6;
      j:=h;
      goto 2;
    end;
{fim trocar item tipo i por tipo h}
```

5 – Algoritmo de Yanasse-Soma para Mochila Irrestrita (restrição de igualdade)

```
{Inicializacao}
  {encontrar pesos minimo e maximo}
  pmin:=p_aux[1];
  for pointer:=2 to maux do begin
    if p_aux[pointer] < pmin then pmin:=p_aux[pointer];
  end;
  pmax:=p_aux[1];
  for pointer:=2 to maux do begin
    if p_aux[pointer] > pmax then pmax:=p_aux[pointer];
  end;
  {inicializacao com zeros}
  for pointer:=1 to Nmax do begin
   index[pointer]:=0;
   lucro[pointer]:=0;
  end;

  {inicialização dos vetores}
  for indice:=maux downto 1 do begin
    x_yansoma[indice]:=0;
    index[p_aux[indice]]:=indice;
    lucro[p_aux[indice]]:=u_aux[indice];
  end;

{Forward Moves}
  {há uma repetição de for's para evitar testes}
  for pointer:=pmin to (wmax - pmax) do begin
    if lucro[pointer] <> 0 then begin
      for indice:=index[pointer] to maux do begin
          aux1:=pointer + p_aux[indice];
          aux2:=lucro[pointer] + u_aux[indice];
```

```
        if lucro[aux1] < aux2 then Begin
          lucro[aux1]:=aux2;
          index[aux1]:=indice;
        end;
    end;
  end;
 end;
 for pointer:=wmax - pmax + 1  to (wmax - pmin) do begin
  if lucro[pointer] <> 0 then begin
    for indice:=index[pointer] to maux do begin
        aux1:=pointer + p_aux[indice];
        aux2:=lucro[pointer] + u_aux[indice];
        if (aux1 <= wmax) and (lucro[aux1] < aux2) then begin
          lucro[aux1]:=aux2;
          index[aux1]:=indice;
      end;
    end;
  end;
 end;

{Recuperar Solucao}
 IterYanSoma:=1;
 aux1:=wint;{compartimento de tamanho intermediario wmin <= wint <= wmax}
 zint:=lucro[wint];
 if zint = 0 then begin
  for i:=1 to Nsup do col[i]:=0;
 end
 else begin
  while aux1 <> 0 do begin
    x_yansoma[index[aux1]]:=x_yansoma[index[aux1]] + 1;
    aux1:=aux1 - p_aux[index[aux1]];
    IterYanSoma:=IterYanSoma + 1;
    if IterYanSoma > LimiteMochila then Exit;
  end;
  {atribuir solucao do YANSOMA ao vetor col}
  for i:=1 to maux do col[i]:=x_yansoma[i];
  for i:=(maux+1) to Nsup do col[i]:=0;
 end;
```

6 – Algoritmo de Yanasse-Soma para Mochila Irrestrita (restrição de desigualdade)

```
{Inicializacao}
 {encontrar pesos minimo e maximo}
 pmin:=p_aux[1];
 for pointer:=2 to maux do begin
  if p_aux[pointer] < pmin then pmin:=p_aux[pointer];
 end;
 pmax:=p_aux[1];
 for pointer:=2 to maux do begin
  if p_aux[pointer] > pmax then pmax:=p_aux[pointer];
 end;
```

```
 {inicializacao com zeros}
 for pointer:=1 to Nmax do begin
  index[pointer]:=0;
  lucro[pointer]:=0;
 end;
 {inicialização dos vetores}
 for indice:=maux downto 1 do begin
   x_yansoma[indice]:=0;
   index[p_aux[indice]]:=indice;
   lucro[p_aux[indice]]:=u_aux[indice];
 end;

{Forward Moves}
 {há uma repetição de for's para evitar testes}
 for pointer:=pmin to (wmax - pmax) do begin
   if lucro[pointer] <> 0 then begin
     for indice:=index[pointer] to maux do begin
          aux1:=pointer + p_aux[indice];
          aux2:=lucro[pointer] + u_aux[indice];
          if lucro[aux1] < aux2 then Begin
            lucro[aux1]:=aux2;
            index[aux1]:=indice;
          end;
     end;
   end;
 end;
 for pointer:=wmax - pmax + 1  to (wmax - pmin) do begin
   if lucro[pointer] <> 0 then begin
     for indice:=index[pointer] to maux do begin
          aux1:=pointer + p_aux[indice];
          aux2:=lucro[pointer] + u_aux[indice];
          if (aux1 <= wmax) and (lucro[aux1] < aux2) then begin
            lucro[aux1]:=aux2;
            index[aux1]:=indice;
       end;
     end;
   end;
 end;

{Recuperar Solucao}
 IterYanSoma:=1;
 aux1:=wint;{compartimento de tamanho intermediario wmin <= wint <= wmax}
 zint:=lucro[wint];
 while (zint = 0) and (wint >= Lmin[r]) do begin
   wint:=wint - 1;
   aux1:=wint;
   zint:=lucro[wint];
 end;
 if zint = 0 then begin
   for i:=1 to Nsup do col[i]:=0;
 end
```

```
 else begin
  while aux1 <> 0 do begin
    {Solucao}
    x_yansoma[index[aux1]]:=x_yansoma[index[aux1]] + 1;
    {Atualizacoes}
    aux1:=aux1 - p_aux[index[aux1]];
    IterYanSoma:=IterYanSoma + 1;
    if IterYanSoma > LimiteMochila then Exit;
  end;

  {Atribuir Solucao do YANSOMA ao vetor col}
  for i:=1 to maux do col[i]:=x_yansoma[i];
  for i:=(maux+1) to Nsup do col[i]:=0;
 end;
```

7 – Algoritmo de Yanasse-Soma para Mochila Restrita (restrição de desigualdade)

```
{Inicializacao}
 {encontrar pesos minimo e maximo}
 pmin:=p_aux[1];
 for pointer:=2 to maux do begin
  if p_aux[pointer] < pmin then pmin:=p_aux[pointer];
 end;
 pmax:=p_aux[1];
 for pointer:=2 to maux do begin
  if p_aux[pointer] > pmax then pmax:=p_aux[pointer];
 end;
 {inicializacao com zeros}
 for pointer:=1 to Nmax do begin
  index[pointer]:=0;
  lucro[pointer]:=0;
 end;
 {inicialização dos vetores}
 for indice:=maux downto 1 do begin
   x_yansoma[indice]:=0;
   index[p_aux[indice]]:=indice;
   lucro[p_aux[indice]]:=u_aux[indice];
 end;

{Forward Moves}
 {há uma repetição de for's para evitar testes}
 for pointer:=pmin to (wmax - pmax) do begin
   if lucro[pointer] <> 0 then begin
     for indice:=index[pointer] to maux do begin
          aux1:=pointer + p_aux[indice];
          aux2:=lucro[pointer] + u_aux[indice];
          if lucro[aux1] < aux2 then Begin
            lucro[aux1]:=aux2;
            index[aux1]:=indice;
          end;
     end;
   end;
 end;
```

```
 for pointer:=wmax - pmax + 1  to (wmax - pmin) do begin
  if lucro[pointer] <> 0 then begin
   for indice:=index[pointer] to maux do begin
       aux1:=pointer + p_aux[indice];
       aux2:=lucro[pointer] + u_aux[indice];
       if (aux1 <= wmax) and (lucro[aux1] < aux2) then begin
         lucro[aux1]:=aux2;
         index[aux1]:=indice;
     end;
   end;
  end;
 end;

{Recuperar Solucao}
 IterYanSoma:=1;
 aux1:=wint;{compartimento de tamanho intermediario wmin <= wint <= wmax}
 zint:=lucro[wint];
 while (zint = 0) and (wint >= Lmin[r]) do begin
  wint:=wint - 1;
  aux1:=wint;
  zint:=lucro[wint];
 end;
 if zint = 0 then begin
  for i:=1 to Nsup do col[i]:=0;
 end
 else begin
  while aux1 <> 0 do begin
   if x_yansoma[index[aux1]] + 1 <= bitem[index[aux1]] then begin
     x_yansoma[index[aux1]]:=x_yansoma[index[aux1]] + 1;
     aux1:=aux1 - p_aux[index[aux1]];
   end
   else aux1:=aux1 - 1;
   IterYanSoma:=IterYanSoma + 1;
   if IterYanSoma > LimiteMochila then Exit;
  end;
  {atribuir solucao do YANSOMA ao vetor col}
  for i:=1 to maux do col[i]:=x_yansoma[i];
  for i:=(maux+1) to Nsup do col[i]:=0;
 end;
```

Bibliografia

A

J. M. Van Den AKKER, C. A. J. HURKENS and M. W. P. SAVELSBERGH (2000). "Time-indexed formulations for machine scheduling problems: column generation." *Journal on Computing*, 12(2), 111-124.

M. AVRIEL, M. PENN and N. SHPIRER (2000). "Container ship stowage problem: complexity and connection to the coloring of circle graphs". *Discrete Applied Mathematics*, 103, 271-279.

D. J. ALOISE e N. MACULAN (1991). "Uma classe de algoritmos aproximativos decrescente para o problema *bin-packing*." *Revista Brasileira de Computação*, 6(3), 3-12.

A. G. de ALVARENGA e N. MACULAN (1986). "Um método aproximado para a solução do problema de cortes de placas a duas dimensões." *RBE (Caderno de Engenharia Naval)*, 1(1), 79-90.

A. R. S. AMARAL and M. WRIGHT (2001). "Efficient algorithm for the constrained two-dimensional cutting stock problem." *Operations Research*, 8, 3-13.

M. N. ARENALES (1993). *Uma teoria para o problema de corte*. Tese de Livre Docência, ICMSC-USP, São Carlos, S.P., Brasil.

M. N. ARENALES (1994). *Programação Linear*. Apostila, ICMSC-USP, São Carlos, S.P., Brasil.

M. N. ARENALES and R. MORABITO (1995). "An and/or graph approach to the solution of two dimensional non-guillotine cutting problems." *European Journal of Operational Research*, 84, 599-617.

M. N. ARENALES, R. MORABITO e H. H. YANASSE (editores) (1997). *O Problema de Corte e Empacotamento e Aplicações Industriais*. XX CNMAC, II ON PCE, Gramado, R.S., Brasil.

M. N. ARENALES, R. MORABITO e H. H. YANASSE (editores) (1999). Edição Especial sobre Corte e Empacotamento. *Pesquisa Operacional - SOBRAPO*, volume 19, edição número 2.

B

D. A. BABAYEV and S. S. MARDANOV (1994). "Reducing the number of variables in integer and linear programming problems." *Computational Optimization and Applications*, 3, 99-109.

D. A. BABAYEV, F. GLOVER and J. RYAN (1997). "A new knapsack solution approach by integer equivalent aggregation and consistency determination." *INFORMS Journal on Computing*, 9(1), 43-50.

E. BALAS and E. ZEMEL (1980). "An algorithm for large zero-one knapsack problems." *Operations Research*, 28, 1130-1154.

C. BARNHART, E. L. JOHNSON, G. L. NEMHAUSER, M. W. P. SAVELSBERG and P. H. VANCE (1998). "Branch-and-price: column generation for solving huge integer programs." *Operations Research*, 46(3), 316-329.

C. BARNHART, C. A. HANE, P. H. VANCE (2000). "Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems." *Operations Research*, 48(2), 318-326.

S. BAUM and L. E. TROTTER, Jr. (1982). "Finite checkability for integer rounding properties in combinatorial programming problems." *Mathematical Programming*, 22, 141-147.

J. E. BEASLEY (1985). "Algorithms for unconstrained two dimensional guillotine cutting." *Journal of the Operational Research Society*, 36(4), 297-306. Cf. Ref. R. MORABITO e M. N. ARENALES (1995).

J. E. BEASLEY (1998). *Population heuristic.* Tutorial. Imperial College, London, England.

R. BELLMAN (1954). "Some applications of the theory of dynamic programming - a review." *Operations Research*, 2, 275-288.

R. BELLMAN (1957). *Dynamic Programming.* Princeton University Press, Princeton, N.J., USA.

R. BELLMAN and S. E. DREYFUS (1962). *Applied Dynamic Programming.* Princeton University Press, Princeton, N.J., USA.

P. J. BILLINGTON, J. O. MacCLAIN and L. J. THOMAS (1986). "Heuristics for multilevel lot-sizing with a bottleneck." *Management Science*, 32(8), 989-1006.

A. BILLIONNET, A. FAYE, É. SOUTIF (1999). "A new upper bound for the 0-1 quadratic knapsack problem." *European Journal of Operational Research*, 112, 664-672.

E. E. BISCHOFF and M. D. MARRIOT (1990). "A comparative evoluation of heuristics for container loading." *European Journal of Operational Research*, 44, 267-276.

E. E. BISCHOFF, F. JANETZ and M. S. W. RATCLIFF (1995). "Loading pallets with non-identical items." *European Journal of Operational Research*, 84, 681-692.

E. E. BISCHOFF and M. S. W. RATCLIFF (1995). "Issues in the development of approaches to container loading." *Omega, The International Journal of Management Science*, 7(2), 145-151.

E. E. BISCHOFF and G. WÄSCHER (1995). "Cutting and packing." *European Journal of Operational Research*, 84, 503-505.

R. G. BLAND (1977). "New finite pivoting rules for the simplex method." *Mathematics of Operations Research*, 2(2), 103-107.

C

R. E. CAMPELLO e N. MACULAN (1994). *Algoritmos e Heurísticas*. Editora da Universidade Federal Fluminense, Niterói, R.J., Brasil.

M. CANTÙ (1996). *Dominando o Delphi 2 "A Bíblia"*. Makron Books do Brasil Editora Ltda., São Paulo, S.P., Brasil.

J. M. V. Valério de CARVALHO (1991). *Um problema de corte em duas fases*. Tese de Doutoramento, Universidade do Minho, Portugal.

J. M. V. Valério de CARVALHO and A. J. Guimarães RODRIGUES (1994). "A computer based interactive approach to a two-stage cutting-stock problem." *INFOR*, 32(4), 243-252.

J. M. V. Valério de CARVALHO and A. J. Guimarães RODRIGUES (1995). "An LP-based approach to a two-stage cutting-stock problem." *European Journal of Operational Research*, 84, 580-589.

J. M. V. Valério de CARVALHO (1998). "Exact solution of cutting-stock problems using column generation and branch-and-bound." *IFORS, Int. Trans. Opl. Res.*, 5(1), 35-44.

D. CATTRYSSE, J. MAES and L. N. Van WASSENHOVE (1990). "Set partitioning and column generation heuristics for capacitated dynamic lot-sizing." *European Journal of Operational Research*, 46, 38-47.

D. CATTRYSSE, M. SALOMON, R. KUIK and L. N. Van WASSENHOVE (1993). "A dual ascent and column generation heuristic for the discrete lot-sizing and scheduling problem with setup times." *Management Science*, 39(4), 477-486.

N. CHRISTOFIDES and C. WHITLOCK (1977). "An algorithm for two dimensional cutting problems." *Operations Research*, 25(1), 30-44.

C. CHU and J. ANTONIO (1999). "Approximation algorithms to solve real-life multicriteria cutting stock problems." *Operations Research*, 47(4), 495-508.

V. CHVÀTAL (1980). *Linear Programming.* W. H. Freeman and Company, New York, USA.

G. CINTRA (1998). *Algoritmos híbridos para problemas de corte unidimensional.* Dissertação de Mestrado, IME-USP, São Paulo, S.P., Brasil.

G. CINTRA e Y. WAKABAYASHI (1998). "Um algoritmo híbrido para o problema de corte unidimensional." *III Oficina Nacional de PCE, XX CNMAC*, Curitiba, P.R., Brasil, 79-96.

E. G. COFFMAN, Jr. and P. W. SHOR (1990). "Average case analysis of cutting and packing in two dimensions." *European Journal of Operational Research*, 44, 134-144.

S. A. COOK. "The complexity of theorem-proving procedures." *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 150-158.

S. A. COOK (1983). "An overview of computational complexity." *Communications of the ACM (Turing Award Lecture)*, 26, 400-408.

Van-Dat CUNG, M. HIFI and B. LE CUN (1997). *Constrained two dimensional cutting-stock problems: a best-first branch-and-bound algorithm.* Report of Research, Université de Versailles, France.

D

C. H. DAGLI (1990). Knowledge based systems for cutting-stock problems. *European Journal of Operational Research*, 44, 160-166.

G. B. DANTZIG (1957). "Discrete variable extremum problems." *Operations Research*, 5, 266-277.

G. B. DANTZIG and P. WOLFE (1959). "Decomposition principle for linear programs." *Operations Research*, 101-111.

G. B. DANTZIG (1963). *Linear Programming and Extensions*. Priceton University Press, N.J., USA.

V. P. DAZA, A. G. de ALVARENGA and J. de DIEGO (1995). "Exact solutions for constrained two dimensional cutting problems." *European Journal of Operational Research*, 84, 633-644.

B. L. DIETRICH and L. F. ESCUDORE (1989). *More coefficient reduction for knapsack-like constraints in 0-1 programs with variable upper bounds*. IBM T.J., Watson Research Center, RC 14389, Yorktown Heights, N. Y., USA.

K. A. DOWSLAND (1990). "Efficient automated pallet loading." *European Journal of Operational Research*, 44, 232-238.

K. A. DOWSLAND and W. B. DOWSLAND (1992). "Packing problems." *European Journal of Operational Research*, 56, 02-14.

K. A. DOWSLAND and W. B. DOWSLAND (1995). "Solution approaches to irregular nesting problems." *European Journal of Operational Research*, 84, 506-521.

A. DREXL and A. KIMMS (1997). "Lot-sizing and scheduling - survey and extensions." *European Journal of Operational Research*, 99, 221-235.

H. DYCKHOFF (1990). "A typology of cutting and packing problems." *European Journal of Operational Research*, 44, 145-159.

H. DYCKHOFF and U. FINKE (1992). *Cutting and Packing in Production and Distribuition*. Springer-Verlag Co., Heidelberg, Germany.

E

K. EISEMANN (1957). "The trim problem." *Management Science*, 3, 279-284.

I. EKELAND and R. TEMAM (1976). *Convex Analysis and Variational Problems*. North-Holand Publishing Company, Amsterdam.

F

I. R. FARIAS Jr., E. L. JOHNSON and G. L. NEMHAUSER (2000). *Branch-and-cut for combinatorial optimization problems without auxiliary binary variables*. Report of Research, State University of New York at Buffalo, Buffalo, New York, USA.

I. R. FARIAS Jr., E. L. JOHNSON and G. L. NEMHAUSER (2000). *Facets of the complementarity knapsack polytope*. Report of Research, State University of New York at Buffalo, Buffalo, New York, USA.

I. R. FARIAS Jr., E. L. JOHNSON and G. L. NEMHAUSER (2000). "A generalized assignment problem with special ordered sets: a polyhedral approach." *Mathematical Programming*, A 89, 187-203.

I. R. FARIAS Jr. and G. L. NEMHAUSER (2000). *A family of inequalities for the generalized assignment polytope*. Report of Research, State University of New York at Buffalo, Buffalo, New York, USA.

I. R. FARIAS Jr. (2000). *A family of facets for the p-median polytope*. Report of Research, State University of New York at Buffalo, Buffalo, New York, USA.

A. A. FARLEY (1990a). "Selection of stockplate characteristics and cutting style for two dimensional cutting-stock situations." *European Journal of Operational Research*, 44, 239-246.

A. A. FARLEY (1990b). "The cutting-stock problem in the canvas industry." *European Journal of Operational Research*, 44, 247-255.

D. FAYARD and G. PLATEAU (1975). Resolution of the 0-1 knapsack problem: comparison of methods. *Mathematical Programming*, 8, 272-301.

D. FAYARD and G. PLATEAU (1982). An algorithm for the solution of 0-1 knapsack problem, *Computing*, 28, 269-287.

D. FAYARD and V. ZISSIMOPOULOS (1995). "An approximation algorithm for solving unconstrained two dimensional knapsack problem." *European Journal of Operational Research*, 84, 618-632.

S. P. FEKETE and J. SCHEPERS (1998). "New classes of lower bounds for bin-packing problems." *Springer-Verlag*, 257-270.

C. E. FERREIRA, F. K. MIYAZAWA and Y. WAKABAYASHI (1999). "Packing squares into squares." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 223-237.

J. S. FERREIRA, M. A. NEVES and P. F. CASTRO (1990). "A two-phase roll cutting problem." *European Jounal of Operational Research*, 44, 185-196.

M. FIELDHOUSE (1990). "The duality gap in trim problems." *SICUP Bulletin, number 5*. Cf. Ref. G. SHEITHAUER and J. TERNO (1991).

H. FOERSTER and G. WÄSCHER (1996). *Simulated annealing for the order spread minimization problem in sequencing cutting patterns*. Technical Report, Universität Halle, Halle, Germany.

L. R. FORD, Jr. and D. R. FULKERSON (1962). *Flows in Networks*. Princeton University Press, N.J., USA.

J.C. FURTADO e L. A. LORENA (1998). "Algoritmo genético construtivo na otimização de problemas combinatoriais de agrupamento." *III Oficina Nacional de PCE, XX CNMAC*, Curitiba, P.R., Brasil, 29-42.

G

M. R. GAREY and D. S. JOHNSON (1979). *Computers and Intractability: A guide to the theory of NP-Completeness*. W. H. Freeman and Company, New York, USA.

T. GAU and G. WÄSCHER (1995). "CUTGEN1: A problem generator for the standard one dimensional cutting-stock problem." *European Journal of Operational Research*, 84, 572-579.

H. GEHRING, K. MENSCHNER and M. MEYER (1990). "A computer based heuristic for packing pooled shipment containers." *European Journal of Operational Research*, 44, 277-288.

D. D. GEMMILL and J. L. SANDERS (1990). "Approximate solutions for the cutting-stock portfolio problem." *European Journal of Operational Research*, 44, 167-174.

J. A. GEORGE and D. F. ROBINSON (1980). "A heuristic for packing boxes in a container." *Computer and Operations Research*, 7, 147-156. Cf. Ref. H. GEHRING, K. MENSCHNER and M. MEYER (1990).

J. A. GEORGE, J. M. GEORGE and B. W. LAMAR (1995). "Packing different sized circles into a rectangular container." *European Journal of Operational Research*, 84, 693-712.

P. C. GILMORE and R. E. GOMORY (1961). "A Linear Programming Approach to the Cutting Stock Problem." *Operations Research*, 9, 849-859.

P. C. GILMORE and R. E. GOMORY (1963). "A Linear Programming Approach to the Cutting Stock Problem, parth II." *Operations Research*, 14, 94-120.

P. C. GILMORE and R. E. GOMORY (1965). "Multistage cutting-stock problems of two and more dimensions." *Operations Research*, 13, 94-120.

P. C. GILMORE and R. E. GOMORY (1966). "The theory and computation of knapsack functions." *Operations Research*, 14, 1045-1074.

F. GLOVER (1965). "A multiphase dual algorithm for the zero-one integer programming problem." *Operations* Research, 13, 879-919.

F. GLOVER (1978). "Parametric branch-and-bound." *Omega, The International Journal of Management Science*, 6(2), 145-152.

B. L. GOLDEN (1976). "Approaches to the cutting-stock problem." *AIIE Transactions*, 8(2), 265-274.

M. GONDRAN and M. MINOUX (1990). *Graphs and Algorithms*. John-Wiley & Sons, ChiChester, England.

C. GOULIMIS (1990). "Optimal solutions for the cutting-stock problems." *European Journal of Operational Research*, 44, 197-208.

M. C. GRAMANI, P. M. FRANÇA e M. N. ARENALES (1998). "Um modelo de otimização do processo de cortagem acoplado ao planejamento da produção." *III Oficina Nacional de PCE, XX CNMAC*, Curitiba, P.R., Brasil, 97-107.

J. N. D. GUPTA, S. JEGANATHAN and C. WHITE (1999). "The cutting stock problem a give sequence of order lengths." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 159-168.

H

R. W. HAESSLER (1971). "A heuristic programming solution to a nonlinear cutting-stock problem." *Management Science*, 17(12), 793-802.

R. W. HAESSLER (1978). "A procedure for solving the 1.5 - dimensional coil slitting problem." *AIIE Transactions*, 10, 70-75.

R. W. HAESSLER (1975). "Controlling cutting pattern changes in one dimensional trim problems." *Operations Research*, 23(3), 483-493.

R. W. HAESSLER (1979). "Solving the two-stage cutting-stock problem." *Omega, The International Journal of Management Science*, 7(2), 145-151.

R. W. HAESSLER (1980). "A note on computational modifications to the Gilmore-Gomory cutting stock algorithm." *Operations Research*, 28(4), 1001-1005.

R. W. HAESSLER and F. B. TALBOT (1990). "Load planning for shipments of low density products." *European Journal of Operational Research*, 44, 289-299.

R. W. HAESSLER and M. A. VONDEREMBSE (1979). "A procedure for solving the master slab problem in the steel industry." *AIIE Transactions*, 11(2), 160-165.

P. HEISE and G. WÄSCHER (1996). *The bin-packing problem: a problem generator and some numerical experiments with FFD packing and MTP*. Technical Report, Universität Halle, Halle, Germany.

M. HIFI (1997). "An improvement of Viswanathan and Bagchi's exact algorithm for cutting-stock problems." *Computers and Operations Research*, 24(8), 727-736. Cf. Ref. Van-Dat CUNG, M. HIFI and B. LE CUN (1997).

M. HIFI (1999). "The strip cutting/packing problem: incremental substrip algorithms-based heuristics." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 169-188.

A. I. HINXMAN (1980). "The trim-loss and assortment problems: a survey." *European Journal of Operational Research*, 5, 8-18.

K. HITOMI (1970). "The analysis of optimal production speed and a practical solving method for a special linear programming." *Journal of Japan Industrial Management Association*, 45, 29-38. (*in japanese*). Cf. Ref. H. SUZUKI (1978).

E. HOROWITZ and S. SAHNI (1974). "Computing partitions with applications to the knapsack problem." *Journal of ACM*, 21, 277-292.

R. HOTO (1996). *Otimização no corte de peças unidimensionais com restrições de agrupamento*. Dissertação de Mestrado, ICMSC-USP, São Carlos, S.P., Brasil.

R. HOTO e M. ARENALES (1996). "Um problema de corte unidimensional com restrições de agrupamento e aplicações industriais." *I ON PCE*, IME-USP, São Paulo, S.P., Brasil.

R. HOTO e M. ARENALES (1997). "O problema do corte em bobinas de aço." *XX CNMAC*, Gramado, R.S., Brasil.

R. HOTO, M. N. ARENALES e N. MACULAN (1999). *O problema da mochila compartimentada*. Relatório Técnico, Departamento de Matemática, Centro de Ciências Exatas, Universidade Estadual de Londrina, Londrina, P.R., Brasil.

R. HOTO, N. MACULAN e M. N. ARENALES (1998). "O problema do corte em bobinas de aço via geração de colunas." *XXX Simpósio Brasileiro de Pesquisa Operacional*, Curitiba, P.R., Brasil, 267-268.

R. HOTO, N. MACULAN e M. N. ARENALES (2000). "Um provável *branch-and-bound* para uma versão simplificada do problema da mochila compartimentada." *XXXII Simpósio Brasileiro de Pesquisa Operacional*, Viçosa, M.G., Brasil.

R. HOTO, N. MACULAN, M. N. ARENALES e F. P. MARQUES (2001). "Um novo procedimento para o cálculo de mochilas compartimentadas." *Revista Investigação Operacional – APDIO – Lisboa – Portugal* (a ser publicado).

I

G. P. INGARGIOLA and J. F. KORSH (1973). "A reduction algorithm for zero-one single knapsack problems." *Management Science*, 20, 460-463.

J

M. H. JARDIM CAMPOS e N. MACULAN (1988). "Problemas de otimização relacionados ao corte de bobinas de papel." *16.o Colóquio Brasileiro de Matemática*, Rio de Janeiro, R.J., Brasil, 159-182.

M. H. JARDIM CAMPOS and N. MACULAN (1995). "Optimization problems related to the cut of paper reels: a dual approach." *Investigación Operativa*, 5(1), 45-53.

D. S. JOHNSON, A. DEMERS, J. D. ULLMAN, M. R. GAREY and R. L. GRAHAM (1974). "Worst-case performance bounds for one dimensional packing algorithms." *SIAM Journal on Computing*, 3(4), 299-325.

E. L. JOHNSON, G. L. NEMHAUSER and M. W. P. SAVELSBERGH (2000). "Progress in linear programming-based algorithms for integer programming." *Journal on Computing*, 12(1), 2-23.

M. P. JOHNSON, C. RENNICK and E. ZAK (1999). "One dimensional cutting stock problem in just-in-time environment." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 145-158.

R. E. JOHNSTON (1981). "OR in the paper industry." *Omega, The International Journal of Management Science*, 9(1), 43-50.

R. E. JOHNSTON and L. R. KHAN (1995). "Bounds for the nested knapsack problems." *European Journal of Operational Research*, 81, 154-165.

K

L. V. KANTOROVICH (1960). "Mathematical Methods of Organizing and Planning Production." *Management Science*, 6(4), 363-422.

R. KARP (1975). *Reducibility among combinatorial problems, complexity of computer computations*. R. E. Miller and J. W. Thatcher (eds.), Plenum Press, New York.

R. KARP (1986). "Combinatorics, complexity and randomness." *Communications of the ACM (Turing Award Lecture)*, 29, 98-109.

G. A. P. KINDERVATER and J. K. LENSTRA (1986). "An introduction parallelism in combinatorial optimization." *Discrete Applied Mathematics*, 14, 135-156.

P. J. KOLESAR (1967). "A branch-and-bound algorithm for the knapsack problem." *Management Science*, 13, 723-735.

J. S. H. KORNBLUTH (1986). "Accounting control in multiple objective linear programming." *Omega, The International Journal of Management Science*, 14(3), 245-249.

B. KRÖGER (1995). "Guillotineable bin-packing: a genetic approach." *European Journal of Operational Research*, 84, 645-661.

L

L. S. LASDON (1970). *Optimization Theory for Large Systems*. Macmillan.

M. LAURIÈRE (1976). An algorithm for the 0-1 knapsack problem, *Mathematical Programming*, vol. 14, 1-10.

L. S. B. S. LEITE (1998). *Programação Semi-Definida*. Versão da Tese de Doutoramento, COPPE-UFRJ, Rio de Janeiro, R.J., Brasil.

E. Yu-Hsien LIN (1998). "A bibliographical survey on some well-known non-standard knapsack problems." *INFOR*, 36(4), 274-317.

A. LINHARES e H. H. YANASSE (1998). "Minimização de pilhas em aberto e problemas relacionados: alguns resultados de complexidade." *III Oficina Nacional de PCE, XX CNMAC*, Curitiba, P.R., Brasil, 19-28.

L. LINS, S. LINS and R. MORABITO (1998). "Packing into a container." *III Oficina Nacional de PCE, XX CNMAC*, Curitiba, P.R., Brasil, 9-18.

D. G. LUENBERGER (1989). *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, California, USA.

M

N. MACULAN (1983). "Relaxation Lagrangienne: le problème du knapsack 0-1." *INFOR (Canadian Journal of Operational Research and Information Processing)*, 21, 315-327.

N. MACULAN (1998). *Programação Linear: Método do Simplex.* Apostila, COPPE-UFRJ, Rio de Janeiro, R.J., Brasil.

N. MACULAN, M. de Mendonça PASSINI, J. A. de Moura BRITO and I. LOISEAU (1999). "Column-generation in integer linear programming." *RAIRO – Operations Research* (a ser publicado).

N. MACULAN, M. de Mendonça PASSINI, J. A. de Moura BRITO and I. LOISEAU (1999). "Column-generation method for network designing." *RAIRO – Operations Research* (a ser publicado).

J. MAES and L. N. Van WASSENHOVE (1988). "Multiiten single level capacitated dynamic lot-sizing heuristics: a general review." *Journal Opl. Res. Soc*, 39(11), 991-1004.

O. MARCOTTE (1982). *Topics in Combinatorial Packing and Covering.* Ph.D. Thesis, School of Operations Research and Industrial Engineering, College of Engineering Cornell University, Ithaca, New York, USA.

O. MARCOTTE (1985). "The cutting-stock problem and integer rounding." *Mathematical Programming*, 33, 82-92.

O. MARCOTTE (1986). "An instance of the cutting-stock problem for which the rounding property does not hold." *Operations Research Letters*, 4(5), 239-243.

F. P. MARQUES (2000). *O problema da mochila compartimentada.* Dissertação de Mestrado, ICMSC-USP, São Carlos, S.P., Brasil.

F. P. MARQUES e M. N. ARENALES (1999). *O problema da mochila compartimentada.* Relatório FAPESP, ICMSC-USP, São Carlos, S.P., Brasil.

F. P. MARQUES e M. N. ARENALES (2000). "O problema da mochila compartimentada." *XXXII Simpósio Brasileiro de Pesquisa Operacional*, Viçosa, M.G., Brasil.

F. P. MARQUES e M. N. ARENALES (2001). "O problema da mochila compartimentada." *Artigo submetido à Revista Pesquisa Operacional – SOBRAPO.*

D. MAURICIO and N. MACULAN (2000). "A boolean penalty method for zero-one nonlinear programming." *Journal of Global Optimization.* 16, 343-354.

S. MARTELLO and P. TOTH (1977a). "An upper bound for the zero-one knapsack problem and a branch-and-bound algorithm." *European Journal of Operational Research*, 1, 169-175.

S. MARTELLO and P. TOTH (1977b). "Branch-and-bound algorithms for the solution of the general unidimensional knapsack problem." In M. Roubens (ed.). *Advances in Operations Research.* North-Holand, Amsterdam, 295-301.

S. MARTELLO and P. TOTH (1990). *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Chichester, England.

A. MARTIN and R. WEISMANTEL (1998). "The intersection of knapsack polyhedral and extensions." *Springer-Verlag*, 243-256.

R. K. MARTIN and D. J. SWEENEY (1983). "An ideal column algorithm for integer programs with special ordered sets of variables." *Mathematical Programming*, 26, 48-63.

V. MARTYNOV (1999). "Geometrical objects regular placement onto a stock sheet or strip." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 121-222.

C. McDIARMID (1999). "Pattern minimization in cutting stock problem." *Discrete Applied Mathematics*, 98, 121-130.

J. MERKER and G. WÄSCHER (1997). "Two new heuristic algorithms for the maximal planar layout problem." *OR Spektrum*, 19, 131-137.

R. MORABITO (1992). *Uma abordagem em grafo E/OU para o problema de empacotamento: aplicação ao carregamento de paletes e contêineres.* Tese de Doutoramento, USP/EESC, São Carlos, S.P., Brasil.

R. MORABITO e V. GARCIA (1996). "Otimização do corte de chapas retangulares na indústria de *hardboards*." *I Oficina Nacional de PCE*, São Paulo, S.P., Brasil, 43-49.

R. MORABITO e M. N. ARENALES (1992). "Um exame dos problemas de corte e empacotamento". *Pesquisa Operacional*, 12, 1-120.

R. MORABITO e M. N. ARENALES (1995). "Performance of two heuristics for solving large scale two dimensional guillotine cutting problems." *INFOR*, 145-155.

R. MORABITO e M. N. ARENALES (1996). "Staged and constrained two dimensional guillotine cutting problems: an and/or graph approach." *European Journal of Operational Research*, 94, 548-560.

R. MORABITO e M. N. ARENALES (1998). "Otimização das operações de cortes de chapas na indústria de móveis." *III Oficina Nacional de PCE, XX CNMAC*, Curitiba, P.R., Brasil, 43-52.

N

J. NELIβEN (1995). "How to use structural constraints to compute an upper bound for the pallet loading problem." *European Journal of Operational Research*, 84, 662-680.

G. L. NEMHAUSER and L. A. WOLSEY (1988). *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, USA.

G. L. NEMHAUSER, A. H. G. RINNOOY KAN and M. J. TODD (editores) (1989). *Handbooks in Operations Research and Management Science*. Vol 1, North-Holland.

C. NITSCHE, G. SCHEITHAUER, J. TERNO (1999). "Tighter relaxations for the cutting stock problem." *European Journal of Operational Research*, 112, 654-663.

O

J. F. OLIVEIRA and J. S. FERREIRA (1990). "An improved version of Wang's algorithm for two dimensional cutting problems." *European Journal of Operational Research*, 44, 256-266.

M. OLIVEIRA dos SANTOS e M. N. ARENALES (1998). "Um problema de dimensionamento de lotes multiestágio com restrições de capacidade e tempo de preparação." *XXX Simpósio Brasileiro de Pesquisa Operacional*, Curitiba, P.R., Brasil, 267-268.

P

C. H. PAPADIMITRIOU and K. STEIGLITZ (1982). *Combinatorial Optimization*. Prentice-Hall, Inc., Englewood Cliffs, N.J., USA.

N. PAPAMARKOS (2000). "On the approximation of real rational functions via mixed-integer linear programming." *Applied Mathematics and Computation*, 112, 113-124.

R. PENNA FRANCA, N. MACULAN, N. Y. SOMA and H. H. YANASSE (1996). "A distributed algorithm for the k-best knapsack problem." *I Oficina Nacional de PCE*, São Paulo, S.P., Brasil, 23-29.

M. A. PEREIRA (1993). *Uma abordagem matemática para o problema de corte e laminação de fitas de aço.* Dissertação de Mestrado, UNICAMP, Campinas, São Paulo, Brasil.

M. J. PINTO e M. N. ARENALES (1998). "O problema de corte bidimensional inteiro." *III Oficina Nacional de PCE, XX CNMAC*, Curitiba, P.R., Brasil, 69-77.

M. J. PINTO (1999). *O problema de corte de estoque inteiro.* Dissertação de Mestrado, USP – São Carlos, São Carlos, S.P., Brasil.

J. F. PIERCE (1966). *On the solution of integer cutting-stock problems by combinatorial optimization, parth I.* Technical Report IBM, 36, Y02, Cambridge Scientific Center, Cambridge. Cf Ref. A. I. HINXMAN (1980).

D. PISINGER (1995). *Algorithms for knapsack problems.* Ph.D. Thesis, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark.

G. PLATEAU and M. ELKINEL (1985). A hybrid algorithm for the 0-1 knapsack problem. *Methods of Operations Research*, 49, 277-293.

R

C. RIBEIRO, M. A. CARRAVILLA and J. F. OLIVEIRA (1999). "Applying constraint logic programming to the resolution of nesting problem." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 239-247.

M. N. ROCHA (1996). "Otimização de corte de barras." *I Oficina Nacional de PCE*, São Paulo, S.P., Brasil, 51-56.

M. N. ROCHA (1997). *Otimização de corte de barras.* Dissertação de Mestrado, Departamento de Ciência da Computação, UFMG, Belo Horizonte, M.G., Brasil.

T. G. ROCHA (1978). *Aprovisionamento de bobinas de cabos telefônicos: uma aplicação de programação combinatória.* Dissertação de Mestrado, COPPE-UFRJ, Rio de Janeiro, R.J., Brasil.

T. G. ROCHA, R. ARAUJO ALMEIDA, A. O. MORENO and N. MACULAN (1981a). "The administration of standard length telephone cable reels." *Annals of Discrete Mathematics*, North-Holand, 109-123.

T. G. ROCHA, R. ARAUJO ALMEIDA, A. O. MORENO and N. MACULAN (1981b). "On the solution of the standard length telephone cable reels problem." *IFORS*, North-Holand, 825-840.

S

H. M. SALKIN and C. A. KLUYVER (1975). "The knapsack problem: a survey". *Naval Research Logistics Quartely*, 22, 127-144.

P. SCHWERIN and G. WÄSCHER (1999). "A new lower bound for the *bin-packing* and its integration into MTP." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 111-129.

G. SHEITHAUER and J. TERNO (1991). "About the gap between the optimal values of the integer and continuous relaxation one dimensional cutting-stock problem." *Operations Research Proceedings*, 439-444.

G. SHEITHAUER and J. TERNO (1995). "The modified integer round-up property of the one dimensional cutting-stock problem." *European Journal of Operational Research*, 84, 562-571.

G. SHEITHAUER and J. TERNO (1999). "Facet-defining inequalities for the cutting stock problem." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 131-144.

J. L. de Castro SILVA e N. Y. SOMA (2001). "Algoritmos aproximativos para o empacotamento de *Bins* tridimensionais." *V ON PCE*, INPE, São José dos Campos, S.P., 38-45.

N. Y. SOMA and P. TOTH (1999). "On the critical item for subset sum problems." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 279-283.

N. Y. SOMA, H. H. YANASSE and N. MACULAN (1997). "O Problema da Mochila", *O Problema de Corte e Empacotamento e Aplicações Industriais*, capítulo 2. *II Oficina Nacional de PCE, XX CNMAC*, Gramado, R.S., Brasil.

N. Y. SOMA, H. H. YANASSE, A. S. I. ZINOBER and P. J. HARLEY (1995). "A polynomial approximation scheme for the subset sum problem." *Discrete Applied Mathematics*, 57, 243-253.

H. STADTLER (1990). "A one dimensional cutting-stock problem in the aluminium industry and its solution." *European Journal of Operational Research*, 44, 209-223.

Yu. G. STOYAN and M. V. NOVOZHILOVA (1999). "Non-guillotine placement of rectangles into a strip of given width." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 189-210.

C. A. A. SANCHES e N. Y. SOMA (2001). "Algoritmos paralelos para o problema da mochila." *V ON PCE*, INPE, São José dos Campos, S.P., 184-192.

H. SUZUKI (1978). "A generalized knapsack problem with variable coefficients." *Mathematical Programming*, 15, 162-176.

P. E. SWEENEY and R. W. HAESSLER (1990). "One dimensional cutting-stock decisions for rolls with multiple quality grades." *European Journal of Operational Research*, 44, 224-231.

P. E. SWEENEY and E. R. PATERNOSTER (1992). "Cutting and packing problems: a cathegorized application-oriented research bibliography". *Journal of Operational Research* Society, 43, 691-706.

M. M. SYSLO, N. DEO and J. S. KOWALIK (1983). *Discrete Optimization Algorithms with PASCAL Programs*. Prentice-Hall, Inc, Englewood Cliffs, N.J., USA.

T

R. TARJAN (1972). "Depth-first search and linear graph algorithms." *SIAM Journal on Computing*, 1, 146-160.

H. TEMPELMEIER and M. DERSTROFF (1996). "A lagrangean-based heuristic for dynamic multilevel multiiten constrained lot-sizing with setup times." *Management Science*, 42(5), 738-757.

J. TEGHEM, D. TYTTENS and E. L. ULUNGU (2000). "A interactive heuristic method for multi-objective combinatorial optimization." *Computers & Operations Research*, 27, 621-634.

J. A. TOMLIN (1970). "Branch-and-bound methods for integer and nonconvex programming", *Integer and Nonlinear Programming*, chapter 21. North-Holland Publishing Company, Amsterdam.

V

P. H. VANCE (1998). "Branch-and-price algorithms for the one dimensional cutting-stock problem." *Computational Optimization and Applications*, 9, 211-228.

P. H. VANCE, C. BARNHART, E. L. JOHNSON and G. L. NEMHAUSER (1994). "Solving binary cutting-stock generation and branch-and-bound." *Computational Optimization and Applications*, 3, 111-128.

F. VANDERBECK (2000). "On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm." *Operations Research*, 48(1), 111-130.

E. VIEIRA Neto (1999). *Uma heurística GRASP para o problema do corte unidimensional.* Dissertação de Mestrado, Centro de Ciências e Tecnologia, Universidade Estadual do Norte Fluminense, Campos dos Goytacazes, R.J., Brasil.

P. R. C. VILLELA and C. T. BORNSTEIN (1983). *An improved bound for the 0-1 knapsack problem*, Relatório Técnico do Programa de Engenharia de Sistemas e Computacao, COPPE-UFRJ, ES31-83.

P. R. C. VILLELA and C. T. BORNSTEIN (1987). An improved bound for the 0-1 knapsack problem, Szigma, 20(1), 89-94.

K. V. VISWANATHAN and A. BAGCHI (1993). "Best-first search methods for constrained two dimensional cutting-stock problems." *Operations Research*, 41(4), 768-776.

W

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

G. WÄSCHER (1990). "An LP based approach to cutting-stock problems with multiple objectives." *European Journal of Operational Research*, 44, 175-184.

G. WÄSCHER and T. GAU (1996). "Heuristics for the integer one dimensional cutting-stock problem: a computacional study." *OR Spektrum*, 18, 131-144.

G. WÄSCHER and J. MERKER (1997). "A comparative evaluation of heuristics for the adjacency problem in facility layout planning." *Int. J. Prod. Res.*, 35(2), 447-466.

Y

H. H. YANASSE (1996). "Uma proposta de redução para resolver um problema de sequenciamento em cortes." *I Oficina Nacional de PCE*, São Paulo, S.P., Brasil, 1-4.

H. H. YANASSE (1997). "Problemas de sequenciamento no contexto de corte", *O Problema de Corte e Empacotamento e Aplicações Industriais*, capítulo 3. *II Oficina Nacional de PCE, XX CNMAC*, Gramado, R.S., Brasil.

H. H. YANASSE, J. C. BECCENERI and N. Y. SOMA (1998). "An exact algorithm for a special case of a pattern sequencing problem." *III Oficina Nacional de PCE, XX CNMAC*, Curitiba, P.R., Brasil, 53-61.

H. H. YANASSE, J. C. BECCENERI and N. Y. SOMA (1999). "Bounds for a problem of sequencing patterns." *Revista Pesquisa Operacional – SOBRAPO*, 19(2), 249-277.

H. H. YANASSE and M. S. LIMEIRA (1998). "A branch-and-bound scheme for solving a pattern sequencing problem." *III Oficina Nacional de PCE, XX CNMAC*, Curitiba, P.R., Brasil, 63-68.

H. H. YANASSE and N. Y. SOMA (1987). "A new enumeration scheme for the knapsack problem." *Discrete Applied Mathematics*, 18, 235-245.

H. H. YANASSE, N. Y. SOMA and N. MACULAN (2000). "An algorithm for determining the k-best solutions of one-dimensional knapsack problem." *Revista Pesquisa Operacional – SOBRAPO*, 20(1), 117-134.

H. H. YANASSE, A. S. I. ZINOBER and R. G. HARRIS (1990). "Cutting-stock: board size selection." *IFORS*.

B. J. YUEN and K. V. RICHARDSON (1995). "Establishing the optimality of sequencing heuristics for cutting-stock problems." *European Journal of Operational Research*, 84, 590-598.

Z

S. H. ZANAKIS, J. R. EVANS and A. A. VAZACOPOULOS (1989). "Heuristic methods and applications: a categorized survey." *European Journal of Operational Research*, 43, 88-110.

G. ZHANG, X. CAI and C. K. WONG (2000). "Linear time-approximation algorithms for bin packing." *Operations Research Letters*, 26, 217-222.

N. ZHU, K. BROUGHAN (1998). "On aggregating two linear diophantine equations." *Discrete Applied Mathematics*, 82, 231-246.