UNIVERSIDADE ESTADUAL DE LONDRINA

EVERTON PEREIRA DA CRUZ

UMA ABORDAGEM HEURÍSTICA LINEAR PARA MOCHILAS COMPARTIMENTADAS RESTRITAS

Londrina
2010

EVERTON PEREIRA DA CRUZ

UMA ABORDAGEM HEURÍSTICA LINEAR PARA MOCHILAS COMPARTIMENTADAS RESTRITAS

Dissertação de Mes trado apres entada ao Departamento de Matemática da Universidade Estadual de Londrina, com o requisito parcial à obtenção do título de Mestre em Matemática Aplicada e Computacional.

Orientador: Prof. Dr. Robinson Sa muel Vieira Hoto

Londrina
2010

Catalogação na publicação elaborada pela Divisão de Processos Técnicos da Biblioteca Central da Universidade Estadual de Londrina.

Dados Internacionais de Catalogação-na-Publicação (CIP)

C957a Cruz, Everton Pereira da.
Uma abordagem heurística linear para mochilas compartimentadas restritas/ Everton Pereira da Cruz. – Londrina, 2010.
62f. : il.

Orientador: Robinson Hoto.
Dissertação (Mestrado em Matemática Aplicada e C omputacional) – Universidade Estadual de Londrina, Centro de Ciências Exatas e da Terra, Programa de Pós -Graduação em matemática Aplicada e Com putacional, 2010.
Inclui bibliografia.

1. Programação matemática – Teses. 2. Programação inteira – Teses. 3. Heurísticas – Teses. 4. Mochila compartimentada restrita – Teses. I. Hoto, Robinson. II. Universidade Estadual de Londrina. Centro de Ciências Exatas e da terra. Programa de Pós-Graduação em Matemática Aplicada e Computacional . III. Título.

CDU 519.85

EVERTON PEREIRA DA CRUZ

UMA ABORDAGEM HEURÍSTICA LINEAR PARA MOCHILAS COMPARTIMENTADAS RESTRITAS

Dissertação de Mestrado apresentada ao Departamento de Matemática da Universidade Estadual de Londrina, como requisito parcial à obtenção do título de Mestre em Matemática Aplicada e Computacional.

BANCA EXAMINADORA

Prof. Dr. Robinson Hoto
UEL – Londrina – PR

Profa. Dra. Maria Angélica de Oliveira Camargo-Brunetto
UEL – Londrina – PR

Prof. Dr. Naresh Kumar Sharma
UEL – Londrina – PR

Londrina, 09 de agosto de 2010.

DEDICATÓRIA

Ao meu Grande Deus, aos meus magníficos pais e meus sinceros amigos...companheiros fiéis..

.

AGRADECIMENTOS

Ao Prof. Dr. Orientador, que foi ne ste período de trabalho, um gra nde amigo, que me mostrou a beleza da Pesquisa Operacional.

A minha família, pela confiança, paciência e motivação.

Ao professor Naresh Kum ar Sharm a por m ostrar c omo é linda a Matemática.

Ao professor Ulysses Sodré pela dicas de LATEX.

Aos professores do PGMAC por acreditarem em mim.

A amiga Poliane Cristina de Farias, pelos bons mom entos neste período de estudo, que permitiu compartilhar as minhas dificuldades dos estudos.

Ao a migo Geovani Nunes Grapiglia, por ajudar nas correções e ser um ótimo amigo.

A amiga Camila Leão Cardozo, pelas risadas e momentos de estudos.

Aos a migos que aqui não expre ssei os nom es, m eus sinc eros agradecimentos.

Cruz, Everton Pereira da. **Uma abordagem heurística linear para mochilas compartimentadas restritas.** 2010. 62 f. Dissertação (Mestrado em Matemática Aplicada e Computacional) – Universidade Estadual de Londrina, Londrina, 2010.

RESUMO

O problem a da Mochila Com partimentada Rest rita tem na form ulação expressões com características não-lineares, a dquirindo com isso, um a m aior dificuldade na resolução. O objetivo deste trabalho é obter soluções pa ra o problem a, decom pondo o problem a em dois estágios, utilizando m étodos lineares, obtendo com isso, as heurísticas de Decomposição, Retro, W retro e a abordagem linear. Nas heurís ticas Retro e W retro são reform uladas as restrições de disponibilidade de itens da mochila, enquanto que na abordagem linear são reformuladas a função objetivo e a restrição de capacidade. A aborda gem linear ao Problema da Mochila Com partimentada Res trita é com parada com as heurísticas de Deco mposição, Retro e W retro, onde se verifi ca a superioridade da aborda gem linear, na qual retorna a melhor solução em um menor tempo.

Palavras-chave: Program ação. Program ação Matem ática. Program ação Inteira. Heurísticas. Mochila Compartimentada Restrita.

Cruz, Everton Pereira da. **Uma abordagem heurística linear para mochilas compartimentadas restritas.** 2010. 62 f. Dissertação (Mestrado em Matemática Aplicada e Computacional) – Universidade Estadual de Londrina, Londrina, 2010.

ABSTRACT

The Bounded Com partmentalized Knapsack Pr oblem has characteri stic of nonlinear expression in its formulation, giving rise to m ore dificulties in solving it. The objective of the present work is to obtain solutions, decom posing the problem into two steps, using linear methods, obtaining as su ch heuristic of decom position, Retro, Wretro and linear app roach. In Retro and Wretro heuristics are rewritten the re strictions about itens in the knapsack where as in the linear approach are rewritten the objec tive function and restriction of capacity. The linear approach to the B ounded Compartm entalized Knapsack Problem is com pared with the heuristics of decom position, Retro and W retro, to verify that it gives better solution in less time.

Keywords: Programming. Mathematical Programming. Integer Programming. Heuristics. Constrained Compartmentalised Knapsack.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

O Problema da Mochila é amplamente estudado visto que os problemas de Programação Inteira podem ser transformados em Problema da Mochila (Goldbarg, 2000). Um novo problema de mochila foi apresentado em (Hoto, Arenales e Maculan, 1999), e chamado por eles de Problema da Mochila Compartimentada. Posteriormente Hoto (2001), apresentou uma aplicação para o Problema da Mochila Compartimentada, onde se modela o problema do corte de bobinas de aço sujeito a laminação (caso restrito).

O vasto uso deste tipo de problema, pode ser visualizado desde o carregamento de containers até o processo de corte de materiais, em uma, duas ou três dimensões, conforme ilustrado pelas figuras abaixo:

(a) Empacotamento de cargas em Containers

(b) Corte de Bobinas de Papel

A figura (a)[1] pode dar uma pequena idéia do problema de organizar itens dentro de uma mochila de tamanho fixo L (capacidade do container). Já a figura (b)[2] representa o problema do corte de grandes bobinas de papel em dimensões menores, de forma a obter os pedidos específicos de cada cliente.

Este trabalho tem por propósito estudar três heurísticas: a de decomposição, a retro e a wretro. A partir deste estudo, apresenta-se a abordagem linear para o problema da mochila compartimentada restrita, como forma de facilitar a obtenção da solução ótima ou quase-ótima do problema da mochila compartimentada, visto que este tipo de problema é muito difícil de resolver usando métodos exatos, dada sua condição de *NP-Difícil*[3].

O trabalho está estruturado da seguinte forma:

Capítulo 1: Apresenta os vários tipos de problema da mochila, como forma de esclarecer o problema da mochila compartimentada.

[1] http://www.newscomex.com.br/adm/fotos/noticia_15725.bmp

Capítulo 2: Este capítulo exibe o conceito do problema da mochila compartimentada, além de expor um exemplo do problema. O problema da mochila compartimentada é estendido ao problema da mochila compartimentada restrita, com a inserção de três restrições.

Capítulo 3: Neste capítulo há o estudo das heurísticas responsáveis por encontrar soluções viáveis ao problema da mochila compartimentada restrita.

Capítulo 4: Este capítulo faz uso de uma abordagem linear ao problema da mochila compartimentada restrita, como outra opção as heurísticas apresentadas no capítulo anterior.

Capítulo 5: Resultados das simulações são apresentados neste capítulo.

Capítulo 6: A conclusão do trabalho é feito neste capítulo, expondo a vantagem da abordagem linear em relação as outras heurísticas e a necessidade da modificação. Também apresentam-se questões para futuros trabalhos.

[2]http://www.imprentaweb.com/imagenes/Papel_bobina_gigantedoble.gif

[3] Para um bom conhecimento sobre complexidade de algoritmos e suas classes (P, NP, entre outras) recomenda-se (Sedgewick, 1983), e em (Martello e Toth, 1997) encontra-se a demonstração de que os problemas da mochila pertencem a classe *NP-Difícil*.

CAPÍTULO 1
MOCHILAS

Este capítulo abordará os principais modelos matemáticos de problema da mochila encontrados na literatura, como forma de facilitar a compreensão do leitor.

1.1 Definição

O problema da mochila é útil em tomadas de decisões, e possivelmente a primeira citação foi feita por (Dantzig, 1957). Uma situação possível para descrevê-lo é:

> Uma pessoa deseja fazer uma viagem e necessita organizar sua mochila com objetos disponíveis em sua casa. Cada objeto tem uma utilidade e um peso. Desta forma, ela precisa organizar(escolher) os objetos para serem alojados em sua mochila de forma que a combinação escolhida seja a mais vantajosa (maior utilidade) e não ultrapasse a capacidade da mochila.

Na situação acima, foram utilizadas as palavras utilidade e peso, no qual elas podem representar a importância do objeto e o espaço ocupado por este na mochila. Na situação que se segue:

> Uma empresa quer investir um capital L em certas aplicações onde os rendimentos e custos são dados por u_i e l_i, respectivamente. Sabe-se que o valor para investir em todas as aplicações é maior que o capital disponível pela empresa. O objetivo da empresa é saber em quais aplicações pode-se investir, de forma a maximizar o retorno dado por estas.

as palavras utilidades e pesos foram representadas por rendimentos e custos, respectivamente, e essas decisões são frequentemente necessárias no mercado financeiro.

O problema da mochila é alvo de vasto estudo e na próxima seção, dar-se-á os tipos decorrentes das situações acima, como também suas formulações matemáticas.

1.2 Problemas envolvendo uma mochila

As várias formas de problemas da mochila serão abordados nesta e na próxima seção, para isso assume-se:

1. A utilidade, o peso e a quantidade de itens do tipo i serão representados por u_i, l_i e a_i, respectivamente, para $i = 1, \ldots, n$, onde n representa a quantidade de tipos de itens.

2. A capacidade da mochila será representada por L.

3. A ordenação dos itens será pela razão da utilidade pelo peso,

$$\frac{u_1}{l_1} \geq \frac{u_2}{l_2} \geq \ldots \geq \frac{u_n}{l_n} \tag{1.1}$$

   onde $\frac{u_i}{l_i}$ refere-se a eficiência da variável x_i. Além disso, a importância desta razão esta na obtenção de limitantes superiores para métodos de obtenção de solução, como pode ser visto em (Chvátal, 1983).

4. A soma dos pesos de todos os itens será superior a capacidade da mochila

$$\sum_{i=1}^{n} l_i > L \tag{1.2}$$

   para não obter somente a solução trivial.

5. O peso de cada item é menor que a capacidade da mochila, isto é,

$$l_i \leq L \text{ para } i \in \{1, \ldots, n\} \tag{1.3}$$

A partir destes conceitos pode-se visualizar o primeiro problema da mochila.

1.2.1 Problema da Mochila Inteira

No problema visualizado na primeira situação acima, dá-se o nome de **Problema da Mochila Inteira** e sua formulação matemática é:

$$\textbf{maximizar:} \quad \sum_{i=1}^{n} u_i a_i \tag{1.4}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_i a_i \leq L \tag{1.5}$$

$$a_i \in \mathbb{Z}, a_i \geq 0, \text{ para } i = 1, \ldots, n \tag{1.6}$$

Na formulação, tem-se:

1.4 *Objetivo do problema*: **Maximizar** a função objetivo definida na expressão, onde esta é representada pela soma dos produtos das utilidades (u_i) pela variável discreta (a_i) que define a quantidade de itens do tipo i;

1.5 *Restrição de capacidade*: A expressão impede que a soma dos pesos dos itens que compõem a solução exceda a capacidade da mochila;

1.6 *Domínio da variável*: Expressão responsável por restringir o domínio da variável a_i ao conjunto dos números inteiros não-negativos, isto é, $a_i \in \mathbb{Z}_+$.

Observe que nas variáveis do problema, há um limitante inferior que é o 0, mas não há um limitante superior, por isso este problema da mochila pode também ser chamado de **Problema da Mochila Irrestrito**, por não haver limitante superior para a quantidade de itens permitidos na mochila ou em estoque.

Admitindo-se (1.1), (1.2) e (1.3), pode-se obter uma solução para este problema por meio do método Branch-and-Bound:

Algoritmo 1.1: Método Branch-and-Bound para resolver Problema da Mochila Inteira

Entrada: n, u_i, l_i, L
Saída : a_i
Inicialização: $M := 0$, $k := 0$;
Repita
  Para todo $j := k+1, k+2, \ldots, n$ **faça**
    $a_j := \left\lfloor \left(L - \sum_{i=1}^{j-1} l_i a_i \right) / l_j \right\rfloor$;
  fim-Para
  $k := n$;
  Se $\sum_{i=1}^{n} u_i a_i > M$ **então**
    $M := \sum_{i=1}^{n} u_i a_i$;
    Para todo $i := 1, 2, \ldots, n$ **faça**
      $a_i^* := a_i$;
    fim-Para
  fim-Se
  Repita
    Repita
      Se $k = 1$ **então**
        Termina e retorna os a_i;
      senão
        $k := k - 1$;
      fim-Se
    enquanto $a_k = 0$;
    $a_k := a_k - 1$;
  enquanto $\sum_{i=1}^{k} u_i a_i + \frac{u_{k+1}}{l_{k+1}} \left(L - \sum_{i=1}^{k} l_i a_i \right) < M + 1$;
enquanto *true* ;
Retorno *Combinação mais vantajosa dos itens*;

e seus detalhes são descritos em (Chvátal, 1983). Em (Martello e Toth, 1997) existem outras formas de encontrar soluções por meio de algoritmos exatos ou aproximados. O problema da mochila inteira é um caso especial do próximo problema, sendo caracterizado por restringir a quantidade de itens.

1.2.2 Problema da Mochila Restrita

No problema anterior, há uma quantidade arbitrária de itens do tipo i disponíveis para compor a solução, porém, pode-se usar no máximo $d_i = \lfloor L/l_i \rfloor$[1], que é a quantidade de vezes que o item do tipo i, com peso l_i pode ser inserido na mochila de capacidade L.

O **Problema da Mochila Restrita** limita a quantidade de cada item do tipo i disponível para compor a solução. A formulação é:

$$\textbf{maximizar:} \quad \sum_{i=1}^{n} u_i a_i \tag{1.7}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_i a_i \leq L \tag{1.8}$$

$$0 \leq a_i \leq d_i, \text{ para } i = 1, \ldots, n \tag{1.9}$$

$$a_i \in \mathbb{Z}, \text{ para } i = 1 \ldots n \tag{1.10}$$

onde d_i é a quantidade máxima de itens do tipo i disponíveis para compor ou permitidos na mochila.

O problema da mochila restrita pode ser solucionada pelo algoritmo (1.1) visto anteriormente, feito a seguinte modificação no código

$$a_j = \min \left\{ \left\lfloor \left(L - \sum_{i=1}^{j-1} l_i a_i \right) / l_j \right\rfloor, d_j \right\} \tag{1.11}$$

O problema seguinte conhecido como problema da mochila 0-1, é muito importante, pois outros problemas podem ser solucionados a partir da transformação nele. Porém, existe um outro problema da mochila, muito estudado pela suas características, visto na próxima subseção.

1.2.3 Problema da Mochila 0-1

Na introdução do livro de Martello e Toth, o problema da mochila 0-1 é visualizado como um caso especial do problema da mochila restrita. Mesmo sendo um caso particular, é amplamente estudado, pois, existem vários métodos

[1] função piso: retorna o maior inteiro menor que L/l_i

de resolvê-lo. Tome $d_i = 1$ no problema anterior e tem-se o **Problema da Mochila 0-1**. A sua formulação matemática é:

$$\textbf{maximizar:} \quad \sum_{i=1}^{n} u_i a_i \tag{1.12}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_i a_i \leq L \tag{1.13}$$

$$a_i \in \{0, 1\}, \text{ para } i = 1, \ldots, n \tag{1.14}$$

Em (Martello e Toth, 1997) são descritos métodos exatos e heurísticos para resolver o problema da mochila 0-1. Estes métodos assumem a ordenação dos itens dada por (1.1), e são novamente indexados os índices dos itens, garantindo com isso um melhor desempenho na busca pela solução ótima. A existência de vários estudos sobre o problema da Mochila 0-1 se dá pela possibilidade de resolver o problema da mochila restrita através dos métodos usados para obter a solução do problema da mochila 0-1, transformando-se cada item do tipo i numa série de $\lfloor log_2 d_i \rfloor$ itens, cujas utilidades e pesos são respectivamente

$$(u_i, l_i), (2u_i, 2l_i), (4u_i, 4l_i), \ldots \tag{1.15}$$

sendo que a utilidade (respectivamente peso) total é igual a $d_i u_i$ (respectivamente $d_i l_i$).

Este problema é um dos mais estudados, pois constitui a base para alguns outros problemas, que podem ser agrupados na mesma categoria, todos com complexidade *NP-Difícil*.

Nas próximas subseções serão exibidos outros problemas decorrentes deste visto acima.

1.2.4 Problema da Mochila de Escolha Múltipla

Este próximo problema é conseqüência da modificação na organização dos tipos dos itens no Problema da Mochila 0-1.

Suponha que estes n itens sejam organizados em m classes A_k distintas de forma a incluir apenas um item de cada classe na solução, e este problema é chamado de **Problema da Mochila de Escolha Múltipla**, sendo sua formulação:

$$\textbf{maximizar:} \quad \sum_{i=1}^{n} u_i a_i \tag{1.16}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_i a_i \leq L \tag{1.17}$$

$$\sum_{i \in A_k} a_i = 1, \text{ para } k = 1 \ldots m \tag{1.18}$$

$$a_i \in \{0,1\}, i \in N = \{1, \ldots, n\} = \bigcup_{k=1}^{m} A_k \tag{1.19}$$

A diferença principal entre este problema e o Problema da Mochila 0-1 está na restrição (1.18), no qual restringe a inclusão de um item por cada classe.

1.2.5 Problema da Soma de Subconjuntos

Outra modificação no Problema da Mochila 0-1, acontece quando o valor da utilidade é igual ao valor do próprio peso do item do tipo i, isto é, $u_i = l_i$. Assim, o problema é chamado de **Problema da Soma de Subconjuntos** ou também de **Problema da Mochila de Valor Independente**. A formulação é exibida logo abaixo:

$$\textbf{maximizar:} \quad \sum_{i=1}^{n} l_i a_i \tag{1.20}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_i a_i \leq L \tag{1.21}$$

$$a_i \in \{0,1\}, \text{ para } i = 1, \ldots, n \tag{1.22}$$

O Problema da Soma de Subconjuntos é um caso particular do Problema da Mochila 0-1, como pode-se observar.

1.2.6 Problema do Tronco

Neste problema também decorrente do Problema da Mochila 0-1, houve três modificações significativas, sendo a primeira, a mudança do critério de **maximizar** para **minimizar**, a segunda, o somatório foi feito somente sobre os a_i (todos os itens tem utilidade $u_i = 1$), e terceiro, $\sum_{i=1}^{n} l_i a_i = L$ (a combinação dos pesos dos itens tem que ser igual a capacidade da mochila). Este novo problema é conhecido como **Problema do Troco**. O problema definido matematicamente é:

$$\textbf{minimizar:} \quad \sum_{i=1}^{n} a_i \tag{1.23}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_i a_i = L \tag{1.24}$$

$$a_i \in \{0, 1\}, \text{ para } i = 1, \ldots, n \tag{1.25}$$

A formulação dele difere dos outros problemas por querer minimizar a quantidade de itens que são inseridos na mochila, descrita pela função objetivo. Na comparação entre a função objetivo do problema atual com a do problema da mochila 0-1, dada em (1.12), nota-se a modificação $u_i = 1$, e conclui-se que a utilidade não é muito relevante neste problema.

1.2.7 Problema da Mochila 0-1 Multidimensional

Neste problema da mochila, acrescenta-se a restrição que, ao inserir os itens, há um pagamento p_i por cada item inserido e há uma limitação de P unidades disponíveis. Este problema é também chamado de **Problema da Mochila n-Dimensional**. A formulação matemática é:

$$\textbf{maximizar:} \quad \sum_{i=1}^{n} u_i a_i \tag{1.26}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_i a_i \leq L \tag{1.27}$$

$$\sum_{i=1}^{n} p_i a_i \leq P \tag{1.28}$$

$$a_i \in \{0, 1\}, \text{ para } i = 1, \ldots, n \tag{1.29}$$

Nesta formulação, inseriu-se apenas uma restrição chamada de *pagamento*, dada por (1.28), mas há a possibilidade de inserir até n restrições ao problema.

1.2.8 Problema da Mochila Quadrática

Neste problema a função objetivo ou quaisquer restrições são compostas de variáveis quadráticas, assim sendo, este problema não é um problema linear. Uma formulação matemática possível encontra-se abaixo:

$$\textbf{maximizar:} \quad a^T Q a + c^T a \tag{1.30}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_i a_i \leq L \tag{1.31}$$

$$a_i \in \mathbb{Z}, a_i \geq 0, \text{ para } i = 1, \ldots, n \tag{1.32}$$

A formulação deste problema é quadrático, pois, nota-se que a função objetivo é quadrática, dada em (1.30).

1.3 Problemas Envolvendo *m* Mochilas

Na seção anterior, abordou-se problemas nos quais assumiu-se a existência de somente uma mochila, porém, nesta seção, abordar-se-á problemas da mochila com múltiplas mochilas.

1.3.1 Problema de Múltiplas Mochilas 0-1

Este problema é caracterizado pela existência de outros problemas da mochila 0-1 em sua formulação. Suponha que há m mochilas com capacidades L_j e n itens com utilidades u_i e pesos l_i. Pressupõe-se que há mais itens do que mochilas, isto é, $n \geq m$. O problema é formulado como:

$$\textbf{maximizar:} \quad \sum_{j=1}^{m}\sum_{i=1}^{n} u_i a_{ij} \tag{1.33}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_i a_{ij} \leq L_j, \text{ para } j = 1, \ldots, m \tag{1.34}$$

$$\sum_{j=1}^{m} a_{ij} \leq 1, \text{ para } i = 1, \ldots, n \tag{1.35}$$

$$a_{ij} \in \{0, 1\}, \text{ para } i = 1, \ldots, n \text{ e } j = 1, \ldots, m \tag{1.36}$$

Note que a variável de decisão agora é indexada por dois indíces i e j, onde i representa os itens e j as mochilas. Com isso, $a_{ij} = 1$ se o item i for inserido na mochila j e $a_{ij} = 0$ caso o contrário.
A restrição (1.35) garante que o item do tipo i pode ser incluído no máximo uma única vez em somente uma das m mochilas, sendo que estes itens tem utilidade e peso fixos em relação as mochilas.

O próximo problema tem na sua formulação uma dependência da utilidade (peso) do item em relação a mochila, e isto é verificado no exemplo descrito na próxima subseção.

1.3.2 Problema de Designação Generalizada

Este problema se assemelha ao anterior, porém, suas utilidades e pesos dependem da mochila a qual o item for inserido. Assim, a utilidade e o peso são dados para cada item em relação a cada mochila, necessitando de mais um índice para as características dos itens, isto é, tornando agora u_{ij} e l_{ij}. O problema matematicamente escrito é:

$$\textbf{maximizar:} \quad \sum_{j=1}^{m}\sum_{i=1}^{n} u_{ij}a_{ij} \tag{1.37}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_{ij}a_{ij} \leq L_j, \text{ para } j = 1, \ldots, m \tag{1.38}$$

$$\sum_{j=1}^{m} a_{ij} = 1, \text{ para } i = 1, \ldots, n \tag{1.39}$$

$$a_{ij} \in \{0, 1\}, \text{ para } i = 1, \ldots, n \text{ e } j = 1, \ldots, m \tag{1.40}$$

A restrição (1.39), expressa que todo item do tipo i pertence a uma única mochila, onde m representa a quantidade de mochilas e n a quantidade do tipo de itens. Este problema pode ser visualizado pela seguinte situação:

> Numa máquina há n tarefas para serem processadas por m processadores (processos), sabe-se que cada tarefa i tem uma vantagem u_{ij} e um gasto de recursos l_{ij} em relação ao processador (processo) j. Sabe-se que há disponíveis recursos L_j para cada processador (processo). O problema consiste em encontrar a forma de distribuir as tarefas de forma a maximizar estes processos, pois em certos aparelhos, há a necessidade de saber o maior tempo de trabalho, como forma de comparar seu desempenho em relação a outros aparelhos, entre outras coisas.

Este problema tem uma grande importância nas resoluções de problemas práticos, sendo que o próximo exemplo aborda o caso quando as mochilas tem capacidades (por exemplo, contaniers que tenham a mesma capacidade) iguais.

1.3.3 Múltiplas Mochilas Idênticas

Suponha que as capacidades L_j das m mochilas sejam iguais a L para todo $j = 1, \ldots, m$. Este problema procura como solução a quantidade mínima de mochilas necessárias e quais os itens que irão compô-las. Para isso, é criada a variável y_j tal que $y_j = 1$ se a mochila j for usada e $y_j = 0$ caso o contrário. Abaixo encontra-se a formulação matemática:

$$\textbf{minimizar:} \quad \sum_{j=1}^{m} y_j \tag{1.41}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_{ij} a_{ij} \leq L_j y_j, \text{ para } j = 1, \ldots, m \tag{1.42}$$

$$\sum_{j=1}^{m} a_{ij} = 1, \text{ para } i = 1, \ldots, n \tag{1.43}$$

$$a_{ij} \in \{0, 1\}, \text{ para } i = 1, \ldots, n \text{ e } j = 1, \ldots, m \tag{1.44}$$

$$y_j \in \{0, 1\}, \text{ para } j = 1, \ldots, m \tag{1.45}$$

Até este momento, pode-se observar que todos os problemas de mochila visto anteriormente, têm em sua formulação o princípio de organização dos itens sobre mochilas. O próximo problema da mochila, tem em sua formulação uma modificação na forma de gerar a mochila, pois as mochilas subseqüentes serão subdivisões da(s) mochila(s) anterior(es).

Este próximo problema, se assemelha muito ao problema de estudo deste trabalho, visto que o nome do problema é Problema da Mochila Encapsulada e o trabalho aqui representado aborda o Problema da Mochila Compartimentada.

1.3.4 Problema da Mochila Encapsulada

Neste problema da mochila, a característica principal é a divisão da mochila em diversos compartimentos de tamanhos conhecidos. A formulação exibida aqui, foi proposta por Robert E. Johnston e Lutfar R. Khan (Johnston e

Khan, 1995). Há outras formulações para o problema da mochila encapsulada, mas foi escolhida este, pois os autores Johnston e Khan (Johnston e Khan, 1995) criam o modelo e analisam um limitante para este. Neste problema proposto pelos autores do artigo, criam-se previamente divisões de tamanhos conhecidos na mochila, e o problema é inserir os itens na mochila, onde as subdivisões são os vários estágios do processo. O problema da mochila encapsulada proposto por Johnston e Khan é:

$$\textbf{maximizar:} \quad \sum_{i=1}^{n} u_i a_{i1}^1 - \sum_{j=1}^{n} \sum_{k=2}^{S} c_k y_{jk} \tag{1.46}$$

$$\textbf{sujeito a:} \quad \sum_{i=1}^{n} l_i a_{i1}^1 \leq L_1 \tag{1.47}$$

$$\sum_{i=1}^{n} l_i a_{ij}^k \leq L_k y_{jk}, \tag{1.48}$$

$$\text{para } j = 1, \ldots, n, \ k = 2, \ldots, S$$

$$a_{i1}^1 \leq \sum_{j=1}^{n} a_{ij}^k, \tag{1.49}$$

$$\text{para } i = 1, \ldots, n, \ k = 1, \ldots, S$$

$$\max_j (a_{pj}^k + a_{qj}^k) \leq \max_j (a_{pj}^{k-1} + a_{qj}^{k-1}), \tag{1.50}$$

$$\text{para } p = 1, \ldots, n, \ q = 1, \ldots, n,$$
$$k = 3, \ldots, S, \text{ e } p \neq q$$

$$a_{ij}^k \in \{0, 1\}, \tag{1.51}$$

$$\text{para } i, j = 1, \ldots, n, \ k = 1, \ldots, S$$

$$y_{jk} \in \{0, 1\}, \tag{1.52}$$

$$\text{para } j = 1, \ldots, n, \ k = 1, \ldots, S$$

A função objetivo é composta da soma de todas as utilidades de todos os itens que compõem a mochila no primeiro estágio menos os custos com as mochilas criadas em cada estágio, omitindo o primeiro estágio, pois este admite-se não ter custo.

A variável a_{ij}^k representa se o item de índice i vai compor a subdivisão j na mochila no estágio k e y_{jk} representa se a subdivisão j vai compor a mochila no estágio k, e as restrições garantem a organização dos itens nos estágios posteriores.

O próximo capítulo apresenta e detalha o problema da mochila compartimentada, que foi introduzido por Hoto (Hoto, Arenales e Maculan, 1999), sendo caracterizado por criar subdivisões de tamanhos variáveis na mochila.

CAPÍTULO 2
PROBLEMA DA MOCHILA COMPARTIMENTADA

No capítulo anterior, o Problema da Mochila Encapsulada criava previamente subdivisões de tamanhos conhecidos, porém, uma proposta apresentada por Hoto e Arenales no trabalho (Hoto, Arenales e Maculan, 1999) estudou um novo problema de mochila, chamado por eles de Problema da Mochila Compartimentada. Em trabalhos anteriores não houve na literatura esta variante de problema da mochila, descrita de maneira formal.

Como se observou no capítulo anterior, o Problema da Mochila Encapsulada tem previamente definido o tamanho das subdivisões, sendo que no Problema da Mochila Compartimentada, os compartimentos tem tamanhos desconhecidos, definidos pelos itens que os compõem. Outra diferença está na organização dos itens, que são dispostos em classes como no Problema da Mochila de Escolha Múltipla, e esta característica não está presente no problema da mochila encapsulada. Nesta formulação, aumentou-se a dificuldade do problema, por causa da inserção de outras restrições, e esta nova formulação de problema da mochila aparece em muitos problemas reais, sendo por isso, alvo de um grande estudo, como pode ser visto na citação (Oliveira e Wäscher, 2007). Alguns trabalhos recentes são: início do trabalho sobre mochila compartimentada (Hoto, Arenales e Maculan, 1999), aplicação do problema ao corte de bobinas de aço (Hoto, 2001), problema da mochila compartimentada restrita (Marques e Arenales, 2002), geração de padrões (Hoto *et al.*, 2003), aplicação no corte de estoques (Marques, 2004), heurísticas (Spolador, 2005), (Hoto *et al.*, 2006) e (Marques e Arenales, 2007), problema de corte de estoque (Hoto, Arenales e Maculan, 2007) e planejamento de cortes (Hoto, Spolador e Maculan, 2007).

2.1 Definição

O Problema da mochila compartimentada tem como objetivo propor a inserção dos itens na mochila, porém, obedecendo o critério de que estes são organizados em classes disjuntas. Assim, o problema também pretende criar compartimentos dentro da mochila de tal forma que cada compartimento criado, a ele é alocado itens pertencentes a classe que irá gerar o comparti-

mento. Sabe-se ainda que cada compartimento tem um custo pela sua criação na mochila, por exemplo, custo com a delimitação do compartimento.

Formalmente o problema da mochila compartimentada é definido como: Seja uma mochila com capacidade L e n tipos de itens com utilidades u_i e pesos l_i. Estes itens são organizados em classes $N_1, N_2, \ldots, N_q$ de tal forma que $\bigcup_{i=1}^{q} N_i = \{1, 2, \ldots, n\}$ e também $N_i \cap N_j = \emptyset$ para todo $i, j = 1, \ldots, q$ e $i \neq j$, isto é, as classes não tem itens comuns. Para cada classe N_k dos índices dos itens, tem-se associado um único conjunto V_k que é composto dos índices dos compartimentos gerados por itens de índices em N_k. Exige-se que $V_j \cap V_k = \emptyset$ para todo $j, k = 1, \ldots, q$ e $j \neq k$ e que cada compartimento com índice em V_k deve cumprir a restrição do limite inferior L^k_{min} e superior L^k_{max} da capacidade permitida para sua geração.

A formulação matemática do problema da mochila compartimentada (Hoto, 2001) é:

$$\textbf{maximizar:} \quad \sum_{k=1}^{q} \sum_{j \in V_1} ((\sum_{i \in N_1} u_i a_{ij}) - c_j) y_j \tag{2.1}$$

$$\textbf{sujeito a:} \quad \sum_{k=1}^{q} \sum_{j \in V_1} (\sum_{i \in N_1} l_i a_{ij}) y_j \leq L \tag{2.2}$$

$$L^k_{min} \leq \sum_{i \in N_k} l_i a_{ij} \leq L^k_{max}, \quad j \in V_k \text{ e } k = 1, \ldots, q \tag{2.3}$$

$$a_{ij}, y_j \geq 0, \text{ inteiros} \tag{2.4}$$

$$i \in N = N_1 \cup \ldots \cup N_q, j \in V = V_1 \cup \ldots \cup V_q$$

Na formulação do problema acima

a_{ij}	é a quantidade de itens do tipo (índice) i no compartimento j
c_j	é o custo por construir o compartimento j
y_j	é a quantidade de compartimentos do tipo (índice) j
L^k_{min}	é a cap. mínima permitida ao compartimento pertecente a V_k
L^k_{max}	é a cap. máxima permitida ao compartimento pertecente a V_k

A capacidade do compartimento j relativo a classe de índice k é

$$L^k_j = \sum_{i \in N_k} l_i a_{ij} \tag{2.5}$$

e a utilidade é

$$U^k_j = \sum_{i \in N_k} u_i a_{ij} \tag{2.6}$$

Pela formulação matemática, o problema da mochila compartimentada é um problema não-linear nas variáveis a_{ij} e y_j, sendo que a não-linearidade do problema da mochila compartimentada encontra-se tanto na função objetivo quanto na restrição de capacidade da mochila. Por estas características, o problema adquire uma dificuldade maior de resolução.

Em (Marques, 2004), há a inclusão no problema da mochila compartimentada a questão de perda da capacidade da mochila ao se incluir um compartimento, obtendo-se uma pequena modificação na restrição de capacidade. Abaixo, encontra-se esta modificação no problema:

$$\textbf{maximizar:} \quad \sum_{k=1}^{q}\sum_{j=1}^{|V_k|}(U_j^k - c_k)y_{jk} \tag{2.7}$$

$$\textbf{sujeito a:} \quad U_j^k = \sum_{i\in N_k} u_i a_{ij}^k, \tag{2.8}$$
$$j = 1,\ldots,|V_k| \text{ e } k = 1,\ldots,q$$

$$L_j^k = \sum_{i\in N_k} l_i a_{ij}^k + S_k, \tag{2.9}$$
$$j = 1,\ldots,|V_k| \text{ e } k = 1,\ldots,q$$

$$L_{min}^k \leq L_j^k \leq L_{max}^k, \tag{2.10}$$
$$j = 1,\ldots,|V_k| \text{ e } k = 1,\ldots,q$$

$$\sum_{j=1}^{|V_k|} a_{ij}^k y_{jk} \leq d_i, \tag{2.11}$$
$$i \in N_k \text{ e } k = 1,\ldots,q$$

$$\sum_{k=1}^{q}\sum_{j=1}^{|V_k|} L_j^k y_{jk} \leq L \tag{2.12}$$

$$a_{ij}^k, y_{jk} \geq 0, \text{ inteiros} \tag{2.13}$$
$$i \in N = N_1 \cup \ldots \cup N_q,$$
$$j \in V = V_1 \cup \ldots \cup V_q \text{ e } k = 1,\ldots,q$$

Na formulação apresentada por Marques, U_j^k e L_j^k são a utilidade e a capacidade do compartimento j associado a classe N_k, respectivamente, e S_k é a perda da capacidade da mochila por incluir um novo compartimento criado a partir de itens da classe N_k. A variável y_{jk} representa a quantidade de compartimentos j associado a classe N_k que pode ser inserida na mochila. A variável a_{ij}^k é a quantidade de itens do tipo $i \in N_k$ que irá compor o compartimento j.

Exemplo: Nas tabelas 2.1, 2.2, 2.3, 2.4, 2.5, 2.6 e 2.7 abaixo, encontram-se as classes com seus respectivos itens, onde cada item tem suas características e disponibilidades (quantidades) e os compartimentos com suas respectivas características.

Tabela 2.1 – Dados dos itens da classe 1

Classe 1: $L^1_{min} = 5$, $L^1_{max} = 15$, $S_1 = 0.2$ e $c_1 = 1$			
itens	utilidade	peso	quantidade
3	8	3	3
4	5	3	2
6	8	6	3

Tabela 2.2 – Dados dos compartilhamentos associados a classe 1

Compartimentos associados a Classe 1			
compartimentos	utilidade	peso	construção
1	16	6	$l_3 + l_3$
2	13	6	$l_3 + l_4$
3	10	6	$l_4 + l_4$
4	8	6	l_6
5	24	9	$l_3 + l_3 + l_3$
6	21	9	$l_3 + l_3 + l_4$
7	18	9	$l_3 + l_4 + l_4$
8	16	9	$l_3 + l_6$
9	13	9	$l_4 + l_6$
10	29	12	$l_3 + l_3 + l_3 + l_4$
11	26	12	$l_3 + l_3 + l_4 + l_4$
12	24	12	$l_3 + l_3 + l_6$
13	23	12	$l_3 + l_4 + l_6$
14	18	12	$l_4 + l_4 + l_6$
15	16	12	$l_6 + l_6$
16	34	15	$l_3 + l_3 + l_3 + l_4 + l_4$
17	32	15	$l_3 + l_3 + l_3 + l_6$

Tabela 2.3 – Dados dos com partilhamentos associados a c lasse 1 *(continuação)*

Compartimentos associados a Classe 1			
compartimentos	utilidade	peso	construção
18	29	15	$l_3 + l_3 + l_4 + l_6$
19	26	15	$l_3 + l_4 + l_4 + l_6$
20	24	15	$l_3 + l_6 + l_6$
21	21	15	$l_4 + l_6 + l_6$

Tabela 2.4 – Dados dos itens da classe 2

Classe 2: $L^2_{min} = 6$, $L^2_{max} = 12$, $S_2 = 0.2$ e $c_2 = 2$			
itens	utilidade	peso	quantidade
1	4	5	2
2	5	4	2
5	6	6	2

Tabela 2.5 – Dados dos compartilhamentos associado a classe 2

Compartimentos associados ao Classe 2			
compartimentos	utilidade	peso	construção
22	6	6	l_5
23	10	8	$l_2 + l_2$
24	9	9	$l_2 + l_1$
25	11	10	$l_2 + l_5$
26	8	10	$l_1 + l_1$
27	10	11	$l_1 + l_5$
28	12	12	$l_5 + l_5$

Tabela 2.6 – Dados dos itens da classe 3

Classe 3: $L^3_{min} = 4$, $L^3_{max} = 10$, $S_3 = 0.2$ e $c_3 = 4$			
itens	utilidade	peso	quantidade
7	3	3	2
8	3	2	2

Tabela 2.7 – Dados dos compartilhamentos associado a classe 3

Compartimentos associados a Classe 3			
compartimentos	utilidade	peso	construção
29	6	4	$l_8 + l_8$
30	6	5	$l_7 + l_8$
31	6	6	$l_7 + l_7$
32	9	7	$l_7 + l_8 + l_8$
33	9	8	$l_7 + l_7 + l_8$
34	12	10	$l_7 + l_7 + l_8 + l_8$

Por este exemplo pode-se ver o poder combinatório do problema, pois na classe 1 utiliza-se apenas 3 itens, e obtém-se 21 compartimentos respeitando as restrições impostas. De posse destes 21 compartimentos, verifica-se qual retorna o maior valor na função objetivo. Isto mostra que, se houver uma quantidade muito grande de classes e itens, pode-se obter uma quantidade enorme de compartimentos, tornando o problema difícil de resolver.

Na próxima seção, encontra-se o problema da mochila compartimentada restrita, que em sua formulação, tem os conceitos do problema da mochila compartimentada, acrescido de outras restrições (necessárias em problemas atuais como corte de bobinas de aço, sujeito à laminação), de forma que este problema adquire um maior grau de dificuldade na resolução.

2.2 Problema da Mochila Compartimentada Restrita

Nesta seção, faz-se o acréscimo de restrições ao problema da mochila compartimentada, porém, antes mostra-se a solução ótima para o problema da mochila compartimentada, tendo como base os dados e o modelo (2.7 - 2.13) da seção anterior, onde se considera $c_k = 0$ e $S_k = 0$. A tabela (2.8) e a figura (2.1) abaixo expressam a solução ótima, sendo a linha mais forte na figura a separação dos compartimentos e a outra, a separação dos itens que os compõem.

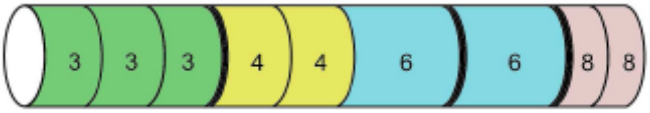

Figura 2.1 – Disposição da solução ótima

Ao problema proposto na seção anterior, os acréscimos serão as seguintes restrições:

- quantidade máxima de itens do tipo i que podem compor a mochila ou disponíveis em estoque (já fazia parte da formulação proposta por (Marques, 2004)) que é representada pela restrição (2.14) abaixo

Tabela 2.8 – Uma solução para o problema

compartimentos	utilidade	capacidade	construção
classe 1 compartimento 5	24	9	$l_3 + l_3 + l_3$
classe 1 compartimento 14	18	12	$l_4 + l_4 + l_6$
classe 1 compartimento 4	6	4	l_6
classe 3 compartimento 29	8	6	$l_8 + l_8$

$$\sum_{j \in V_k} a_{ij} y_j \leq d_i, \ i \in N_k \text{ e } k = 1, \ldots, q \tag{2.14}$$

- quantidade máxima de compartimentos que podem ser criados na mochila, expressa por

$$\sum_{k=1}^{q} \sum_{j \in V_k} y_j \leq F_1 \tag{2.15}$$

onde $F_1 \in \mathbb{Z}_+^*$.

- quantidade máxima de itens que pode conter cada compartimento

$$\sum_{i \in N_k} a_{ij} \leq F_2, \ j \in V = V_1 \cup \ldots \cup V_k, \ k = 1 \ldots q \tag{2.16}$$

onde $F_2 \in \mathbb{Z}_+^*$.

A partir da formulação (2.1 - 2.4) proposta por (Hoto, 2001) e inserindo a perda conforme o problema proposto por (Marques, 2004), tem-se a formulação do **Problema da Mochila Compartimentada Restrita**:

$$\textbf{maximizar:} \quad \sum_{k=1}^{q} \sum_{j \in V_k} ((\sum_{i \in N_k} u_i a_{ij}) - c_j) y_j \tag{2.17}$$

$$\textbf{sujeito a:} \quad \sum_{k=1}^{q} \sum_{j \in V_k} ((\sum_{i \in N_k} l_i a_{ij}) + S_j) y_j \leq L \tag{2.18}$$

$$L^k_{min} \leq (\sum_{i \in N_k} l_i a_{ij}) + S_j \leq L^k_{max}, \tag{2.19}$$

$$j \in V_k \text{ e } k = 1, \ldots, q$$

$$\sum_{j \in V_k} a_{ij} y_j \leq d_i, \tag{2.20}$$

$$i \in N_k \text{ e } k = 1, \ldots, q$$

$$\sum_{k=1}^{q} \sum_{j \in V_k} y_j \leq F_1 \tag{2.21}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2, \tag{2.22}$$

$$j \in V = V_1 \cup \ldots \cup V_k, \; k = 1 \ldots q$$

$$a_{ij}, y_j \geq 0, \text{ inteiros} \tag{2.23}$$

$$i \in N = N_1 \cup \ldots \cup N_q,$$

$$j \in V = V_1 \cup \ldots \cup V_q$$

Os próximos capítulos assumem $c_j = 0$ e $S_j = 0$ (c_j e S_j constantes) na formulação do problema (2.17 - 2.23), pois a dificuldade do problema não é alterada, não sendo necessário a geração destes valores.

O capítulo seguinte trata-se de heurísticas, decompondo o problema (2.17 - 2.23) em outros problemas lineares (pois sabe-se que há um custo muito maior para resolver problemas não-lineares do que lineares e atualmente os vários metódos existentes exploram esta eficiência) como forma de obter soluções viáveis.

CAPÍTULO 3
HEURÍSTICAS

Este capítulo apresenta procedimentos para obter soluções viáveis (soluções ótimas ou próximas da ótima) de forma rápida em relação a métodos exatos como programação dinâmica e métodos de enumeração. Eles baseiam-se numa progressiva aproximação da solução ótima e serão abordadas as seguintes heurísticas: Decomposição, Retro e Wretro.

Estas heurísticas são usadas para resolver o problema da mochila compartimentada restrita em dois estágios, onde se utiliza a decomposição do problema em dois problemas lineares, e desta forma, pode-se aplicar métodos e técnicas de problema da mochila restrita e programação linear inteira aos estágios. Das heurísticas citadas acima, Retro e Wretro são chamadas de heurísticas de retro-alimentação, por fazer uso da solução do segundo estágio no próprio estágio várias vezes.

3.1 Heurística de Decomposição

Esta heurística foi escolhida para compor o trabalho, sendo abordada em (Hoto, Spolador e Marques, 2005), pois retorna soluções melhores em relação as outras heurísticas abordadas no mesmo artigo. A heurística de Decomposição, decompõe (divide) o problema da mochila compartimentada restrita em dois problemas lineares, onde o primeiro é um problema da mochila restrita e o outro, um problema de programação inteira com uma imensa quantidade de "itens".

No primeiro estágio da heurística, gera-se a melhor combinação dos itens para cada compartimento de capacidade representada por $L_s \in \{L^k_{min}, L^k_{min}+1, \ldots, L^k_{max}\}$, associado à classe de índice k. Estas melhores combinações dos itens que pertencem à classe de índice k, são encontradas pela resolução do seguinte problema da mochila:

$$\textbf{maximizar:} \quad U_s = \sum_{i \in N_k} u_i a_{is} \tag{3.1}$$

$$\textbf{sujeito a:} \quad \sum_{i \in N_k} l_i a_{is} = L_s \tag{3.2}$$

$$0 \leq a_{is} \leq d_i, \quad i \in N_k \tag{3.3}$$

$$\sum_{i \in N_k} a_{is} \leq F_2 \tag{3.4}$$

$$a_{is} \in \mathbb{Z}_+ \text{ e } i \in N_k \tag{3.5}$$

O segundo estágio da heurística de Decomposição utiliza os dados U_j, L_j e a_{ij} da primeira fase. Admite que cada compartimento será visto como um "super-item"[1], e a solução viável do problema da mochila compartimentada é dado pelo problema de programação inteira abaixo, que oferece a melhor combinação dos compartimentos gerados no primeiro estágio.

$$\textbf{maximizar:} \quad \sum_{j \in V_1} U_j y_j + \ldots + \sum_{j \in V_q} U_j y_j \tag{3.6}$$

$$\textbf{sujeito a:} \quad \sum_{j \in V_1} L_j y_j + \ldots + \sum_{j \in V_1} L_j y_j \leq L \tag{3.7}$$

$$\sum_{j \in V_1} a_{ij} y_j + \ldots + \sum_{j \in V_q} a_{ij} y_j \leq d_i \tag{3.8}$$

$$i \in N = N_1 \cup \ldots \cup N_q$$

$$\sum_{j \in V_1} y_j + \ldots + \sum_{j \in V_1} y_j \leq F1 \tag{3.9}$$

$$y_j \geq 0, \text{ inteiros}, j \in V = V_1 \cup \ldots \cup V_q \tag{3.10}$$

No problema acima, a_{ij} não são variáveis, sendo apenas valores resultantes da resolução do problema (3.1 - 3.5). Observe que o problema depende apenas de y_j, e com isso, obtém-se um problema linear.

Abaixo, encontram-se o algoritmo da heurística de Decomposição e o algoritmo eficiente de encontrar estes compartimentos viáveis é visto em (Christofides & Whitlock, 1977).

[1]Será definido super-item, como sendo o compartimento a unidade básica em relação a mochila, não sendo definido item.

Algoritmo 3.1: Heurística de Decomposição

Entrada: u_i, l_i, d_i, N_k, L^k_{min}, L^k_{max}, F_1, F_2, q
Saída : a_{ij}, U_j, L_j e compartimentação da mochila
Inicialização: $\Omega = \emptyset$;
Para todo $k = 1, \ldots, q$ **faça**
    Para cada compartimento viável
    $L_j \in \{L^k_{min}, L^k_{min} + 1, L^k_{min} + 2, \ldots, L^k_{max}\}$, com $j \in V_k$;
    Resolva a mochila dada pelo problema (3.1 - 3.5)
Para todo $j \in V$ **faça**
    Resolva a compartimentação resolvendo o problema (3.6 - 3.10)
Retorno *Compartimentação mais vantajosa da mochila*;

Algoritmo 3.2: Construção de Compartimentos Viáveis a N_k

Entrada: N_k, L^k_{min}, L^k_{max}, F_2, l_i
Saída : V'_k associados a classe N_k
Inicialização: $V'_k = \{l_i : i \in N_k\}$;
Para todo $j = \min N_k, \ldots, L^k_{max}$ **faça**
    Para todo $i \in N_k$ **faça**
        Se $j - l_i \in V'_k$ **então**
            Insira j em V'_k;
            break;
Para todo $j \in V'_k$ **faça**
    Se $j < L^k_{min}$ **então**
        Retire j de V'_k;
Retorno *Compartimentos viáveis associados a* N_k;

No algoritmo (3.2) acima, V'_k representa o conjunto com os "pesos"dos compartimentos viáveis associado a classe de índice k.

Veja agora o diagrama da heurística de Decomposição:

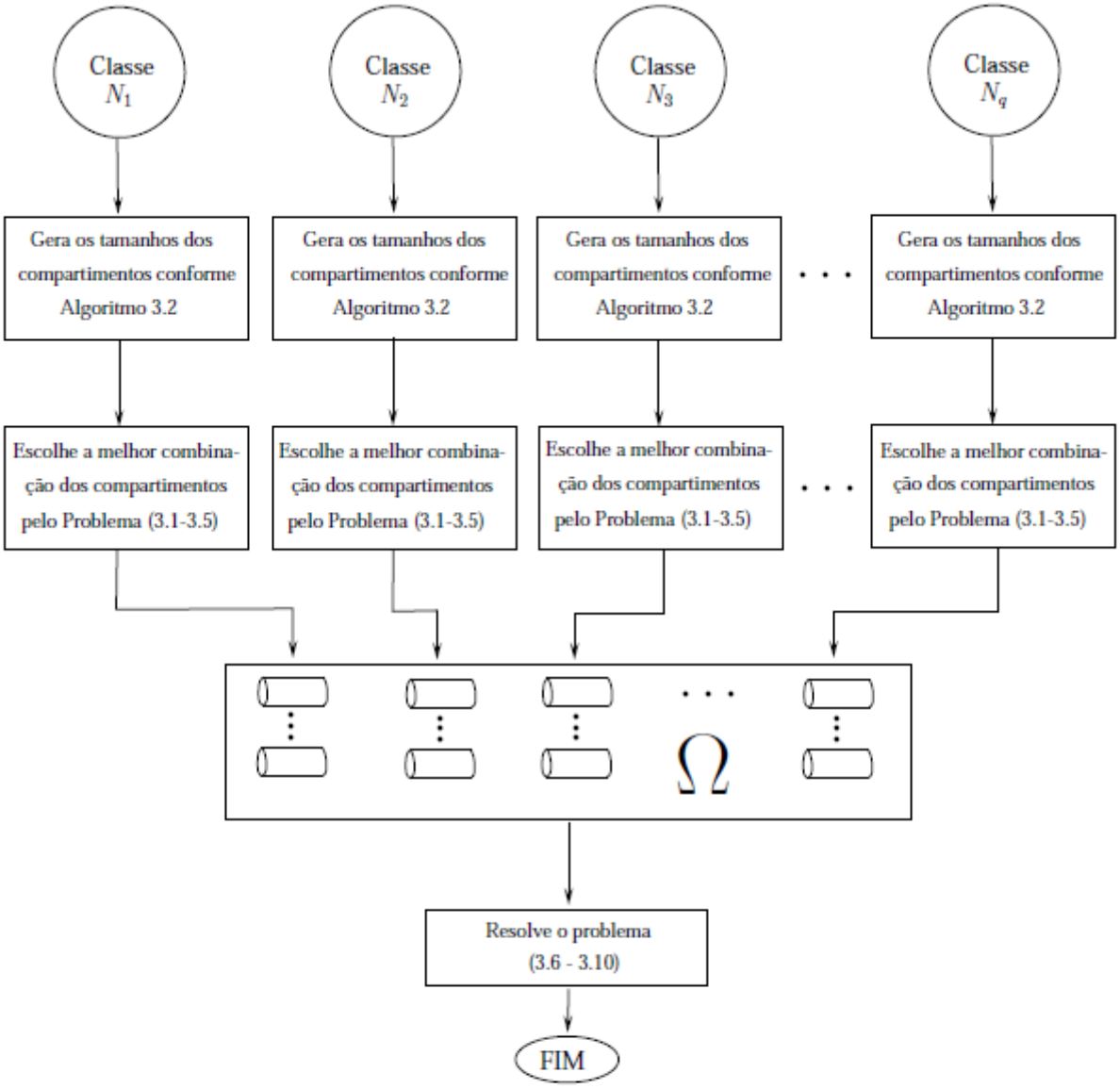

Diagrama da Heurística de Decomposição

No diagrama acima, $\Omega = \bigcup_{k=1}^{q} V_k$, isto é, Ω representa a união das classes dos compartimentos gerados através dos itens das respectivas classes.

A próxima seção apresenta outra forma de decompor o problema da mochila compartimentada restrita em problemas lineares, sendo que no segundo estágio da próxima heurística, a obtenção da melhor compartimentação se dá pela resolução do passo anterior (realimentação).

A heurística de Decomposição, vista anteriormente, retorna uma solução viável, pois esta gera no primeiro estágio apenas a melhor combinação para os compartimentos de tamanhos $L^k_{min}, L^k_{min}+1, L^k_{min}+2, \ldots, L^k_{max}$, sendo estes associados a itens da classe k. Como forma de obter um resultado também satisfatório, sem necessitar de construir todas as melhores combinações dos itens para os compartimentos de tamanhos $L^k_{min}, L^k_{min}+1, L^k_{min}+2, \ldots, L^k_{max}$, gera-se apenas o melhor compartimento associado aos itens da classe k. A heurística Retro (como também a Wretro, que será apresentada na próxima seção) abordada nos trabalhos (Spolador, 2005) e (Hoto *et al.*, 2006), foi escolhida por retornar uma solução viável que pode ser a solução ótima ou não, sendo que esta solução é apenas inferior as soluções da heurística Wreto e da abordagem linear. Esta heurística decompõe também o problema da mochila compartimentada restrita em dois outros problemas da mochila, sendo esses problemas lineares, de forma a facilitar a obtenção da solução (ótima ou viável).

O primeiro estágio desta heurística resolve q mochilas, associadas as q classes, conforme formulação (3.12 - 3.17) abaixo, e para cada $k = 1, \ldots, q$, insere no conjunto Ω (conjunto que armazena os compartimentos criados para serem usados no segundo estágio da heurística) estes compartimentos gerados no primeiro estágio e também no segundo. Assim, para cada problema da mochila tem-se somente um compartimento associado à classe de índice k neste primeiro estágio.

Nesta heurística, define δ_j como

$$\begin{array}{rcl} \delta_j = 0 & \Leftrightarrow & y_j = 0 \\ \delta_j = 1 & \Leftrightarrow & y_j > 0 \end{array} \tag{3.11}$$

para obter a consistência da equação 2.19, pois no trabalho (Hoto *et al.*, 2006), foi concebido o uso de δ_j no problema da Mochila Compartimentada Restrita, porém, este não foi elaborado na heurística Retro e WRetro. Abaixo encontra-se a formulação do problema do primeiro estágio para a classe k:

$$\textbf{maximizar:} \quad U_j = \sum_{i \in N_k} u_i a_i \tag{3.12}$$

$$\textbf{sujeito a:} \quad L_j = \sum_{i \in N_k} l_i a_i \tag{3.13}$$

$$\delta_j L^k_{min} \leq L_j \leq \delta_j L^k_{max} \tag{3.14}$$

$$0 \leq a_i \leq d_i, \quad i \in N_k \tag{3.15}$$

$$\sum_{i \in N_k} a_i \leq F_2 \tag{3.16}$$

$$a_i \in \mathbb{Z}_+, \ \delta_j \in \{0,1\} \text{ e } i \in N_k \tag{3.17}$$

O segundo estágio da heurística é responsável por obter a melhor combinação dos compartimentos em Ω (gerados no primeiro estágio, e posteriormente, pelo segundo), além de verificar a possível inclusão de mais m compartimentos associados à classe de índice k.

Sabe-se que para a classe de índice k pode-se criar compartimentos com capacidades entre L^k_{mim} e L^k_{max}, além da possibilidade de inserir no máximo

$$p_k = \min \left\{ F_1, \left\lfloor \frac{L}{L^k_{min}} \right\rfloor \right\} \tag{3.18}$$

compartimentos associados à classe de índice k na mochila. Na expressão (3.18), foi usado o mínimo entre a quantidade máxima de compartimentos permitidos na mochila e o piso[2] da quantidade máxima de compartimentos com capacidade mínima que podemos colocar sem a restrição na mochila.

Como a expressão (3.18) obedece a restrição do problema da mochila compartimentada (problema (2.17 - 2.23)), o próximo procedimento é resolver o problema para todas as classes, isto é, $k = 1, \ldots, q$, sendo que o compartimento gerado será inserido em Ω, obedecendo a quantidade máxima de vezes que um compartimento poderá ser usado, sendo possível, $m = 1$, ou 2, ..., ou p_k vezes. Assim, resolve-se o problema (3.19 - 3.26) abaixo, e a cada solução do segundo estágio, há a inserção deste compartimento em Ω. Veja a formulação:

[2] $\lfloor x \rfloor$ é a função que retorna o maior inteiro menor ou igual a x.

$$\textbf{maximizar:} \quad z = \sum_{j \in \Omega} U_j y_j + m \sum_{i \in N_k} u_i \alpha_i \tag{3.19}$$

$$\textbf{sujeito a:} \quad \sum_{j \in \Omega} L_j y_j + m \sum_{i \in N_k} l_i \alpha_i \leq L \tag{3.20}$$

$$\delta_j L^k_{min} \leq \sum_{i \in N_k} l_i \alpha_i \leq \delta_j L^k_{max} \tag{3.21}$$

$$\sum_{j \in \Omega} a_{ij} y_j \leq d_i, \quad i \in N - N_k \tag{3.22}$$

$$\sum_{j \in \Omega} a_{ij} y_j + m\alpha_i \leq d_i, \quad i \in N_k \tag{3.23}$$

$$(\sum_{j \in \Omega} y_j) + m \leq F_1 \tag{3.24}$$

$$\sum_{i \in N_k} \alpha_i \leq F_2 \tag{3.25}$$

$$\alpha_i, y_j \in \mathbb{Z}_+, \ \delta_j \in \{0, 1\}, \ i \in N \text{ e } j \in \Omega. \tag{3.26}$$

Na formulação proposta nos trabalhos (Spolador, 2005) e (Hoto *et al.*, 2006), a restrição de disponibilidade de itens na mochila, no segundo estágio das heurísticas Retro e WRetro é

$$\sum_{j \in \Omega} a_{ij} y_j \leq d_i, \quad i \in N \tag{3.27}$$

A proposta deste trabalho é decompor a restrição acima (3.27) nas duas restrições

$$\sum_{j \in \Omega} a_{ij} y_j \leq d_i, \quad i \in N - N_k \tag{3.28}$$

$$\sum_{j \in \Omega} a_{ij} y_j + m\alpha_i \leq d_i, \quad i \in N_k \tag{3.29}$$

como forma de satisfazer a função objetivo (3.19) e a restrição de capacidade (3.20) da mochila.

Os problemas nos dois estágios são lineares e podem ser resolvidos por métodos para problema da mochila e programação inteira. Veja o pseudocódigo da heurística:

Algoritmo 3.3: Heurística Retro

Entrada: u_i, l_i, d_i, N_k, L^k_{min}, L^k_{max}, F_1, F_2, q
Saída : a_{ij}, U_j, L_j e compartimentação da mochila
Inicialização: $\Omega = \emptyset$;
Para todo $k = 1, \ldots, q$ **faça**
  Construa o compartimento resolvendo o problema (3.12 - 3.17);
  Insira o compartimento gerado em Ω;
Para todo $k = 1, \ldots, q$ **faça**
  Faça $p_k = \min\{F_1, \lfloor L/L^k_{min} \rfloor\}$;
  Para todo $m = 1, \ldots, p_k$ **faça**
    Construa o compartimento resolvendo o problema (3.19 - 3.26);
  Insira o compartimento mais vantajoso dentre os p_k em Ω;
Retorno *Compartimentação mais vantajosa da mochila*;

Na próxima seção há uma modificação capaz de aumentar a possibilidade de termos a solução ótima, aumentando a quantidade de compartimentos em Ω, porém, aumenta-se o custo computacional pela necessidade de maior combinação de itens e "superitens".

Abaixo tem-se o diagrama da heurística Retro:

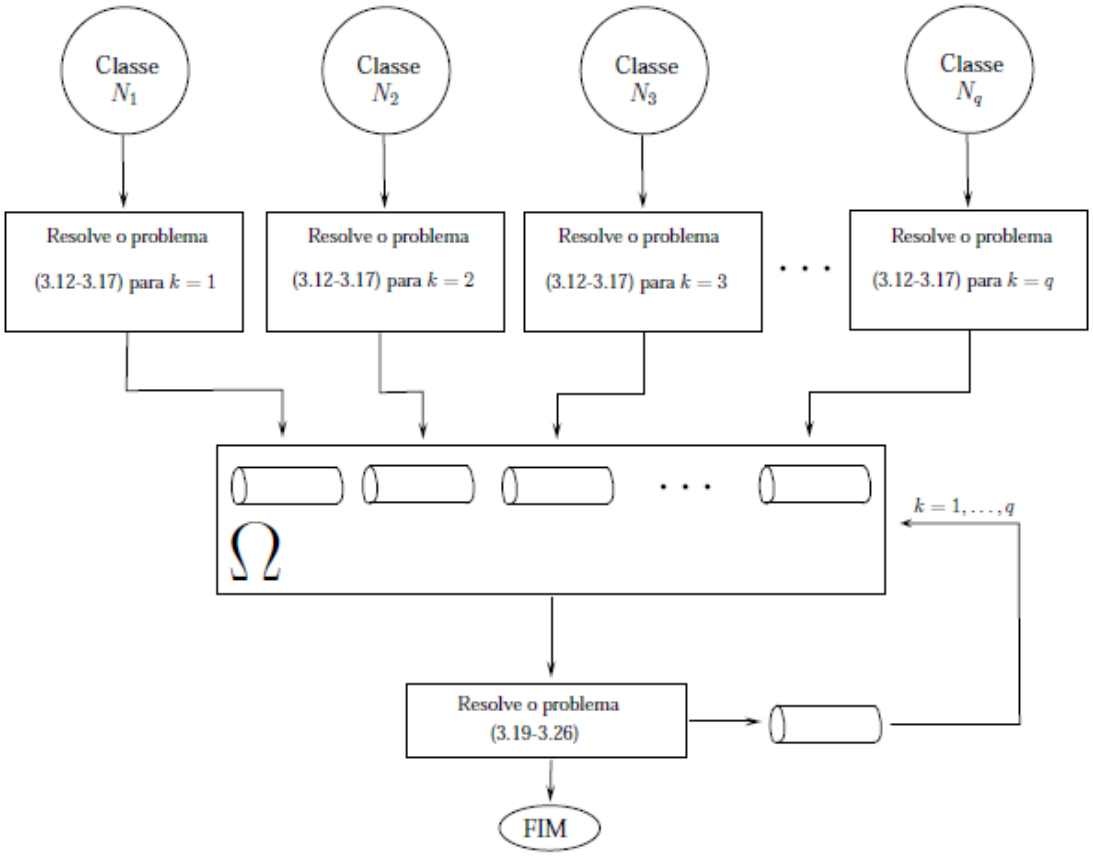

Diagrama da Heurística Retro

A próxima seção é caracterizada pela modificação no primeiro estágio da heurística Retro, afim de obter uma quantidade maior de compartimentos no primeiro estágio.

3.3 HEURÍSTICA WRETRO

Esta heurística abordada em (Hoto *et al.*, 2006), associa a cada classe k até w compartimentos (diferentemente da heurística Retro que associa apenas um compartimento por classe), estes construídos pela resolução do problema da mochila abaixo:

$$\text{maximizar:} U_j = \sum_{i \in N_k} u_i a_i \tag{3.30}$$

$$\text{sujeito a:} L_j = \sum_{i \in N_k} l_i a_i \tag{3.31}$$

$$L^k_{min} \leq L_j \leq L^k_{comp} \tag{3.32}$$

$$0 \leq a_i \leq d_i, \quad i \in N_k \tag{3.33}$$

$$\sum_{i \in N_k} a_i \leq F_2 \tag{3.34}$$

$$a_i \in \mathbb{Z}_+ \text{ e } i \in N_k \tag{3.35}$$

onde L^k_{comp} é atualizado pela expressão

$$L^k_{comp} = \begin{cases} L^k_{max} & \text{para } j = 1 \text{ e } k = 1, \ldots, q \\ (\sum_{i \in N_k} l_i a^*_i) - 1 & \text{para } j = 2, \ldots, w \text{ e } k = 1, \ldots, q \end{cases} \tag{3.36}$$

sendo que a^*_i, onde $i \in N_k$, é a solução do problema do compartimento anterior.

O segundo estágio da heurística Wretro é igual ao do segundo estágio da heurística Retro, contudo, a melhoria na heurística Wretro foi obtida pelo maior número de "superitens"no primeiro estágio, possibilitando uma maior combinação destes. Porém, com a possibilidade de melhorar a solução com este acréscimo de compartimentos, tem-se um aumento na quantidade das combinações, obtendo com isso um maior tempo em relação a heurística Retro na obtenção da solução.

A heurística Retro é um caso particular da heurística Wretro, pois se fizer $w = 1$ na heurística Wretro, obtém-se a heurística Retro, e com esta afirmação pode-se concluir que a heurística Wretro exibe resultados melhores em relação a heurística Retro.

Abaixo, encontra-se o algoritmo da heurística Wretro:

Algoritmo 3.4: Heurística WRetro

Entrada: u_i, l_i, d_i, N_k, L^k_{min}, L^k_{max},, F_1, F_2, q e w
Saída : a_{ij}, U_j, L_j e compartimentação da mochila
Inicialização: $\Omega = \emptyset$;
Para todo $k = 1, \ldots, q$ **faça**
  $L^k_{comp} = L^k_{max}$;
  Para todo j=1,. . . ,w **faça**
    Construa o compartimento resolvendo o problema (3.30 - 3.35);
    Insira o compartimento gerado em Ω;
    `Seja` a^*_i `a solução do problema (3.30 - 3.35);`
    Atualize $L^k_{comp} = \left(\sum_{i \in N_k} l_i a^*_i\right) - 1$;
Para todo $k = 1, \ldots, q$ **faça**
  Faça $p_k = \min\{F_1, \lfloor L/L^k_{min} \rfloor\}$;
  Para todo $m = 1, \ldots, p_k$ **faça**
    Resolva o problema (3.19 - 3.26);
  Insira o compartimento mais vantajoso dentre os p_k em Ω;
Retorno *Compartimentação mais vantajosa da mochila*;

A seguir pode-se ver o diagrama da heurística Wretro:

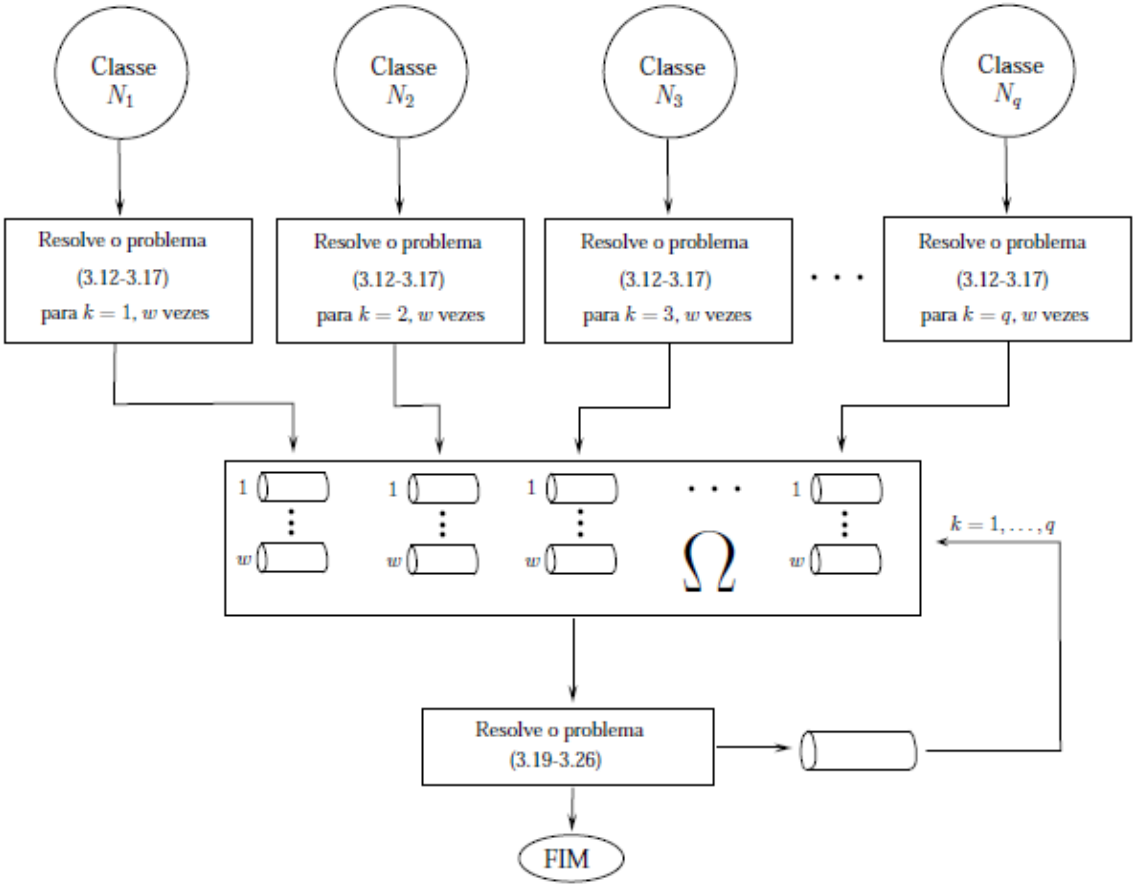

Diagrama da Heurística WRetro

Todas estas heurísticas são opções de resolução do problema da mochila compartimentada restrita (formulação não-linear), e estas fornecem uma solução razoável, resolvendo o problema em dois estágios, sendo cada estágio, vários problema da mochila e um problema de programação inteira, respectivamente. Assim, pode-se aplicar métodos para problemas de mochila restrita, como pode-se ver em (Martello & Toth, 1997), e métodos de resolução de problemas de programação inteira.

No próximo capítulo abordar-se-á o assunto que rege este texto, o processo de "transformar"o problema da mochila compartimentada restrita num problema linear, e a partir disso, obter a solução.

CAPÍTULO 4
UMA ABORDAGEM LINEAR AO PROBLEMA DA MOCHILA COMPARTIMENTADA RESTRITA

O problema da mochila compartimentada, quanto também o problema restrito, tem em sua formulação duas variáveis, a_{ij} e y_j, que definem os problemas como não-lineares, porque estas se multiplicam tanto na função objetivo quanto na restrição de capacidade da mochila. Mas há a possibilidade de fazer modificação na variável a_{ij} de tal forma a "linearizar"o poblema da mochila compartimentada restrita, e isto foi proposto em (Spolador, 2005) e (Hoto *et al.*, 2006). Abaixo encontra-se o processo (heurística) de transformar o problema da mochila compartimentada restrita em problema da mochila compartimentada restrita linear (abordagem linear).

4.1 Definição

Seja $N = \{1, \ldots, n\}$ o conjunto dos índices dos itens e q a quantidade de partições do conjunto, onde $N = N_1 \cup N_2 \cup \ldots \cup N_q$, $N_s \cap N_t = \emptyset$ se $s \neq t$.

Para cada partição N_k associa um único conjunto V_k, que será o conjunto dos índices dos compartimentos gerados pelos itens com índice em N_k.

Tome por exemplo, $N = \{1, 2, 3, \ldots, 10\}$, $N_1 = \{1, 2, 3, 4\}$, $N_2 = \{5, 6, 7\}$ e $N_3 = \{8, 9, 10\}$, e os possíveis conjuntos dos compartimentos são: $V_1 = \{1, 2, 3, \ldots, 8\}$, $V_2 = \{9, 10, \ldots, 17\}$ e $V_3 = \{18, 19, \ldots, 32\}$. Abaixo encontra-se a figura (4.1) que representa a visualização global[1], usado no problema da mochila compartimentada restrita.

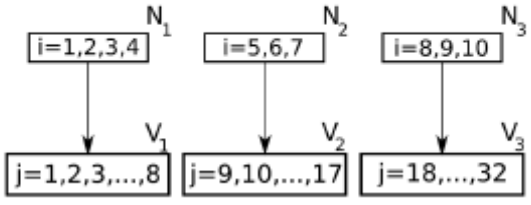

Figura 4.1 – Indexação global dos compartimentos

[1] A indexação global é definido pela quantidade de compartimentos em $V = \bigcup_{i=1}^{k} V_i$

Para se obter a quantidade de itens de índice i em um compartimento de índice j, utiliza-se as seguintes variáveis a_{ij} e y_j, de forma que o resultado é obtido através da multiplicação de ambas.

Note que na variável y o índice da classe não foi indexado, tendo apenas índice j. A idéia será colocar na variável a a respectiva partição onde o item de índice i pertence e o compartimento associado, admitindo que a multiplicidade y_j será decomposta em "novos"compartimentos com multiplicidade 1.

A nova variável será a_{ij}^k, que representa a quantidade de itens de índice i no compartimento j da classe N_k. Como cada compartimento j pode ser inserido uma quantidade y_j de vezes na mochila, onde a expressão $a_{ij}y_j$ representa a quantidade de itens de índice i no compartimento j, porém, sem o devido conhecimento ao qual classe pertence. Desta forma, propõe-se com esta modificação, tornar o problema de indexação global dos índices dos compartimentos em uma indexação local[2]. Observe a modificação na indexação, visualizado na figura abaixo:

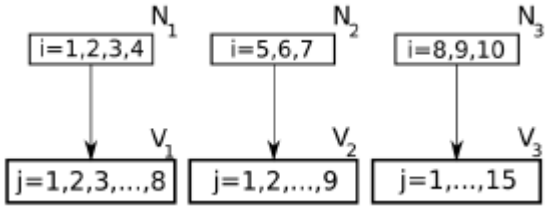

Figura 4.2 – Indexação local dos compartimentos

A necessidade da mudança do endereçamento global para local, tem por objetivo escrever os y_j compartimentos como sendo compartimentos "distintos"associados a mesmo classe.

Para facilitar a compreensão, suponha que há duas classes: $N_1 = \{1, 2, 3\}$ e $N_2 = \{4, 5, 6, 7\}$, onde cada classe gere os conjuntos $V_1 = \{1, 2, 3, 4\}$ e $V_2 = \{5, 6, 7, 8, 9\}$ dos compartimentos associados a cada classe, respectivamente, e suponha que a figura (4.3) abaixo seja a solução ótima do problema.

[2] A indexação local é definido pela quantidade de compartimentos em cada V_k

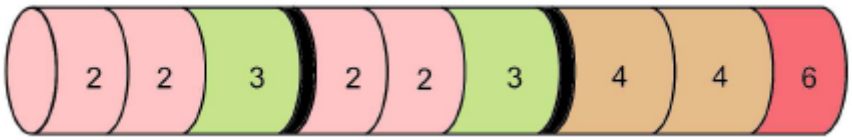

Figura 4.3 – Disposição dos compartimentos e seus respectivos itens

Exibe-se a solução, tendo como base o problema da mochila compartimentada restrita não-linear, como:

$$\left.\begin{array}{ll} a_{21} & =2 \\ a_{31} & =1 \end{array}\right\} y_1 = 2 \quad \Big| \quad \left.\begin{array}{ll} a_{45} & =2 \\ a_{65} & =1 \end{array}\right\} y_5 = 1$$

Agora, faz-se uso da idéia de indexação local, gerando novos compartimentos com a multiplicidade 1, e obtém-se

$$\left.\begin{array}{ll} a^1_{21} & =2 \\ a^1_{31} & =1 \end{array}\right\} y_1 = 1 \quad \Big| \quad \left.\begin{array}{ll} a^1_{22} & =2 \\ a^1_{32} & =1 \end{array}\right\} y_{1'} = 1 \quad \Big| \quad \left.\begin{array}{ll} a^2_{41} & =2 \\ a^2_{61} & =1 \end{array}\right\} y_5 = 1$$

não havendo a necessidade de exibição dos $y_j = 1$, pois implicitamente, os a^k_{ij} são responsáveis por isto, e desta forma, pode-se omitir y no problema, afim de obter a "linearização"do problema da mochila.

Assuma o problema da Mochila Compartimentada Restrita[3] (4.1-4.7), conforme abaixo

$$\textbf{maximizar:} \quad \sum_{k=1}^{q} \sum_{j \in V_k} \Big(\sum_{i \in N_k} u_i a_{ij} \Big) y_j \tag{4.1}$$

$$\textbf{sujeito a:} \quad \sum_{k=1}^{q} \sum_{j \in V_k} \Big(\sum_{i \in N_k} l_i a_{ij} \Big) y_j \leq L \tag{4.2}$$

$$\delta_j L^k_{min} \leq \sum_{i \in N_k} l_i a_{ij} \leq \delta_j L^k_{max}, \quad j \in V_k \text{ e } k = 1, \ldots, q \tag{4.3}$$

$$\sum_{j \in V_k} a_{ij} y_j \leq d_i, \quad i \in N_k \text{ e } k = 1, \ldots, q \tag{4.4}$$

$$\sum_{k=1}^{q} \sum_{j \in V_k} y_j \leq F_1 \tag{4.5}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2, \quad j \in V = V_1 \cup \ldots \cup V_q, \; k = 1, \ldots, q \tag{4.6}$$

$$\delta_j \in \{0, 1\} \text{ e } a_{ij}, y_j \in \mathbb{Z}_+, i \in N \text{ e } j \in V \tag{4.7}$$

[3]Conforme (Hoto *et al.*, 2006)

A partir do modelo matemático (4.1 - 4.7), há possibilidade de escolher no máximo p_k compartimentos associados a classe N_k para serem inseridos na mochila, onde

$$p_k = \min \left\{ F_1, \left\lfloor \frac{L}{L^k_{min}} \right\rfloor \right\} \tag{4.8}$$

Com o uso da afirmação acima (como também da expressão (4.8)), pode-se utilizar a função objetivo (4.1) na forma

$$\sum_{k=1}^{q} \sum_{j \in V_k} \sum_{i \in N_k} u_i a_{ij} y_j \tag{4.9}$$

para obter a seguinte função objetivo

$$\sum_{k=1}^{q} \sum_{j=1}^{p_k} \sum_{i \in N_k} u_i a^k_{ij} \tag{4.10}$$

De forma equivalente, tem-se a restrição de capacidade da mochila, dada por

$$\sum_{k=1}^{q} \sum_{j=1}^{p_k} \sum_{i \in N_k} u_i a^k_{ij} \leq L \tag{4.11}$$

Na restrição de capacidade de cada compartimento houve uma pequena modificação, pois há a indexação do índice k em δ

$$\delta^k_j L^k_{min} \leq \sum_{i \in N_k} l_i a^k_{ij} \leq \delta^k_j L^k_{max} \tag{4.12}$$

para $j = 1, \ldots, p_k$ e $k = 1, \ldots, q$.

Já a restrição de quantidade de itens disponíveis ou permitidos na mochila é

$$\sum_{j=1}^{p_k} a^k_{ij} \leq d_i, \quad i \in N_k \text{ e } k = 1, \ldots, q \tag{4.13}$$

Na restrição de quantidade de compartimentos permitidos na mochila, houve a substituição da variável y_j, que era responsável por informar a quantidade de compartimentos de índice j. Porém, para esta nova formulação faz-se uso da variável δ^k_j, que expressa se o compartimento de índice j foi criado ou não através de itens de índice da classe N_k, obtendo a seguinte formulação desta restrição

$$\sum_{k=1}^{q}\sum_{j=1}^{p_k}\delta_j^k \leq F_1 \tag{4.14}$$

A restrição de itens por compartimentos, também é analoga

$$\sum_{i\in N_k} a_{ij}^k \leq F_2 \tag{4.15}$$

Neste modelo linear do problema da mochila compartimentada restrita, tem-se a inclusão de uma restrição que exclui as soluções simétricas

$$\sum_{i\in N_k} l_i a_{ij}^k \geq \sum_{i\in N_k} l_i a_{i(j+1)}^k \tag{4.16}$$

que pode ser vista como

$$\sum_{i\in N_k} \left(l_i a_{ij}^k - l_i a_{i(j+1)}^k\right) \geq 0 \tag{4.17}$$

para todo $j = 1, \ldots, p_k - 1$ e $k = 1, \ldots, q$.

As variáveis agora são somente $\delta_j^k \in \{0, 1\}$ e $a_{ij}^k \in \mathbb{Z}_+$ para todo $i \in N$, $j = 1, \ldots, p_k$ e $k = 1, \ldots, q$.

O problema da mochila compartimentada restrita como modelo linear é

$$\textbf{maximizar:} \quad \sum_{k=1}^{q}\sum_{j=1}^{p_k}\sum_{i\in N_k} u_i a_{ij}^k \tag{4.18}$$

$$\textbf{sujeito a:} \quad \sum_{k=1}^{q}\sum_{j=1}^{p_k}\sum_{i\in N_k} l_i a_{ij}^k \leq L \tag{4.19}$$

$$\delta_j^k L_{min}^k \leq \sum_{i\in N_k} l_i a_{ij}^k \leq \delta_j^k L_{max}^k, \tag{4.20}$$

$$j = 1,\ldots,p_k, \ , k = 1,\ldots,q$$

$$\sum_{j=1}^{p_k} a_{ij}^k \leq d_i, \tag{4.21}$$

$$i \in N_k \text{ e } k = 1,\ldots,q$$

$$\sum_{k=1}^{q}\sum_{j=1}^{p_k} \delta_j^k \leq F_1 \tag{4.22}$$

$$\sum_{i\in N_k} a_{ij}^k \leq F_2, \tag{4.23}$$

$$j = 1,\ldots,p_k, \ k = 1,\ldots,q$$

$$\sum_{i\in N_k} l_i a_{ij}^k \geq \sum_{i\in N_k} l_i a_{i(j+1)}^{k,} \tag{4.24}$$

$$j = 1,\ldots,p_k - 1, \ k = 1,\ldots,q$$

$$\delta_j \in \{0,1\} \text{ e } a_{ij}, y_j \in \mathbb{Z}_+, \tag{4.25}$$

$$i \in N \text{ e } j = 1,\ldots,p_k, \ k = 1,\ldots,q$$

A função objetivo (4.18) e a restrição de capacidade da mochila (4.19) foram modificadas em relação aos trabalhos (Spolador, 2005) e (Hoto *et al.*, 2006), como forma de expressar melhor o problema (4.1 - 4.7).

É interessante expor uma demonstração que mostre a equivalência dos problemas (4.1 - 4.7) e (4.18 - 4.25), isto é, mostrar que a solução ótima de um é solução ótima do outro e vice-versa. A necessidade de exibir este resultado é a garantia de obter a solução do problema (4.1 - 4.7) apenas resolvendo o problema (4.18 - 4.25), que tem apenas características lineares em suas expressões.

A proposição 4.1 abaixo afirma que a solução do problema (4.1 - 4.7) é uma solução viável do problema (4.18 - 4.25).

Proposição 4.1. *Seja $\tilde{a}_{ij}$ e $\tilde{y}_j$ soluções do problema (4.1 - 4.7) onde $i \in N_k$, $j \in V_k$ e $k = 1, \ldots, q$. Uma solução viável do problema (4.18 - 4.25) é dada por*

$$\overline{a}^k_{ij'} = \tilde{a}_{ij} \tag{4.26}$$

onde $j' = j + t$ e $t = 0, \ldots, \tilde{y}_j - 1$.

Demonstração: Fazer-se-á a demonstração sobre as restrições[4]

$$\sum_{k=1}^{q} \sum_{j \in V_k} \left(\sum_{i \in N_k} l_i \tilde{a}_{ij} \right) \tilde{y}_j \leq L \tag{4.27}$$

e

$$\sum_{k=1}^{q} \sum_{j \in V_k} \tilde{y}_j \leq F1 \tag{4.28}$$

Há $\tilde{y}_j$ compartimentos com capacidades $\sum_{i \in N_k} l_i \tilde{a}_{ij}$ e utilidades $\sum_{i \in N_k} u_i \tilde{a}_{ij}$ e podem ser observados como distintos, obtendo com isso

$$a^k_{ij'} = \tilde{a}_{ij} \tag{4.29}$$

onde $j' = j + t$ para $t = 0, \ldots, \tilde{y}_j - 1$.

Nesse caso, a restrição (4.27) pode ser escrita como

$$\sum_{k=1}^{q} \sum_{j' \in V_k} \sum_{i \in N_k} l_i a^k_{ij'} \leq L \tag{4.30}$$

e de todos os $j \in V_k$, a solução é composta apenas de $p_k = \min\{F1, \lfloor L/L^k_{min} \rfloor\}$ compartimentos. Com isso, a restrição é escrita

$$\sum_{k=1}^{q} \sum_{j'=1}^{p_k} \sum_{i \in N_k} l_i a^k_{ij'} \leq L \tag{4.31}$$

Para a restrição (4.28), usa-se

$$\begin{array}{rcl} \delta_{j'} = 0 & \Leftrightarrow & \tilde{y}_j = 0 \\ \delta_{j'} = 1 & \Leftrightarrow & \tilde{y}_j > 0 \end{array} \tag{4.32}$$

onde $j' = j + t$ para $t = 0, \ldots, \tilde{y}_j - 1$, e obtém-se com isso

[4] A demonstração das outras restrições é semelhante.

$$\sum_{k=1}^{q}\sum_{j=1}^{p_k}\delta_{j'}^{k} \leq F_1 \tag{4.33}$$

e assim conclui a demonstração.

No próximo capítulo, há os resultados das simulações baseados nos estudos sobre heurísticas (capítulo 3) e na abordagem linear (capítulo 4).

CAPÍTULO 5
SIMULAÇÕES

Este capítulo abordará os procedimentos para a execução das heurísticas e também do problema da mochila compartimentada na forma linear. Os dados usados para elaboração das tabelas, necessárias para comparação, foram gerados de forma que sejam distribuídos uniformemente, garantindo um ótimo desempenho frente a dados realísticos.

5.1 Introdução

Esta seção destina a descrever a geração dos dados como também as dificuldades na resolução das implementações das heurísticas, afim de tornar claro os resultados gerados pelas tabelas da próxima seção.

Os dados utilizados neste trabalho para as simulações, baseiam-se no trabalho (Belov e Scheithauer, 2002). Porém, foram escolhidos as seguintes composições: (5 classes/5 itens), (5 classes/10 itens), (5 classes/30 itens), (10 classes/5 itens), (10 classes/10 itens) e (10 classes/15 itens), de maneira a comparar com o trabalho (Hoto *et al.*, 2006).

A capacidade máxima da mochila, a capacidade mínima e máxima de cada compartimento são, respectivamente, $L = 100.000$, $L^k_{min} = 10.000$ e $L^k_{max} = 15.000$.

As utilidades (u_i) e pesos (l_i), foram gerados a partir dos seguintes parâmetros:

$$\mu_1 \cdot L^k_{max} \leq u_i \leq \mu_2 \cdot L^k_{max} \tag{5.1}$$

$$\mu_1 \cdot L^k_{max} \leq l_i \leq \mu_2 \cdot L^k_{max} \tag{5.2}$$

distribuidos uniformente onde

$$\mu_1 < \mu_2 \tag{5.3}$$

e

$$\mu_1 = \{0.001, 0.05, 0.15, 0.20\},$$
$$\mu_2 = \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$$

As abordagens foram feitas em três tentativas, sendo a primeira tentativa, a de implementar através da linguagem MOSEL[1] (Xpress-Mosel 1.6.0) com o ambiente IVE (Xpress-IVE 1.16.0 e Xpress-Optimizer 16.01.02), sobre o sistema operacional Microsoft Windows XP 32 bits (service pack 3). Porém, nas heurísticas Retro e Wretro houve problema com os pacotes do XPRESS-MP no processo de realimentação. O processo de retroalimentação do sistema armazenava a ultima solução, não havendo a possibilidade de limpar da memória resultados da função **maximize** para o mesmo loop.

Na segunda tentativa, fez-se uso do ambiente de resolução ILOG[2]CPLEX 11.2 com tecnologia CONCERT 2.7 e linguagem de programação C++. A máquina que executou estas heurísticas estava equipada com Linux Ubuntu (compilador g++ 4.1), 2GB de memória RAM e um processador Intel Core 2 com 2.13GHz. No entanto, havia muitos "warnigs"no código, transformando o processo de debugar muito difícil. Foi necessário outra tentativa, agora usando o sistema operacional Windows XP.

Nesta última tentativa utilizada, foi feita a migração para o compilador do Visual Studio 2008, sobre o sistema operacional Microsoft Windows XP 64 bits (4GB de memoria RAM) e CPLEX 12.1 com tecnologia CONCERT 2.9, obtendo com isso um ótimo desempenho, e facilidade em solucionar os problemas.

Nas implementações onde foram usada a linguagem C++, houve a necessidade do uso dos containers **set** (usado no algoritmo (3.2)) e **vector** ("usado"como Ω), ambos da **Standard Template Library**[3]. No caso do container **set**, usou-se as seguintes funções: $insert(x)$, quem tem complexidade logarítmica, mas pode-se ter outras complexidades, depende de como se insere o elemento, $find(x)$ tem complexidade logarítmica, $erase(x)$, tem complexidade constante se x for a posição do elemento, e logarítmica se x for o valor do elemento.

No container **vector** foram usadas as seguintes funções: $push\_back(x)$, que tem complexidade constante e $insert(indice, x)$, que tem complexidade linear sobre o número de elementos.

Da biblioteca CPLEX, necessitou usar a função (outras foram usadas, porém, esta teve importância na limitação de tempo)

[1]Para saber sobre a linguagem MOSEL acesse http://www.fico.com/xpress

[2]Pode-se saber mais em http://www.ilog.com, mas atualmente encontra-se no endereço http://www-01.ibm.com/software/websphere/products/optimization/

[3]Para saber mais sobre a STL, veja http://www.cplusplus.com

$$cplex.setParam(IloCplex :: TiLim, 1200);$$

como forma de estabelecer o tempo limite de 1200 segundos (20 minutos) para a resolução de cada problema da mochila que compõe a primeira fase das heurísticas.

A proxima seção, exibe as tabelas resultante das simulações.

5.2 Discussão dos Resultados Computacionais

Nesta seção, encontram-se as tabelas com os resultados das heurísticas. As tabelas estão organizados por colunas, de forma a facilitar a compreensão. A coluna encabeçada por moch. representa a quantidade de mochilas, segs representa o tempo médio em segundos para a obtenção da solução da heurística e obj. representa a razão entre a média aritmética das funções objetivas da coluna representada pela heurística que encabeça a coluna pela média aritmética das funções objetivas da abordagem linear multiplicada por 100, gerando com isso uma porcentagem. Nas tabelas foram usadas as seguintes notações: $\star$ expõe que o problema não satifaz a restrição (1.2), $\bullet$ expressa que não satisfaz a restrição (5.3) e $*$ expõe que o tempo foi superior 172800 segundos (armazena a solução atual).

As tabelas (5.1), (5.2), (5.3), (5.4), (5.5) e (5.6) abaixo, exibem os dados das implementações, e a partir delas, pode-se ver a superioridade da abordagem linear e da heurística Wretro, por retornarem soluções em menor tempo, além de serem as maiores em relação as outras heurísticas.

A heurística Wretro, tem a vantagem de não variar muito o tempo para a obtenção de soluções com o aumento do números de itens na classe, enquanto que a abordagem linear retorna a maior solução em relação as outras heurísticas.

No próximo capítulo apresenta-se a conclusão deste trabalho, que baseia-se na modificação das formulações das heurísticas Retro e Wretro e também da abordagem linear.

Tabela 5.1 – Resultados com 5 classes / 5 itens

μ_1	μ_2	Linear		Retro		WRetro		Decomp.	
		moch.	segs	obj.	segs	obj.	segs	obj.	segs
	0.200	•	•	•	•	•	•	•	•
	0.300	•	•	•	•	•	•	•	•
	0.400	10	0.388	96.34	0.097	96.63	0.146	97.22	2.810
0.001	0.500	10	0.560	96.15	0.055	97.25	0.117	98.21	7.448
	0.600	10	0.573	96.32	0.068	96.72	0.107	98.87	11.110
	0.700	10	0.315	95.32	0.068	98.87	0.094	98.83	5.695
	0.800	10	0.156	97.96	0.065	98.38	0.114	99.91	4.011
	0.200	•	•	•	•	•	•	•	•
	0.300	•	•	•	•	•	•	•	•
	0.400	10	0.292	98.46	0.068	99.87	0.125	99.72	1.211
0.050	0.500	10	0.273	98.40	0.065	99.59	0.112	99.80	1.675
	0.600	10	1.961	98.76	0.060	98.81	0.109	99.08	1.474
	0.700	10	1.677	98.31	0.065	99.08	0.105	99.05	1.404
	0.800	10	0.232	98.48	0.060	99.42	0.094	99.78	0.583
	0.200	•	•	•	•	•	•	•	•
	0.300	•	•	•	•	•	•	•	•
	0.400	10	8.034	96.55	0.068	99.38	0.159	99.12	0.752
0.150	0.500	10	0.667	97.16	0.063	98.52	0.138	99.06	0.398
	0.600	10	1.076	97.25	0.063	98.12	0.112	98.58	0.242
	0.700	10	0.354	98.49	0.063	99.28	0.125	99.48	0.255
	0.800	10	0.258	97.59	0.060	97.74	0.130	97.62	0.232
	0.200	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
	0.300	•	•	•	•	•	•	•	•
	0.400	10	0.630	98.52	0.063	99.89	0.284	99.73	0.490
0.200	0.500	10	0.477	98.59	0.073	99.04	0.169	98.89	0.265
	0.600	10	0.274	97.76	0.068	98.23	0.130	99.18	0.161
	0.700	10	0.341	98.35	0.065	98.55	0.148	99.80	0.125
	0.800	10	0.206	98.48	0.068	99.35	0.135	99.24	0.125

Tabela 5.2 – Resultados com 5 classes / 10 itens

μ_1	μ_2	Linear		Retro		WRetro		Decomp.	
		moch.	segs	obj.	segs	obj.	segs	obj.	segs
	0.200	10	211.859	97.04	0.218	97.37	0.452	98.64	2450.344
	0.300	10	57.890	97.60	0.172	99.71	0.470	99.03	1993.109
	0.400	10	3.641	96.36	0.281	97.60	0.406	98.17	961.453
0.001	0.500	10	1.402	98.00	0.328	98.91	0.437	98.71	859.797
	0.600	10	0.344	96.30	0.281	98.78	0.423	98.39	616.046
	0.700	10	0.317	98.09	0.105	98.43	0.456	98.45	608.762
	0.800	10	0.207	98.67	0.082	98.45	0.406	98.67	537.494

Tabela 5.3 – Resultados com 5 classes / 30 itens

μ_1	μ_2	Linear		Retro		WRetro		Decomp.	
		moch.	segs	obj.	segs	obj.	segs	obj.	segs
	0.200	10	29.875	96.20	0.091	97.26	0.154	98.67	*
	0.300	10	74.253	98.73	0.117	99.42	0.167	97.05	*
	0.400	10	320.216	93.83	0.115	94.35	0.185	94.76	*
0.001	0.500	10	67.773	96.76	0.104	98.28	0.180	93.23	*
	0.600	10	713.208	95.71	0.104	98.70	0.172	95.07	*
	0.700	10	597.922	93.60	0.112	95.07	0.185	94.34	*
	0.800	10	95.245	97.09	0.109	97.64	0.187	98.97	*
	0.200	10	6.026	90.35	0.107	91.32	0.172	96.58	*
	0.300	10	875.320	89.83	0.133	91.20	0.206	92.01	*
	0.400	10	252.456	91.53	0.123	92.49	0.195	96.84	*
0.050	0.500	10	613.716	94.24	0.130	96.51	0.188	97.12	*
	0.600	10	135.685	94.37	0.123	97.64	0.198	98.34	*
	0.700	10	209.292	92.86	0.120	96.59	0.193	96.78	*
	0.800	10	497.167	94.45	0.143	96.87	0.184	98.56	*

Tabela 5.4 – Resultados com 10 classes / 5 itens

μ_1	μ_2	Linear		Retro		WRetro		Decomp.	
		moch.	segs	obj.	segs	obj.	segs	obj.	segs
	0.200	10	0.297	96.85	0.367	99.23	1.024	99.44	172.141
	0.300	10	0.695	99.80	0.430	99.82	0.609	99.90	267.531
	0.400	10	1.063	96.04	1.702	99.90	2.704	99.96	242.359
0.001	0.500	10	0.766	95.35	1.468	98.07	2.609	98.71	708.969
	0.600	10	0.890	98.71	1.188	98.79	2.782	99.56	65.094
	0.700	10	0.234	98.96	0.445	99.07	0.641	99.17	6.422
	0.800	10	0.468	98.66	0.383	99.95	0.664	99.54	12.913
	0.200	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
	0.300	10	0.679	99.45	0.516	99.59	1.095	99.73	6.188
	0.400	10	5.969	99.33	2.655	99.79	3.096	99.84	14.265
0.200	0.500	10	2.407	98.20	2.015	99.00	4.856	99.38	9.281
	0.600	10	11.485	99.65	2.094	99.97	13.374	99.98	3.844
	0.700	10	0.937	98.88	0.477	99.81	0.734	99.80	0.656
	0.800	10	1.031	99.02	0.524	99.42	0.772	99.09	0.594

Tabela 5.5 – Resultados com 10 classes / 10 itens

μ_1	μ_2	Linear		Retro		WRetro		Decomp.	
		moch.	segs	obj.	segs	obj.	segs	obj.	segs
	0.200	10	2.672	98.83	1.258	99.00	0.820	99.11	6395.868
	0.300	10	1.047	94.09	0.493	99.90	0.821	99.53	4668.656
	0.400	10	3.891	94.38	3.376	96.05	9.861	99.65	3981.827
0.001	0.500	10	3.297	96.88	2.295	99.34	3.000	99.98	3129.974
	0.600	10	0.890	95.94	1.564	99.20	2.000	99.45	2830.102
	0.700	10	0.563	97.43	0.780	99.76	0.891	99.77	1346.759
	0.800	10	0.359	94.20	0.734	99.74	1.031	99.67	374.097

Tabela 5.6 – Resultados com 10 classes / 15 itens

μ_1	μ_2	Linear		Retro		WRetro		Decomp.	
		moch.	segs	obj.	segs	obj.	segs	obj.	segs
	0.200	10	1.906	98.80	0.534	99.97	0.875	99.82	12073.091
	0.300	10	2.187	98.96	0.750	99.03	1.091	99.42	9780.313
	0.400	10	2.644	99.28	0.913	99.77	1.062	99.96	8297.696
0.001	0.500	10	2.159	98.31	0.819	99.92	1.003	99.97	5694.247
	0.600	10	2.841	98.45	0.969	99.05	1.621	99.74	3772.490
	0.700	10	92.409	98.23	0.616	99.37	0.950	99.48	2522.731
	0.800	10	6.966	98.57	0.894	99.36	1.557	99.88	2741.909
	0.200	10	7.481	97.36	1.662	98.98	8.970	99.08	21713.520
	0.300	10	48.288	97.82	1.403	98.23	7.122	99.42	13393.319
	0.400	10	2.647	98.82	0.922	99.73	2.334	99.93	4367.984
0.050	0.500	10	0.828	98.34	0.616	98.69	1.516	99.13	2870.296
	0.600	10	0.514	98.76	0.421	98.47	0.782	99.56	1836.711
	0.700	10	0.412	99.02	0.288	99.15	0.259	99.91	453.184
	0.800	10	0.173	98.71	0.117	99.05	0.190	99.63	218.918

CAPÍTULO 6
CONCLUSÃO

Este trabalho abordou propostas para modificações nas restrições de disponibilidade das heurísticas Retro e Wretro, além da função objetivo e a restrição de capacidade da abordagem linear, afim de obter um ótimo desempenho em relação as heurísticas em sua forma original, apresentado nos trabalhos (Spolador, 2005) e (Hoto *et al.*, 2006).

Nas simulações, observou-se que a heurística Wretro tem a característica de retornar uma solução melhor em relação a heurística Retro (caso particular da heurística Wretro), além de retornar uma solução viável em um curto tempo em relação a heurística de Decomposição.

A abordagem linear supera as três heurísticas (Decomposição, Retro e Wretro) em relação a solução, sem a necessidade de muito tempo, e isto dá a abordagem linear uma vantagem frente as heurísticas abordadas.

O estudo sobre a abordagem linear continuará, pela expectativa do problema (4.1 - 4.7) ser equivalente ao problema (4.18 - 4.25), e estudos se darão para mostrar a possível equivalência dos problemas.

Outro estudo que pretende-se abordar é o problema bidimensional (o trabalho teve como base um problema unidimensional) que chamar-se-á Problema da Mochila Bidimensional com Compartimentos Guilhotinados. Este problema pode ser expresso como: organizar itens de duas dimensões (comprimento e largura), apenas permitindo a rotação dos itens, exibido na figura (6.1), que irão compor a mochila bidimensional, obedecendo as restrições do problema da mochila compartimentada restrita unidimensional, tal que este problema respeite os compartimentos (itens de mesma classe). Abaixo, a figura (6.2) exibe a idéia do problema.

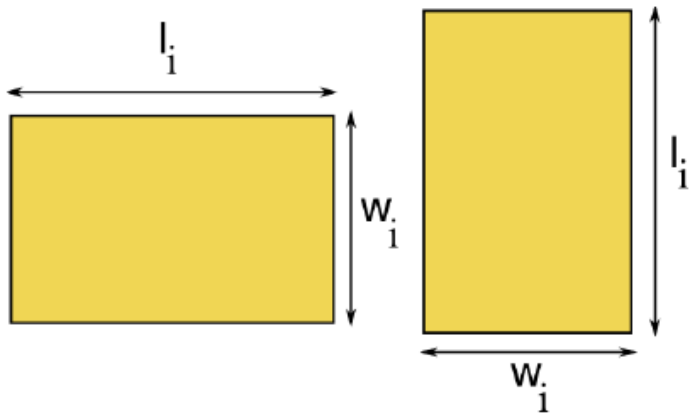

Figura 6.1 – Rotação permitida pelo problema

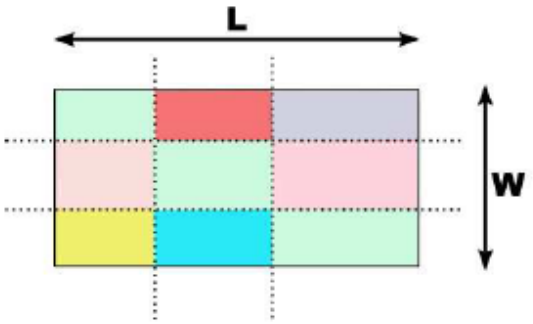

Figura 6.2 – Compartimentação Guilhotinada

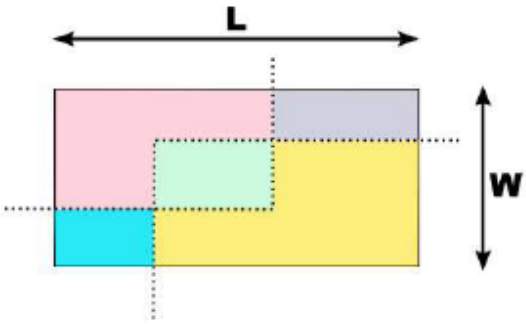

Figura 6.3 – Compartimentação Não-guilhotinada

Há outro tipo de problema de compartimentação, feito por cortes não-guilhotinados e a figura (6.3) exibe este. O problema feito por cortes não-guilhotinados é dificil, pela demarcação dos cortes (compartimentos), sendo comum o uso de compartimentação guilhotinada.

REFERÊNCIAS

[1] M. C. Goldbarg, *Otimização Combinatória e Programação Linear*. Editora Campus, 2000.

[2] R. Hoto, M. Arenales, and N. Maculan, "O problema da mochila compartimentada," tech. rep., Universidade Estadual de Londrina, 1999.

[3] R. Hoto, *O Problema da Mochila Compartimentada aplicado no Corte de Bobinas de Aço*. PhD thesis, Universidade Federal do Rio de Janeiro, 2001.

[4] R. Sedgewick, *Algorithms*. Addison-Wesley, 1983.

[5] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. Chichester: John Wiley & Sons, March 1997.

[6] G. Dantzig, "Discrete-variable extremum problems," *Operations Research*, vol. 5, no. 2, pp. 266–288, 1957.

[7] V. Chvátal, *Linear Programming*. W. H. Freeman and Company, 1983.

[8] R. E. Johnston and L. R. Khan, "Bounds for nested knapsack problems," *European Journal of Operational Research*, vol. 81, pp. 154–165, 1995.

[9] J. F. Oliveira and G. Wäscher, "Editorial: Cutting and packing," *European Journal of Operational Research*, vol. 183, pp. 1106–1108, 2007.

[10] F. do Prado Marques and M. N. Arenales, "O problema da mochila compartimentada e aplicações," *Pesquisa Operacional*, vol. 22, pp. 285–304, 2002.

[11] R. Hoto, N. Maculan, F. Marques, and M. Arenales, "Um problema de corte com padrões compartimentados," *Pesquisa Operacional*, vol. 23, pp. 169–187, 2003.

[12] F. do Prado Marques, *O problema da mochila compartimentada e aplicações*. PhD thesis, USP, São Carlos, 2004.

[13] F. L. Spolador, "O problema da mochila compartimentada - o caso restrito." TCC, 2005.

[14] R. Hoto, A. Fenato, H. Yanasse, N. Maculan, and F. Spolador, "Uma nova abordagem para o problema da mochila compartimentada," *XXXVIII SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL*, vol. 38, pp. 1760–1771, 2006.

[15] F. do Prado Marques and M. N. Arenales, "The constrained compartmentalised knapsack problem," *Computers & Operations Research*, vol. 34, pp. 2109–2129, 2007.

[16] R. Hoto, M. Arenales, and N. Maculan, "The one dimensional compartmentalised knapsack problem: A case study," *European Journal of Operational Research*, vol. 183, pp. 1183–1195, 2007.

[17] R. S. Hoto, F. Spolador, and N. Maculan, "Um sistema computacional para auxiliar no planejamento de cortes em bobinas de aço," *Semina: Ciências Exatas e Tecnológicas*, vol. 28, pp. 55–64, 2007.

[18] R. Hoto, F. Spolador, and F. Marques, "Resolvendo mochilas compartimentadas restritas," *XXXVII SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL*, vol. 37, pp. 1756–1766, 2005.

[19] N. Christofides and C. Whitlock, "An algorithm for two dimensional cutting problems.," *Operations Research*, vol. 25, pp. 30–44, 1977.

[20] G. Belov and G. Scheithauer, "A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths," *European Journal of Operational Research*, vol. 141, pp. 274–294, 2002.

[21] D. Pisinger, *Algorithms for Knapsack Problems*. PhD thesis, University of Copenhagen, Fevereiro 1995.