UNIVERSIDADE ESTADUAL DE LONDRINA

MATHEUS HENRIQUE PIMENTA ZANON

SOLUÇÕES HEURÍSTICAS PARA O PROBLEMA DA MOCHILA COMPARTIMENTADA

Londrina
2019

MATHEUS HENRIQUE PIMENTA ZANON

SOLUÇÕES HEURÍSTICAS PARA O PROBLEMA DA MOCHILA COMPARTIMENTADA

Dissertação de mestrado apresentada ao Departamento de Matemática da Universidade Estadual de Londrina, como requisito parcial para a obtenção do Título de MESTRE em Matemática Aplicada e Computacional.

Orientador: Prof. Dr. Robinson Samuel Vieira Hoto
Coorientador: Fábio Sakuray

Londrina
2019

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

Dados Internacionais de Catalogação-na-Publicação (CIP)

S232c Matheus Henrique Pimenta Zanon.
Soluções Heurísticas para o Problema da Mochila Compartimentada / Matheus Henrique Pimenta Zanon – Londrina, 2019.
74 f. : il.

Orientador: Robinson Samuel Vieira Hoto
Coorientador: Fábio Sakuray
Dissertação (Mestrado em Matemática Aplicada e Computacional) Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Matemática Aplicada e Computacional, 2019.

Inclui Bibliografia.

1. Computação - Matemática aplicada - Teses. 2. Criptografia - Teses. 3. Curvas elípticas - Teses. 4. Anéis de endomorfismo - Teses. 5. Corpos finitos (álgebra) - Teses. I. Robinson Samuel Vieira Hoto, Fábio Sakuray. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Matemática Aplicada e Computacional. III. Soluções Heurísticas para o Problema da Mochila Compartimentada.

519.681-7

MATHEUS HENRIQUE PIMENTA ZANON

SOLUÇÕES HEURÍSTICAS PARA O PROBLEMA DA MOCHILA COMPARTIMENTADA

Dissertação de mestrado apresentada ao Departamento de Matemática da Universidade Estadual de Londrina, como requisito parcial para a obtenção do Título de MESTRE em Matemática Aplicada e Computacional.

BANCA EXAMINADORA

Prof. Dr. Fábio Sakuray
Universidade Estadual de Londrina

Profª. Dra. Glaucia Maria Bressan
Universidade Tecnológica Federal do Paraná

Prof. Dr. Wesley Attrot
Universidade Estadual de Londrina

Londrina, 26 de fevereiro de 2019.

Dedico este trabalho a minha mãe, Maria de Lourdes, pessoa que mais amo nesta vida.

AGRADECIMENTOS

A minha mãe, Maria de Lourdes, por ser a grande responsável pela minha formação pessoal, pelo seu amor e dedicação direcionada à mim.

A minha parceira, Heloiza, por me apoiar, ajudar e dar apoio durante todo o tempo.

A minha família por ser base de minha formação, especialmente ao Paulo Sérgio, pelas conversas e conselhos direcionados durante este período.

Ao meu orientador, Robinson Hoto, pela confiança, atenção dispensada ao trabalho, pelas sugestões e decisões tomadas.

Ao meu coorientador, Fábio Sakuray, pessoa pelo qual sou muito grato pelo auxílio, ensino, sugestões e atenção disponibilizadas ao trabalho de modo permanente.

A professora Glaucia pela amizade, considerações e ajuda durante a conclusão desta etapa. Ao professor Wesley pelas ideias e observações realizadas durante a elaboração do trabalho.

Aos meus amigos e professores do PGMAC e do SIMULAB, os quais foram de enorme importância no decorrer do período, mostrando meus erros, auxiliando, contribuindo e apoiando. As conversas durante esse período foram de grande importância para a conclusão desta etapa.

A CAPES pelo apoio concedido para o desenvolvimento deste estudo.

A FICO pela disponibilização do *software* Xpress para a execução das simulações numéricas.

"Ter um cérebro inteligente não é suficiente, também precisamos de um coração acolhedor."
S. S. Dalai Lama

PIMENTA-ZANON, Matheus Henrique. **Soluções Heurísticas para o Problema da Mochila Compartimentada**. 2019. 52. Dissertação (Mestrado em Matemática Aplicada e Computacional) – Universidade Estadual de Londrina, Londrina, 2019.

RESUMO

Este trabalho aborda o Problema da Mochila Compartimentada em sua modelagem linear proposta por Inarejos (2015) [13]. Utilizando-se da particularidade do modelo linear, são propostas três novas heurísticas. A heurística denominada p_kX utiliza o *software* FICO Xpress na resolução dos subproblemas, apresentando soluções próximas ao ótimo; outra heurística é definida como $p_kGULOSO$ e usa o método guloso em sua resolução, gerando soluções em um tempo de execução baixo. Por fim, a heurística $p_kMTComp$ utiliza o método de resolução exata (MTU2) proposto por Martello e Toth (1991) [21]. Experimentos preliminares indicam que a heurística $p_kMTComp$, apresenta soluções próximas ao ótimo, sendo um método promissor na resolução do Problema da Mochila Compartimentada, quando comparada com outra heurística reconhecida na literatura.

Palavras-chave: Problema da Mochila Compartimentada; Heurística; Otimização

PIMENTA-ZANON, Matheus Henrique . **Heuristic Solutions for the Compartmentalized Knapsack Problem**. 2019. 52. Dissertação (Mestrado em Matemática Aplicada e Computacional) – Universidade Estadual de Londrina, Londrina, 2019.

ABSTRACT

This work deals with the Problem of the Compartmentalized Knapsack in its linear modeling proposed by Inarejos (2015) [13]. Using the particularity of the linear model, three new heuristics are proposed. The heuristic called pk_X uses FICO Xpress software in solving subproblems, presenting solutions close to the optimum, another heuristic defined as $p_kGULOSO$ uses the greedy method in its resolution,generating solutions at a low runtime. Finally, the heuristic $p_kMTComp$ uses the exact resolution method (MTU2) proposed by Martello and Toth (1991) [21]. Preliminary experiments indicate that the heuristic $p_kMTComp$, presents solutions close to the optimum, being a promising method in solving the Problem of the Compartmentalized Knapsack, when compared with other heuristics recognized in the literature.

Keywords: Compartmentalized Knapsack Problem; Heuristic; Optimization

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

1 INTRODUÇÃO

O desenvolvimento industrial e tecnológico está em constante expansão no cenário nacional [17]. Desta forma, as indústrias tem buscado métodos mais eficazes na resolução de problemas, como por exemplo, otimizar recursos e obter a melhor solução, podendo ser em termos de tempo de execução mais rápidos ou em valores obtidos. Buscar alternativas de métodos que apresentem soluções em um espaço de tempo cada vez menor é um dos objetivos dos estudos da otimização matemática e pesquisa operacional.

Durante as etapas do processo produtivo industrial, diversas decisões são tomadas. Essas decisões podem ser transcritas matematicamente, como por exemplo definir a melhor aplicação financeira a investir, apresentar a combinação de corte que resulta no melhor aproveitamento da matéria prima ou expor os melhores itens a serem inseridos dentro de um certo compartimento. Até mesmo na reprodução via *streaming* de um filme *online*, na definição da melhor resolução a ser reproduzida baseada na disponibilidade da rede no momento, são problemas que envolvem a tomada de decisões.

Para problemas nos quais o número de possibilidades é pequeno, o número de possíveis combinações é menor. No caso de um grande número de possibilidades e restrições dentro das decisões, este universo de possibilidades torna-se complexo e com alto custo computacional. Devido a isso, a elaboração de métodos que resultem na solução ótima em tempos mais ágeis e com custo computacional pequeno é necessária.

O Problema da Mochila é um tipo clássico de problema de otimização que exige uma tomada de decisão. O Problema da Mochila Clássico consiste na escolha de um subconjunto de itens aos quais estão associados uma utilidade e largura (ou peso) que definirá a quantidade de espaço da mochila que este utilizará. O objetivo de tal escolha é proporcionar a melhor utilidade ao final do preenchimento da mochila [21]. O Problema da Mochila é uma classe de Problema de Programação Linear inteira e sua classificação é baseada na complexidade de resolução como problemas *NP-Difícéis* [14]. Esta é uma das razões que motivam o desenvolvimento de heurísticas para a resolução dos problemas em busca de soluções "boas" em sacrifício da otimalidade.

Uma variação do Problema da Mochila clássico é o Problema da Mochila Compartimentada, que consiste na criação de compartimentos de capacidades limitadas por um intervalo, dentro da mochila de capacidade conhecida [11].

Em Martello e Toth [21] tem-se um exemplo inicial sobre o Problema da Mochila e como a elaboração de novos algoritmos fornecem a solução em poucos segundos. Assim, suponha que uma pessoa tenha um capital (c) (ou parte dele) para investimentos, e considera-se analisar n tipos de investimentos. Então considere p_j o retorno esperado para o investimento j e w_j o valor necessário para o investimento do tipo j. Neste caso, a solução ótima deste Problema de Mochila, será a melhor escolha de investimento a ser realizado. Para $n = 60$,

um computador, levaria em torno de 30 anos para a geração de todas as possibilidades binárias de solução para esse problema em específico, se $n = 61$, este tempo subiria para em média 60 anos, assim, um algoritmo eficaz consegue apresentar o valor ótimo em poucos segundos para $n = 100000$ [21].

Com esta motivação, o objetivo deste trabalho é propor três novas heurísticas para a resolução do Problema da Mochila Compartimentada, utilizando uma adaptação do método exato de resolução proposto por Martello e Toth [21], e a particularidade do modelo linear do Problema da Mochila Compartimentada que é proposto por Inarejos [12], o detalhamento é apresentado no capítulo 3. Tais heurísticas buscam ser ágeis, utilizando poucos recursos computacionais e com resultados próximos ao ótimo para o Problema da Mochila Compartimentada.

Os experimentos preliminares indicam que a heurística proposta é promissora, apresentando resultados mais próximos ao ótimo quando comparada com a heurística $w - capacidades$, proposta por Marques [18], em relação ao tempo de execução os testes prévios indicam um tempo de execução inferior, também em comparação com a heurística usual da literatura.

O texto está dividido em cinco capítulos com a seguinte estrutura: o capítulo 1 apresenta uma visão geral do Problema da Mochila e os objetivos do trabalho, no capítulo 2 é definido o Problema da Mochila Compartimentada de maneira ampla, em suas modelagens não-linear e linear, uma aplicação do Problema da Mochila Compartimentada na indústria de corte de bobinas de aço e apresenta-se duas heurísticas reconhecidas na literatura. Ainda no capítulo 2 é apresentado o método de *branch and bound* e o método de Martello e Toth [21], bastante explorado na elaboração de uma heurística proposta. No capítulo 3 são detalhadas as três heurísticas propostas e seus algoritmos. As simulações numéricas e os resultados obtidos através das análises de execução das heurísticas são apresentadas no capítulo 4. O capítulo 5 dedica-se às conclusões e trabalhos futuros.

2 A MOCHILA COMPARTIMENTADA

Este capítulo apresenta uma breve revisão bibliográfica sobre o Problema da Mochila, com ênfase no Problema da Mochila Compartimentada.

2.1 O PROBLEMA DA MOCHILA

Diversas situações práticas recaem em tomar decisões, que podem ser simples, com duas alternativas, ou complexas envolvendo mais variáveis. No caso da decisão *binária*, caso mais simples, pode-se associar uma variável binária $x \in \{0, 1\}$, onde $x_j = 1$ toma a alternativa j como verdade e $x_j = 0$ indica a rejeição da alternativa j [14].

Fundamentalmente, um modelo de decisão linear é definido por n variáveis binárias, isto é $x_j \in \{0, 1\}$, que representa a *j-ésima* decisão e por valores de utilidade u_j, os quais representam a *j-ésima* utilidade. Sem perda de generalidade, os valores das utilidades serão estritamente positivos e $\sum_{i=1}^{n} x_i$ representa a soma total das utilidades, quando $x_j = 1$ para todo $j = 1, \ldots, n$. [14].

O Problema da Mochila pode ser formalizado como um conjunto de itens N, com n itens pertencendo ao conjunto N, onde para cada item n está associada uma utilidade u e uma largura (ou peso) l, usualmente estes valores são inteiros e positivos. O objetivo é selecionar um subconjunto de N no qual a soma de todas as utilidades associadas a este subconjunto seja a máxima e a soma de todos os pesos associados a este subconjunto não exceda a capacidade da mochila (L). Esta última informação é denominada como restrição física da mochila. Na Figura 2.1 tem-se a ilustração de uma mochila simples, de capacidade $L = 4$kg e estão dispostos inúmeros itens com tamanhos e utilidades para serem inseridos no interior da mesma, ilustrando o caso mais simples do Problema da Mochila.

Figura 2.1: Ilustração da Mochila.

O Problema da Mochila pode ser expresso como:

$$\text{Maximizar: } z = \sum_{j=1}^{n} u_j x_j \tag{2.1}$$

$$\text{sujeito a: } \sum_{j=1}^{n} l_j x_j \leq L \tag{2.2}$$

$$x_j \in \{0, 1\}, j = 1, 2, \ldots, n. \tag{2.3}$$

A equação 2.1 denota o objetivo de maximizar a utilidade dos itens, a restrição 2.2 denota a capacidade total da mochila, isto é, a soma dos itens selecionados não deve ser superior a capacidade total da mochila (L), por fim a restrição 2.3 denota o domínio do problema, neste caso variáveis binárias. Este problema (2.1) - (2.3) é definido como *Problema da Mochila* $0-1$ [21], e ocorre quando o domínio das variáveis de decisão é binário.

Se a soma de todos os itens for menor ou igual a capacidade total da mochila, a restrição 2.2 não terá efeito sobre o problema, resultando na solução trivial $x_j = 1$ para todo $j = 1, 2, \ldots, n$. Dessa forma, suponha que $\sum_{j=1}^{n} l_j \geq L$, e também de forma análoga suponha que o peso de cada item não ultrapasse o valor da capacidade total da mochila. O vetor da solução ótima é representado por $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ e a solução ótima é representada por z^*. O conjunto X^* representa o conjunto da solução ótima, isto é, o conjunto dos itens que correspondem à solução ótima [14].

Em outro tipo de problema da mochila denominado *Problema da Mochila Restrito*, o domínio da variável de interesse não é binário, com isto é possível inserir mais de um item do tipo j na mochila, até sua demanda d_j, que é a quantidade disponível do item do

tipo j, ser satisfeita, a demanda pode ser designada em outros textos como um limitante superior para cada item do tipo j, [21]. A modelagem matemática para o *Problema da Mochila Restrito* é apresentada pelas equações (2.4) - (2.7).

$$\text{Maximizar: } z = \sum_{j=1}^{n} u_j x_j \tag{2.4}$$

$$\text{sujeito a: } \sum_{j=1}^{n} l_j x_j \leq L \tag{2.5}$$

$$0 \leq x_j \leq d_j, \text{ para } j = 1, \ldots, n \tag{2.6}$$

$$x_j \text{ é inteiro, para } j = 1, \ldots, n \tag{2.7}$$

A nova restrição 2.6 representa a quantidade de itens do tipo j disponível. São aceitas as seguintes condições para evitar a solução trivial:

$$\sum_{j=1}^{n} d_j l_j \geq L \tag{2.8}$$

$$d_j l_j \leq L, j = 1, \ldots, n \tag{2.9}$$

No caso de $d_j \to \infty$ tem-se o *Problema da Mochila Irrestrito*, assim a restrição 2.8 é sempre verdadeira, já a restrição 2.9 é substituída por $l_j \leq L$, $j = 1, \ldots, n$ [11].

Um outro caso do Problema da Mochila $0 - 1$ é o *Problema da Soma de Subconjuntos* [22], que ocorre no caso quando $l_j = u_j$ para todo $j = 1, \ldots, n$. Neste caso, tem-se a seguinte modelagem matemática:

$$\text{Maximizar: } z = \sum_{j=1}^{n} l_j x_j \tag{2.10}$$

$$\text{sujeito a: } \sum_{j=1}^{n} l_j x_j \leq L \tag{2.11}$$

$$x_j \in \{0, 1\}, j = 1, 2, \ldots, n. \tag{2.12}$$

Tem-se ainda o *Problema da Mochila Quadrático*, um dos problemas com uma grande quantidade de estudos disponíveis, [3, 25], na área de Problemas de Mochila Não Lineares [19]. Caracteriza-se por apresentar a função objetivo não linear ou que envolve restrições não lineares. A formulação matemática é dada por:

$$\text{Maximizar: } f(x) \tag{2.13}$$

$$\text{sujeito a: } \sum_{j=1}^{n} l_j x_j \leq L \tag{2.14}$$

$$x_j \geq 0 \text{ e inteiros}, j = 1, 2, \ldots, n. \tag{2.15}$$

Onde $f(x)$ representa uma função quadrática de x na forma $xSx^T + cx^T$ com $x = (x_1, x_2, \ldots, x_n)$, $c = (c_1, c_2, \ldots, c_n)$ e $S = (s_1, s_2, \ldots, s_n)$ para algum j com $s_j \neq 0$, para $j = 1, 2, \ldots, n$.

Outros Problemas de Mochila apresentados por Martello e Toth [21], Kellerer [14], Hoto [11] e Marques [19] são os *Problema de Múltiplas Mochilas 0-1*, *Problema de Designação Generalizada*, *Bin-Packing* e *Problema da Mochila Encapsulada*, serão apresentadas as formulações matemáticas e as descrições de cada tipo de problema com diversas mochilas.

O Problema de Múltiplas Mochilas $0-1$ compõem-se em carregar um número finito de mochilas m, onde $m \leq n$, com as capacidades L_j, $j = 1, 2, \ldots, m$ associadas a cada mochila m. As demais informações são análogas ao Problema da Mochila $0 - 1$. O objetivo é selecionar subconjuntos distintos de itens de maneira que cada subconjunto possa ser associado a uma mochila, de forma que a soma dos pesos dos itens dos subconjuntos não ultrapasse a capacidade da mochila associada a este. A formulação matemática é apresentada por:

$$\text{Maximizar: } z = \sum_{i=1}^{m} \sum_{j=1}^{n} u_j x_{ij} \tag{2.16}$$

$$\text{sujeito a: } \sum_{j=1}^{n} l_j x_{ij} \leq L \text{ , } i = 1, 2, \ldots, m \tag{2.17}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 \text{ , } j = 1, 2, \ldots, n \tag{2.18}$$

$$x_{ij} \in \{0, 1\}, j = 1, 2, \ldots, n \text{ , i = 1, 2, ..., m.} \tag{2.19}$$

Quando $m = 1$, tem-se o Problema da Mochila $0 - 1$ simples.

No artigo de Xavier [28] é aplicado uma generalização do Bin-Packing a sistemas de *Video-on-Demand*, atualmente muito utilizados, como por exemplo no YouTube, Netflix, Amazon e outros fornecedores deste tipo de solução. Os autores, propõem a criação de caixas de capacidade B com C compartimentos e n itens de Q diferentes classes, e cada item $i \in \{1, \ldots, n\}$ está associado a uma classe c_i e tamanho s_i. O objetivo é obter um servidor que atenda todos os pedidos de filmes com o menor número de discos rígidos. O Problema da Designação Generalizada o Bin-Packing é estudado em [14, 21].

Em um outro artigo, Sobhani [26] realiza-se a elaboração de um novo algo-

ritmo de otimização de *Video-on-Demand* que utiliza o protocolo DASH (*Dynamic Adaptive Streaming over HTTP* [1]) para redes *wireless*. São propostas duas funções objetivo: a primeira tem o objetivo de maximizar a taxa média de satisfação dos clientes; essa taxa média é obtida através de um parâmetro que envolve diversas variáveis. A segunda função tem como objetivo minimizar o impacto negativo das trocas de qualidade na reprodução dos filmes. Os resultados indicam uma transmissão com um menor número de interrupções e uma qualidade média superior da qualidade de transmissão, quando comparada com o método de Thang [27], como trabalho futuro, os autores buscam a elaboração de novas heurísticas para uma solução mais ágil, quando comparada com métodos disponíveis no mercado atualmente.

O Problema da Mochila Encapsulada consiste na criação de cápsulas ou *casulos* de tamanhos predefinidos onde serão depositados os itens de interesse. Dentro de cada cápsula podem ser construídas novas cápsulas de tamanhos predefinidos [11]. Este problema é muito próximo do problema de interesse desta dissertação, que é o *Problema da Mochila Compartimentada*. A diferença sutil entre os dois problemas é que no Problema da Mochila Encapsulada os compartimentos (cápsulas) possuem tamanhos fixos, enquanto no Problema da Mochila Compartimentada, o tamanho dos compartimentos são maleáveis.

2.2 O Problema da Mochila Compartimentada

Na seção anterior foram apresentados alguns exemplos do Problema da Mochila, incluindo um exemplo no qual a criação de compartimentos dentro da mochila é necessário. Esta seção apresenta o *Problema da Mochila Compartimentada* com suas formulações não-linear [11, 16] e linear [12].

Para uma visualização sobre o Problema da Mochila Compartimentada, tome a seguinte aplicação proposta por Hoto [10]: precisa-se selecionar itens para serem transportados até um ponto de difícil acesso (por exemplo uma região montanhosa). Inicialmente os itens são transportados num vagão de trem, em seguida, os itens são transportados por meio de pequenos furgões até outro ponto em que o transporte só pode ser feito por carregadores.

A carga de cada carregador deve ser proveniente de somente um dos furgões. Para cada um dos três tipos de transporte existe uma capacidade de carga e custo operacional associado. O problema consiste em selecionar itens que serão carregados em cada etapa do transporte, de tal forma que seja maximizada a utilidade dos itens e minimizado os custos de transporte, com isto é proposto um modelo por Johnston e Khan (1995 apud Hoto *et al.*, 2003), onde as variáveis de decisão são: x_{ij}^s, onde $x_{ij}^s = 1$ se o item i for designado para a mochila j na etapa de transporte s, e 0 no caso contrário, e y_j^s, onde $y_j^s = 1$ se a mochila j é usada na etapa de transporte s, e igual a 0 caso contrário.

Tem-se também os parâmetros γ^s e c^s que representam, respectivamente, o custo e a capacidade da mochila nas etapas de transporte $s = 1, ..., k$. O modelo é dado por:

$$\text{Maximizar: } \sum_{i=1}^{m} u_i x_{i1}^1 - \sum_{j=1}^{m}\sum_{s=2}^{k} \gamma^s y_j^s \tag{2.20}$$

$$\text{sujeito a: } \sum_{j=1}^{m} p_i x_{i1}^1 \leq c \tag{2.21}$$

$$\sum_{j=1}^{m} p_i x_{ij}^s \leq c^s y_j^s, \text{ para } j = 1, ..., m, \; s = 2, ..., k \tag{2.22}$$

$$x_{i1}^1 \leq \sum_{j=1}^{m} x_{ij}^s, \text{ para } i = 1, ..., m, \; s = 1, ..., k \tag{2.23}$$

$$\max_j (x_{pj}^s + x_{qj}^s) \leq \max_j (x_{pj}^{s-1} + x_{qj}^{s-1}), \tag{2.24}$$

$$\text{para } s = 3, ..., k \text{ e } p, q = 1, ..., m, \; p \neq q$$

$$x_{ij}^s, \; y_j^s \in \{0, 1\}, \; i, j = 1, ..., m, \; s = 1, ..., k. \tag{2.25}$$

Após o exemplo, considere uma mochila de capacidade L e um conjunto de índices N, particionado em subconjuntos N_k, onde $k = 1, 2, \ldots, q$, $N = \bigcup_{k=1}^{q} N_k$ e $N_j \cap N_k = \varnothing$, para $k \neq j$ e $k, j \in (1, 2, \ldots, q)$. Para cada índice em N está associado uma largura (ou peso) l_i, uma utilidade u_i e uma demanda d_i. O interior da mochila é organizado de modo que apenas itens da mesma classe podem ser inseridos dentro do mesmo compartimento.

Cada compartimento possui um tamanho variável, porém limitados inferiormente por (L_{min}^k) e superiormente por (L_{max}^k). Cada compartimento possui um custo fixo c_k e uma perda fixa S_k, por serem incluídos, que variam de acordo com cada classe. Em cada classe pode ser construído mais de um compartimento para a alocação dos itens. Desta maneira o *Problema da Mochila Compartimentada* visa definir quais compartimentos irão compor a mochila de maneira que subtraindo os custos dos compartimentos a utilidade seja máxima.

Uma formulação matemática para o caso não-linear é dada por Leão [16], sendo que os dados do modelo são:

- L: capacidade da Mochila;
- $N = 1, 2, \ldots, n$: o conjunto de índices dos itens;
- l_i: a largura (ou peso) de cada item, $i = 1, 2, \ldots, n$;
- u_i: a utilidade de cada item, $i = 1, 2, \ldots, n$;
- d_i: a demanda de cada item, isto é, o número máximo de itens permitido na mochila, $i = 1, 2, \ldots, n$;
- q: o número total de classes;

- A_k: o conjunto dos itens pertencentes a classe k, onde $k = 1, 2, \ldots, q$, $N = \bigcup_{k=1}^{q} A_k$ e $A_j \cap A_k = \varnothing$, para $k \neq j$ e $k, j \in (1, 2, \ldots, q)$;
- c_k: custo de incluir um compartimento para itens da classe k, $k = 1, 2, \ldots, q$;
- L_{min}^k e L_{max}^k: são respectivamente os limitantes inferiores e superiores para cada classe k, $k = 1, 2, \ldots, q$;
- S_k: perda da capacidade da mochila devido à inclusão do compartimento para itens da classe k, $k = 1, 2, \ldots, q$;
- N_k: número total dos possíveis compartimentos da classe k, $k = 1, 2, \ldots, q$.

E as variáveis são:

- x_{ijk}: número de itens do tipo i da classe k no compartimento j, onde $i = 1, 2, \ldots, n$, k, $k = 1, 2, \ldots, q$ e $j = 1, 2, \ldots, N_k$;
- β_{jk}: número de vezes que o compartimento j para itens da classe k é repetido na mochila, onde k, $k = 1, 2, \ldots, q$ e $j = 1, 2, \ldots, N_k$.

Algumas observações são realizadas de modo a evitar soluções triviais, para $k = 1, 2, \ldots, q$:

- $\sum_{i \in A_k} d_i l_i > L_{max}^k$;
- $l_i < L_{max}^k$, $i \in A_k$;
- $L_{max}^k < L_{max}^k \leq L$.

O modelo não-linear para o Problema da Mochila Compartimentada é:

$$\text{Maximizar: } \sum_{k=1}^{q} \sum_{j=1}^{N_k} (\sum_{i \in A_k} u_i x_{ijk} - c_k) \beta_{jk} \tag{2.26}$$

$$\text{sujeito a: } \sum_{k=1}^{q} \sum_{j=1}^{N_k} (\sum_{i \in A_k} l_i x_{ijk} + S_k) \beta_{jk} \leq L \tag{2.27}$$

$$\sum_{j=1}^{N_k} x_{ijk} \beta_{jk} \leq d_i \text{ , } i \in A_k, \, k = 1, 2, \ldots, q \tag{2.28}$$

$$L_{min}^k \leq \sum_{i \in A_k} l_i x_{ijk} + S_k \leq L_{max}^k \text{ ,} k = 1, 2, \ldots, q \text{ , } j = 1, 2, \ldots, N_k \tag{2.29}$$

$$x_{ijk} \geq 0 \text{ , inteiro, } i = 1, 2, \ldots, n, \, k = 1, 2, \ldots, q, \, j = 1, 2, \ldots, N_k. \tag{2.30}$$

$$\beta_{jk} \geq 0, \text{ inteiro, } k = 1, 2, \ldots, q, \, j = 1, 2, \ldots, N_k \tag{2.31}$$

A função objetivo 2.26 maximiza o valor da utilidade menos o custo da inclusão do compartimento, a restrição 2.27 é referente à capacidade da mochila, considerando os itens que a compõem mais a perda ocasionada pela inclusão de um compartimento, a restrição 2.28 diz respeito à quantidade máxima de itens do tipo i que são permitidos na mochila, a restrição 2.29 é sobre a capacidade dos compartimentos e as restrições 2.30 e 2.31 determinam o domínio do problema.

Uma aplicação do Problema da Mochila Compartimentada é no Problema de Corte e Empacotamento, definido como uma classe de problemas que envolve a tarefa de cortar itens maiores em itens menores ou empacotar itens menores em itens maiores. Ambos os problemas são equivalentes do ponto de vista matemático. Por exemplo, se o objetivo é minimizar a quantidade de itens maiores a serem cortados em itens menores, o problema tem a mesma formulação matemática para minimizar a quantidade de itens maiores em que serão empacotados os itens menores.

A classificação dos Problemas de Corte e Empacotamento é baseada na dimensão, na seleção e na ordem dos itens, assim, em problemas de corte de estoque, são selecionados os itens maiores e demandados os itens menores, o estoque pode ser homogêneo ou heterogêneo.

Quanto à dimensão, os Problemas de Corte e Empacotamento, são categorizados em unidimensionais, isto é, somente uma dimensão é considerada no problema. Como exemplos de problemas unidimensionais, cita-se o corte de bobinas, rolos, tubos ou canos de diversos materiais. Problemas unidimensionais são tratados como os problemas de mochila apresentados. Um problema de corte de estoque pode ser também bidimensional, onde uma placa é cortada em itens menores, geralmente retangulares, por exemplo um corte de uma placa de compensado de madeira para a construção de móveis em uma indústria moveleira. Já o problema de carregamento de um contêiner é um problema categorizado como tridimensional [12].

Em Hoto [11] apresenta-se o Problema de Corte de Bobinas de Aço, que consiste em cortar de maneira mais eficaz as bobinas mestres de aço[1] para obter-se as fitas[2]. Durante a etapa de corte, o número de facas que realizam o corte das bobinas é limitado, na primeira fase são produzidas as bobinas intermediárias, já na segunda fase de corte são produzidas as fitas, maiores informações sobre o tratamento e mecanismos da produção das bobinas são obtidos em [6, 11].

As fitas possuem diversas utilidades, podendo ser aplicadas na indústria automobilística, na construção civil, em confecção de bicicletas, em tubos para caldeiras entre

[1] São as bobinas em estoque que serão submetidas ao primeiro processo de corte, são comparadas de maneira análogas com a mochila, no caso do Problema da Mochila Compartimentada.

[2] As fitas são obtidas após as bobinas mestre de aço passarem pelo primeiro processo de corte, resultando nas bobinas de aço intermediárias, que são submetidas a segunda fase de corte, na qual resulta-se nas fitas, as bobinas de aço intermediárias são comparadas de maneira análoga aos compartimentos no caso do Problema da Mochila Compartimentada, enquanto as fitas, são considerados como os itens a serem incluídos na mochila.

outras aplicações, como apresentado em Hoto [11]. Conforme sua aplicação, os tubos necessitam de dimensões particulares. Cada bobina mestre de aço pesa entre 1200Kg e 13500Kg, com larguras que variam de 1100mm à 1200mm e espessuras entre $0,90$mm à $5,10$mm, conforme apresenta Hoto [11].

Após a geração das bobinas de aço intermediárias são realizados alguns processos, como por exemplo o Processo de Corte e Laminação[3], que consiste em obter-se a espessura necessária para a aplicação da referida bobina.

O processo de corte das bobinas de aço é apresentado na Figura (2.2), primeiramente as bobinas mestre são cortadas conforme a necessidade de espessura de cada bobina, esta é a primeira fase de corte, em seguida obtém-se as bobinas intermediárias, as quais podem ser submetidas ao processo de laminação, isso depende da finalidade de cada bobina intermediária, caso a bobina passe pelo processo de laminação, esta é submetida ao processo do encruador[4]. Por fim, as bobinas intermediárias são cortadas para a obtenção das fitas, esta é a segunda fase do processo de corte, após as fitas serem obtidas, estas passam por um processo de recozimento[5]. Os processos de laminação, encruador e recozimento não influenciam significativamente os estágios de corte.

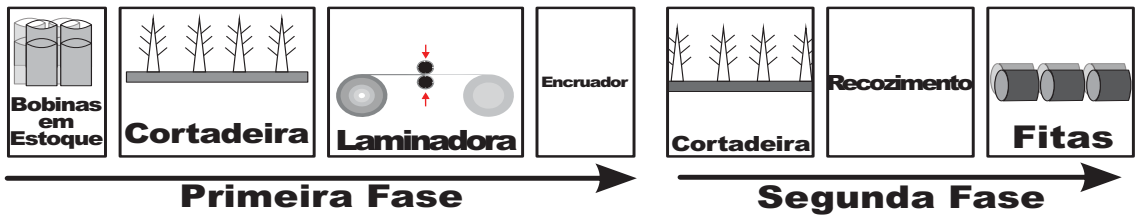

Figura 2.2: Processo de corte das bobinas de aço.

Com isto, busca-se obter a menor quantidade de sobras durante todo o processo, isto é, criar padrões de corte em cada fase buscando o maior aproveitamento das bobinas e a maior utilidade na criação das fitas.

Para a formulação matemática do problema de corte de bobinas de aço, as classes serão os conjuntos de itens que serão laminados juntos, isto é, os itens que passarão pelo processo de laminação devem estar juntos devido às restrições técnicas do maquinário utilizado no processo, por exemplo a largura mínima e máxima dos itens a serem processados. A formulação matemática dada por Hoto [11] é a apresentada pelo modelo (2.32) - (2.35):

[3]É o processo que consiste na diminuição da espessura da bobina de aço intermediária, onde as bobinas são submetidas a um processo de pressão para a definição da espessura necessária.

[4]Processo no qual são corrigidas imperfeições resultantes do processo de laminação.

[5]Processo final que tem como objetivo corrigir imperfeições resultantes durante todo o processo de corte das bobinas.

$$\text{Maximizar } \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i a_{ij} - \gamma_j\right) y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} u_i a_{ij} - \gamma_j\right) y_j \tag{2.32}$$

$$\text{sujeito a: } \sum_{j\in V_1}\left(\sum_{i\in N_1} l_i a_{ij}\right) y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} l_i a_{ij}\right) y_j \leq L \tag{2.33}$$

$$L^k_{min} \leq \sum_{i\in N_k} l_i a_{ij} \leq L^k_{max}, \text{ para } j \in V_k,\ k = 1, ..., q \tag{2.34}$$

$$a_{ij}, y_j \geq 0 \text{ e inteiros, com } i \in N = \bigcup_{k=1}^{q} N_k \text{ e } j \in V = \bigcup_{k=1}^{q} V_k \tag{2.35}$$

A função objetivo (2.32) busca maximizar a utilidade das fitas e minimizar o custo (γ_j), a restrição (2.33) estabelece que a soma das capacidades dos compartimentos criados juntamente com os itens inseridos não ultrapasse a capacidade da mochila, a restrição (2.34) refere-se à limitação da capacidade que tem cada compartimento, a restrição (2.35) determina o domínio do problema formulado, e $N = \bigcup_{k=1}^{q} N_k$ representa o conjunto de índices dos itens e $V = \bigcup_{k=1}^{q} V_k$ representa o conjunto de índices dos compartimentos viáveis gerados. Para evitar soluções triviais ao Problema da Mochila Compartimentada admita-se que $l_i \leq L^k_{max} \leq L$, para todo $i \in N_k$, $k = 1, .., q$.

Na hipótese de que cada compartimento só poder ser utilizado somente uma vez, a variável γ_j torna-se binária, assumindo o valor 1 no caso do compartimento ser utilizado e 0 caso contrário, obtendo-se assim o Problema da Mochila Compartimentada $0-1$, a formulação para este último é dada por Hoto [11], de forma análoga ao problema (2.32) - (2.35):

$$\text{Maximizar } \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i a_{ij} - \gamma_j\right) y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} u_i a_{ij} - \gamma_j\right) y_j \tag{2.36}$$

$$\text{sujeito a: } \sum_{j\in V_1}\left(\sum_{i\in N_1} l_i a_{ij}\right) y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} l_i a_{ij}\right) y_j \leq L \tag{2.37}$$

$$L^k_{min} \leq \sum_{i\in N_k} l_i a_{ij} \leq L^k_{max}, \text{ para } j \in V_k,\ k = 1, ..., q \tag{2.38}$$

$$y_j \in \{0, 1\},\ a_{ij} \geq 0 \text{ e inteiros, com } i \in N = \bigcup_{k=1}^{q} N_k \text{ e } j \in V = \bigcup_{k=1}^{q} V_k \tag{2.39}$$

Uma outra modelagem matemática para o problema da mochila compartimentada, no caso não-linear é dada por Hoto [11]:

$$\text{Maximizar } \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i a_{ij} - \gamma_j\right) y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} u_i a_{ij} - \gamma_j\right) y_j \tag{2.40}$$

$$\text{sujeito a: } \sum_{j\in V_1}\left(\sum_{i\in N_1} l_i a_{ij}\right) y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} l_i a_{ij}\right) y_j \leq L \tag{2.41}$$

$$\delta_j L^k_{min} \leq \sum_{i\in N_k} l_i a_{ij} \leq \delta_j L^k_{max}, \text{ para } j \in V_k,\ k = 1, ..., q \tag{2.42}$$

$$\sum_{k=1}^{q}\sum_{j\in V_k} a_{ij} y_j \leq d_i, \text{ com } i \in N = \bigcup_{k=1}^{q} N_k \tag{2.43}$$

$$\sum_{k=1}^{q}\sum_{j\in V_k} y_j \leq F_1 \tag{2.44}$$

$$\sum_{i\in N_k} a_{ij} \leq F_2, \text{ para } j \in V, k = 1, ..., q \tag{2.45}$$

$$\delta_j \in \{0, 1\} \text{ e } a_{ij}, y_j \geq 0 \text{ e inteiros, com } i \in N = \bigcup_{k=1}^{q} N_k \text{ e } j \in V = \bigcup_{k=1}^{q} V_k \tag{2.46}$$

A função objetivo (2.40) busca maximizar as utilidades dos itens e minimizar os custos da criação dos compartimentos Em relação às restrições, a de número (2.41) trata da capacidade da mochila, as inequações (2.42) são referentes à capacidade de cada compartimento a ser criado, a restrição (2.43) apresenta as demandas de cada item, ou seja, não é permitido exceder essa quantidade, as restrições (2.44) e (2.45) são referentes às facas do processo de corte das bobinas e por fim a restrição (2.46) apresenta o domínio de cada variável do problema.

A abordagem linear do Problema da Mochila Compartimentada é trabalhada por Hoto [9] na elaboração de novas heurísticas para a solução do problema, após isto, Cruz [5] elabora heurísticas para o modelo linear, fortalecendo a conjectura da linearidade como verdade. Por fim, Inarejos [12] propõe um modelo linear para o Problema da Mochila Compartimentada Restrito juntamente com a prova da linearidade do proposto.

Para o tratamento do modelo linear são feitas algumas observações: a primeira é sobre a quantidade de compartimentos, denotados como p_k, a serem criados para cada classe N_k, que é definida como $p_k = min\left\{F_1, \left\lfloor \frac{L}{L^k_{min}} \right\rfloor\right\}$ eliminando-se a não-linearidade do modelo proposto por Hoto [11]. Embora sejam construídos todos os p_k compartimentos, alguns não serão utilizados, isto é, não terão itens em seu interior, sendo considerados compartimentos nulos. Outra mudança é na questão da indexação dos itens, anteriormente era considerada a indexação global dos itens, nesta nova abordagem os itens são indexados localmente, isto é, os compartimentos são indexados para cada classe, enquanto a indexação das demandas, por exemplo, continua a nível global. O quesito indexação é teoricamente irrelevante, pois a alteração é somente nos índices, alterando os valores de N_k, os itens ainda continuam conectados às

suas respectivas classes.

A variável é dada por a_{ij}^k, os índices significam a quantidade de itens i da classe k no compartimento j, onde $i \in N_k$ e j também refere-se a classe k. A variável binária δ também tem o índice modificado, representada como δ_j^k, recebendo o valor 1 caso o compartimento j da classe k for não nulo e recebendo o valor 0 caso contrário.

Em cada classe $k = 1, \ldots, q$ a quantidade de compartimentos a serem incluídos na mochila não pode ser superior a $\left\lfloor \frac{L}{L_{min}^k} \right\rfloor$. Considerando-se as facas, o número máximo de compartimentos a serem construídos para cada classe será $p_k = min\{F_1, \left\lfloor \frac{L}{L_{min}^k} \right\rfloor\}$. Para a criação não são consideradas a quantidade de itens que devem ser inseridos em cada compartimento, logo, a variável de interesse será o número de itens a serem considerados em cada compartimento para compor a mochila. Disso, em teoria são criados $\sum_{k=1}^{q} p_k$ compartimentos ao todo, alguns compartimentos serão nulos, isto é, não terão itens em seu interior, e outros compartimentos possuirão o mesmo número de itens em seu interior, denominados compartimentos implícitos.

Para uma visualização, segue um exemplo em relação à criação dos p_k compartimentos para cada classe.

Exemplo 1. Considere a Tabela 2.1:

Classe k	L_{min}^k	L_{max}^k	**L**	F_1
1	7	13		
2	10	12	25	5
3	2	15		

Tabela 2.1: Valores referentes a um exemplar do Problema da Mochila Compartimentada.

Realizando os cálculos referentes aos valores de p_k para $k = 1, 2$ e 3 tem-se: $p_1 = min\{5, \left\lfloor \frac{25}{7} \right\rfloor\} = 3$; $p_2 = min\{5, \left\lfloor \frac{25}{10} \right\rfloor\} = 2$; $p_3 = min\{5, \left\lfloor \frac{25}{2} \right\rfloor\} = 5$; o que indica que no máximo podem ser construídos 3 compartimentos da classe 1; 2 compartimentos da classe 2 e para a classe 3 serão no máximo 5 compartimentos, totalizando 10 compartimentos para a mochila, alguns podem ser nulos ou possuírem a mesma quantidade de itens.

Um primeiro modelo linear é proposto por Hoto [9], contudo este modelo não é equivalente ao modelo não-linear, como mostrado por Inarejos [12], o qual propõe o modelo adotado nesta dissertação.

O modelo linear para o Problema da Mochila Compartimentada Restrita proposto por Inarejos [12] é apresentado pelas equações (2.47) - (2.53):

$$\text{Maximizar: } z = \sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} u_i a_{ij}^k \tag{2.47}$$

$$\text{sujeito a: } \sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} l_i a_{ij}^k \leq L \tag{2.48}$$

$$\delta_j^k L_{min}^k \leq \sum_{i \in N_k} l_i a_{ij}^k \leq \delta_j^k L_{max}^k \text{ ,} j = 1, 2, \ldots, p_k \text{ , } k = 1, 2, \ldots, q \tag{2.49}$$

$$\sum_{j=1}^{p_k} a_{ij}^k \leq d_i \text{ , } i \in N_k, k = 1, 2, \ldots, q \tag{2.50}$$

$$\sum_{k=1}^{q} \sum_{j=1}^{p_k} \delta_j^k \leq F_1 \tag{2.51}$$

$$\sum_{i \in N_k} a_{ij}^k \leq F_2 \text{ , } j = 1, 2, \ldots, p_k \text{ , } k = 1, 2, \ldots, q \tag{2.52}$$

$$\delta_j^k \in \{0, 1\} \text{ e } a_{ij}^k \geq 0 \text{ e inteiros, } i \in N_k, j = 1, 2, \ldots, p_k, k = 1, 2, \ldots, q \tag{2.53}$$

A função objetivo 2.47 busca maximizar a utilidade total dos itens, u_i só será somado se $i \in N_k$, a restrição 2.48 é a restrição física da mochila, l_i só será somado se $i \in N_k$, só será incluído o item i do compartimento j se são referentes a mesma classe N_k. A restrição 2.49 fornece os limitantes inferiores e superiores para os compartimentos de cada classe, só serão incluídos compartimentos não nulos, isto é $\delta_j^k = 1$, a restrição 2.50 limita a quantidade disponível de cada item i, as restrições 2.51 e 2.52 são respectivamente a faca 01 e faca 02 do processo de corte, que informam a quantidade de compartimentos não nulos que serão inseridos na mochila e a quantidade de itens que serão inseridos em cada compartimento, respectivamente, por fim a restrição 2.53 apresenta o domínio das variáveis.

A demonstração da linearidade do modelo (2.47) - (2.53) pode ser acessada em [12] e [13].

Como apresentado em Inarejos [12], o modelo linear (2.47) - (2.53) é equivalente a modelagem não-linear do Problema da Mochila Compartimentada, apresentando sempre a solução ótima do problema, isto é, o modelo linear de Inarejos [12] resulta em soluções idênticas ao modelo não-linear e a execução de forma bruta do Problema da Mochila Compartimentada na qual são geradas todas as possibilidades de soluções, e dentre o universo gerado são selecionadas as melhores para compor o interior da mochila.

2.3 Algumas Heurísticas para o Problema da Mochila Compartimentada

Heurísticas são métodos para a obtenção de soluções viáveis, isto é, soluções ótimas ou próximas da ótima de forma rápida [5]. Serão apresentadas as heurísticas dos z

melhores compartimentos e heurística das w capacidades, ambas propostas por Marques [18] e conhecidas na literatura.

No modelo não-linear (2.40) - (2.46) são considerados todos os compartimentos viáveis para a elaboração da mochila e, dentre esses, são selecionados os melhores compartimentos para a compartimentação da mochila. Num caso ideal, todos os compartimentos para a resolução exata do problema são conhecidos, porém em exemplares volumosos, torna-se computacionalmente inviável a elaboração de todos os compartimentos. São considerados apenas uma parcela destes compartimentos, dando origem às Heurísticas de Decomposição. No modelo não-linear (2.40) - (2.46) um compartimento é viável (dado pela variável a_{ij}) se satisfaz as restrições (2.54) - (2.58), independente da variável y_j.

$$\sum_{i \in N_k} l_i a_{ij} \leq L \tag{2.54}$$

$$L^k_{min} \leq \sum_{i \in N_k} l_i a_{ij} \leq L^k_{max} \text{ ou } a_{ij} = 0, \text{ para todo } i \in N \tag{2.55}$$

$$a_{ij} \leq d_i, \text{ para todo } i \in N \tag{2.56}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2 \tag{2.57}$$

$$a_{ij} \geq 0 \text{ e inteiro}, i \in N \tag{2.58}$$

A restrição (2.54) é sempre satisfeita, já que $L^k_{max} \leq L$ (verificar restrição (2.55)), portanto não é necessária, já que possui função redundante. Caso o valor de a_{ij} seja nulo para todo $i \in N$, nenhum compartimento será construído, logo a restrição (2.55) será disposta como $L^k_{min} \leq \sum_{i \in N_k} l_i a_{ij} \leq L^k_{max}$, as demais restrições são mantidas sem nenhuma alteração.

Definindo-se todos os valores possíveis para a_{ij} pode-se associar novos valores para suas utilidades e pesos (ou larguras) como se fossem "grandes itens" para cada classe, isto é:

$$U_j = \sum_{i \in N_k} u_i a_{ij} \tag{2.59}$$

A equação (2.59) define uma utilidade para cada compartimento de cada classe e

$$L_j = \sum_{i \in N_k} l_i a_{ij} \tag{2.60}$$

define uma largura (ou peso) para cada compartimento de cada classe.

Com essas novas definições pode-se formular o modelo (2.61) - (2.65) defi-

nido como Problema Mestre:

$$\text{Maximizar: } \sum_{j \in V_1} U_j y_j + \cdots + \sum_{j \in V_q} U_j y_j \tag{2.61}$$

$$\text{sujeito a: } \sum_{j \in V_1} L_j y_j + \cdots + \sum_{j \in V_q} L_j y_j \leq L \tag{2.62}$$

$$\sum_{k=1}^{q} \sum_{j \in V_k} a_{ij} y_j \leq d_i, \, i \in N \tag{2.63}$$

$$\sum_{k=1}^{q} \sum_{j \in V_k} y_j \leq F_1 \tag{2.64}$$

$$y_j \geq 0 \text{ e inteiro}, \, j \in V \tag{2.65}$$

Com as Heurísticas de Decomposição são gerados apenas alguns compartimentos viáveis (dentro de alguma condição) que satisfazem (2.56) - (2.58), após essa etapa será iniciada a solução do Problema Mestre.

Inicialmente é formulado um problema que irá designar o "melhor compartimento" $j \in V_k$ de capacidade máxima $L_{cap} \geq L^k_{max}$ para cada classe $k = 1, 2, \ldots, q$.

A Heurística de Decomposição constitui-se em gerar o "melhor compartimento" para cada classe de itens, utilizando-se do problema de Mochila Restrito apresentado pelas equações (2.66) - (2.70):

$$\text{Maximizar: } U_j = \sum_{i \in N_k} u_i a_{ij} \tag{2.66}$$

$$\text{sujeito a: } L^k_{min} \leq \sum_{i \in N_k} u_i a_{ij} \leq L_{cap} \tag{2.67}$$

$$a_{ij} \leq d_i, \, i \in N \tag{2.68}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2 \tag{2.69}$$

$$a_{ij} \geq 0 \text{ e inteiro}, \, i \in N \tag{2.70}$$

A *Heurística dos z melhores compartimentos* resume-se para cada classe k obter z "melhores compartimentos" com a Heurística de Decomposição, resolvendo o Problema de Mochila Restrito (2.66)-(2.70) e após isso, resolver o Problema Mestre (2.61)-(2.65), note que se $z = 1$ tem-se a Heurística de Decomposição.

A Heurística dos z Melhores Compartimentos é apresentada no Algoritmo 1,

proposta por Marques [18].

Algoritmo 1: Heurística dos z Melhores Compartimentos

Entrada: N, u_i, l_i, d_i, L, L^k_{max}, L^k_{min}, z

Saída: a_{ij}, y_j

Inicialização: $V = \varnothing$, $comp = 1$

para todo $k = 1, ..., q$ **faça**

  $L_{cap} = L^k_{max}$; Calcule as z melhores soluções do (2.66)-(2.70);

  para todo $contador = 1, ..., z$ **faça**

    $j = comp + contador - 1$; $j \in V_k$;

    Salve a_{ij}, U_j e L_j de acordo com a $contador - $ ***ésima*** melhor solução de (2.66)-(2.69); $comp = comp + z$

  fim

fim

Resolva (2.61)-(2.65)

Com o objetivo de não utilizar o Problema Mestre na resolução do Problema da Mochila Compartimentada, a Heurística do Melhor Compartimento define o "melhor compartimento" de cada classe k, e após isso, utilizando algum critério, como por exemplo compartimentos de maior eficiência, seleciona o "melhor compartimento" a partir deste critério para compor a compartimentação da mochila, atualizando-se a demanda e inserindo o próximo "melhor compartimento" dentre o critério em uso, o processo encerra-se quando a mochila for preenchida completamente com o "melhor dos melhores compartimentos".

A *Heurística das w capacidades*, calcula para cada classe k o melhor compartimento para w capacidades diferentes, isto é, modificando a largura máxima L_{cap}, obtém-se novas larguras para os compartimentos, totalizando $q.w$ compartimentos, que irão compor a compartimentação da mochila, o algoritmo da Heurística das w capacidades está representado no Algoritmo 2, proposta por Marques [18].

Algoritmo 2: Heurística das w Capacidades

Entrada: N, u_i, l_i, d_i, L, L^k_{max}, L^k_{min}, w

Saída: a_{ij}, y_j

Inicialização: $V = \varnothing$, $comp = 1$

para todo $k = 1, ..., q$ **faça**

  $L_{cap} = L^k_{max}$;

  para todo $j = comp, ..., comp + w - 1$ **faça**

    Resolva (2.66)-(2.69); $j \in V_k$;

    Salve a_{ij}, U_j e L_j ; $L_{cap} = L_j - 1$; $comp = comp + w$

  fim

  Resolva (2.61)-(2.65)

fim

Para a resolução dos subproblemas (2.66) - (2.70) e (2.61) - (2.65) das heurísticas dos z melhores compartimentos e das w capacidades, Marques [18] utiliza o método de

resolução exata de Gilmore e Gomory [8]. O método consiste em uma busca em profundidade numa árvore de decisões, sendo um método viável de resolução para problemas com dezenas ou centenas de itens a disposição.

Como uma forma de aperfeiçoar o algoritmo 2, Quiroga-Orozco [24] utiliza o fato do modelo linear disponibilizar a quantidade de compartimentos que podem ser utilizados (p_k), utilizando esse valor para a seleção dos compartimentos mais eficientes na resolução do problema da mochila compartimentada, produzindo uma nova heurística, determinada como p_k *Strong Capacities*.

Outras heurísticas são disponibilizadas na literatura [15, 20], no capítulo 3 serão apresentadas três novas heurísticas, que utilizam a particularidade do modelo linear (2.47) - (2.53). Serão considerados itens e classes utilizando um critério de ordenação por eficiência, que é o quociente entre a utilidade e a largura dos itens e após a elaboração dos compartimentos viáveis de cada classe.

A próxima seção apresenta um método de busca da solução exata para o Problema da Mochila, denominado *branch and bound*.

2.4 O MÉTODO *Branch and Bound*

O método *branch and bound* busca realizar uma enumeração implícita de soluções viáveis em busca da solução ótima para o problema da Mochila. O método *branch and bound* é utilizando amplamente em Problemas de Programação Linear. A apresentação dos conceitos é baseado em Chvatal [4].

Inicialmente o problema P é um problema de otimização:

$$\begin{aligned} &\text{(P) Maximizar: } f(x) \\ &\text{s.a.: } x \in X \end{aligned} \tag{2.71}$$

Onde f é a função objetivo a ser maximizada na região discreta X. Baseado na formulação de (2.71) tem-se:

Definição 2.1. *Seja:*

$$\begin{aligned} &\textit{(RP) Maximizar: } g(x) \\ &\textit{s.a.: } x \in Y \end{aligned} \tag{2.72}$$

Onde g é a função objetivo do problema (2.72) e a sua região viável discreta é dada por Y. Um problema será do tipo $relaxado$ quando satisfazer as seguintes propriedades:

- $X \subseteq Y$ *e*

- $x \in X$ *tem que* $g(x) \geq f(x)$

As próximas definições apresentam as etapas do processo de *branch and bound*.

Definição 2.2. *(Branching) É o processo de criação de novos sub-problemas a partir do problema inicial (neste caso P), sucessivamente até serem esgotadas as possibilidades de criação de novos sub-problemas. Em outra palavras, seja P_0 um problema relaxado de P (ou sub-problema) cuja região viável seja dada por X_0, então gera-se novos sub-problemas restritos em relação a P_0, isto é, $P_1, P_2, \ldots, P_n$ onde para cada sub-problema restrito desses há uma partição da região viável X_0, com isso novos sub-problemas restritos serão gerados a partir de cada P_i onde $i = 1, 2, \ldots, n$. As seguintes observações são validas:*

- *Não há eliminação de soluções viáveis que são candidatas a solução ótimas durante a execução do algoritmo;*
- *Cada solução dos sub-problemas P_i deverá estar mais próxima da solução ótima do problema P;*
- *O número de sub-problemas criados é da ordem polinomial (respeitando alguma medida);*
- *Os sub-domínios dos sub-problemas deverão ser, inicialmente, mutualmente exclusivos.*

Uma outra forma de particionar a resolução do problema P é através da fixação das variáveis de decisão do problema. Considere, por exemplo, a variável x_i, tal que $i \in \{1, 2, \ldots\}$, de P, onde os possíveis valores são o número máximo de itens que compõem o problema, seja neste caso dado por n, disso pode-se particionar o problema P em n sub-problemas $P_1, \ldots, P_n$ tal que $\bigcup_{j=1}^{n} P_j = P$. P_j e P_j corresponderá ao $j - simo$ valor da variável de decisão x_i.

Definição 2.3. *Cada sub-problema restrito gerado a partir do problema relaxado é definido como nó da árvore de enumeração, quando ocorre a fixação das variáveis, cada nó corresponderá a uma variável fixada.*

Definição 2.4. *(Bounding) A ação de avaliar os sub-problemas gerados de P é denominada como bounding. É avaliado se a solução obtida na resolução deste sub-problema em relação a solução ótima armazenada é superior ou não. Caso a avaliação demonstre que o valor atual é inferior ao valor ótimo armazenado o processo de branching neste nó é encerrado, então o nó é "podado", realizando um Teste de Sondagem no nó. A ação de branching é interrompida num Problema de Programação Inteira nos seguintes casos:*

1. *Por infactibilidade da solução: o sub-problema restrito gerado é infactível;*

2. *Por otimalidade da solução: a solução do problema relaxado é inteira;*

3. *Por qualidade: os valores de qualquer solução viável do problema relaxado é pior que a solução viável atual para todos os próximos nós.*

No caso de fixação das variáveis a ação de bounding ocorre através da análise de Limitantes Superiores.

Definição 2.5. *(Árvore de Decisão) O método de Branching (separação) e Bounding (avaliação) pode ser visualizado como uma árvore de decisões para o problema P, onde o sub-problema P_0 é tido como o nó principal (ou raiz), os nós sucessores são relacionados a cada sub-problema P_i gerado a partir de P_0. Quando ocorre a fixação das variáveis, cada nó representa uma solução e quando ocorre um branching significa uma solução viável para o problema. A Figura 2.3 exibe um exemplo da árvore de decisões.*

Quando em um nó são gerados dois ou mais nós, é dito que o nó foi ramificado. Nós que ainda não foram ramificados são classificados como nós abertos e os demais são os nós fechados.

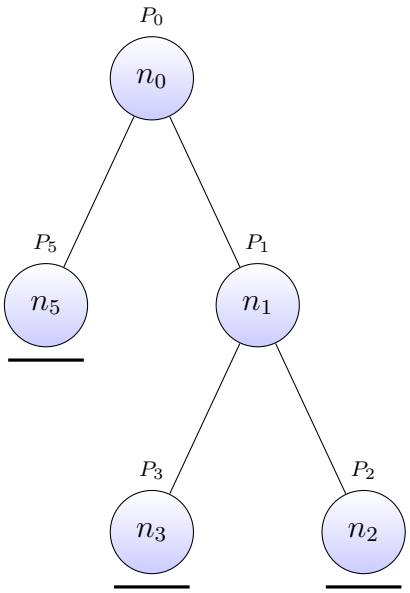

Figura 2.3: Exemplo de Árvore de decisão *Branch and Bound*.

Proposição 2.6. *Seja x^* uma solução ótima para o Problema da Mochila Irrestrito então $L - \sum_{i=1}^{n} l_i x_i^* < l_r$ para qualquer $r \in \{1, \ldots, n\}$.*

Demonstração. Seja x^* uma solução ótima para o Problema da Mochila Irrestrito onde tem-se $\sum_{i=1}^{n} l_i x_i^* < L$, assim pode-se acrescentar em uma unidade qualquer o valor de x_r onde $r \in \{1, 2, \ldots, n\}$ disso $l_1 x_1^* + \cdots + l_r(x_r^* + 1) + \cdots + l_n x_n^* > L$ o que implica em $L - \sum_{i=1}^{n} l_i x_i^* < l_r$ para qualquer $r \in \{1, \ldots, n\}$. □

Definição 2.7. *Uma solução é definida como sensível se* $L - \sum_{i=1}^{n} l_i x_i < l_r$ *para qualquer* $r \in \{1, \ldots, n\}$.

Definição 2.8. *A eficiência associada a variável* x_j *é o quociente entre utilidade e largura associada ao item* j*, isto é* $\frac{u_j}{l_j}$.

O próximo exemplo ilustra a construção da árvore de decisões e foi retirado de Chvatal [4].

Exemplo 2.

$$\text{Maximizar: } 4x_1 + 5x_2 + 5x_3 + 2x_4 \tag{2.73}$$

$$\text{sujeito a: } 33x_1 + 49x_2 + 51x_3 + 22x_4 \leq 120 \tag{2.74}$$

$$x_1, x_2, x_3, x_4 \geq 0 \text{ e inteiro} \tag{2.75}$$

Inicialmente é necessário a ordenação das variáveis em ordem decrescente dos itens através de sua eficiência associada, isto é $\left(\frac{u_1}{l_1} > \frac{u_2}{l_2} > \cdots > \frac{u_n}{l_n}\right)$ e realizar a enumeração das soluções sensíveis.

Neste exemplo a ordenação necessária já esta realizada, isto é $\left(\frac{4}{33} > \frac{5}{49} > \frac{5}{51} > \frac{2}{22}\right)$. A apresentação de todas as soluções sensíveis esta na Figura (2.4), ao todo são 13 soluções sensíveis.

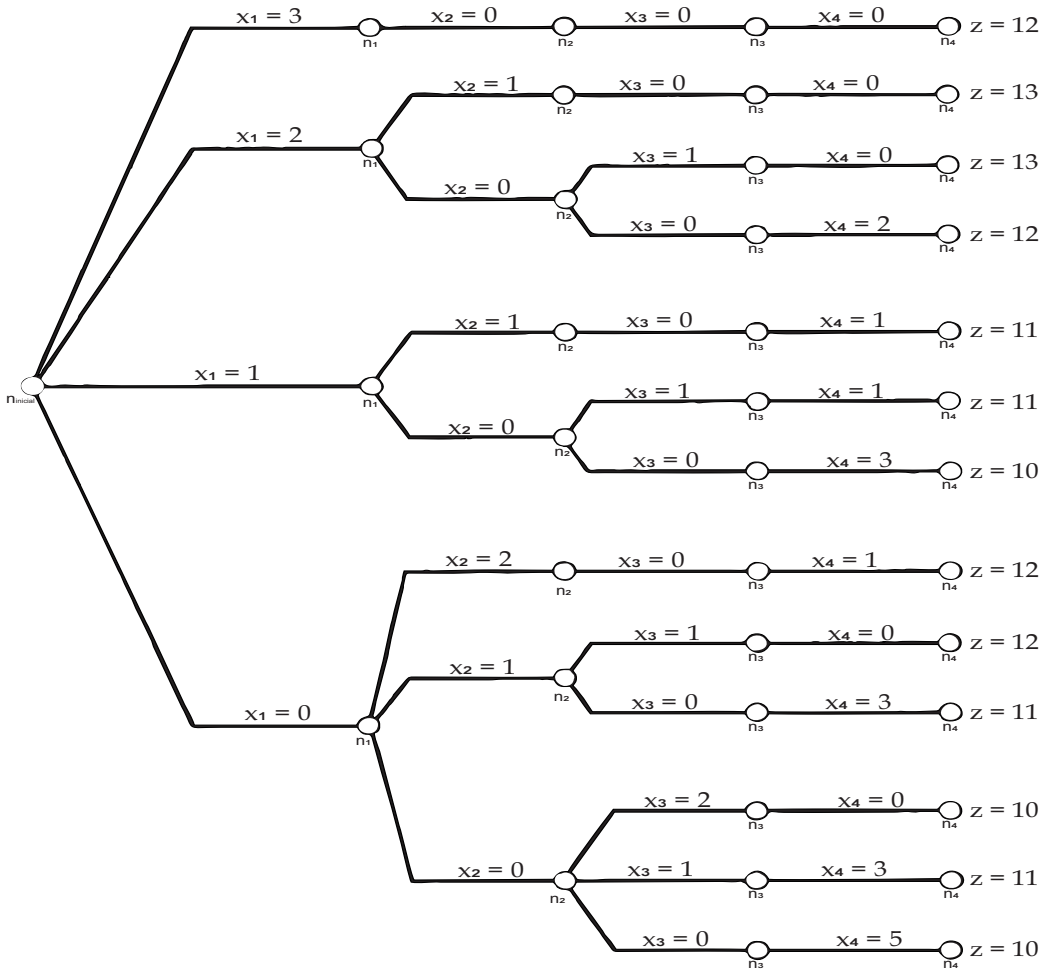

Figura 2.4: Árvore para o Exemplo 2 sem Podas.

Do ramo inicial da árvore são fixadas as variáveis da esquerda para a direita (por ordem de eficiência), e em cada variável é definido um nó. O desenvolvimento dos ramos é de cima para baixo, respeitando a ordem das variáveis fixadas.

O processo de *branching* inicia-se adicionando $\left\lfloor \frac{L}{l_1} \right\rfloor$ itens em x_1, isto é a quantidade máxima de itens que é possível inserir de x_1 no interior da mochila. Para a próxima variável fixada x_2 são inseridos $\left\lfloor \frac{L-\sum_{i=1}^{j-1} l_i x_i}{l_j} \right\rfloor$ itens, realizando-se sucessivamente até x_n, como exemplo, o primeiro ramo de (2.73) é dado através de $x_1 = \lfloor \frac{120}{33} \rfloor = 3$ e $x_2 = x_3 = x_4 = 0$.

Após a geração do primeiro ramo, armazena-se a solução corrente e realiza-se o processo de volta, denominado *backtracking*, no qual do último nó até o nó x_k onde $x_k \neq 0$,

após definir qual é o nó x_k subtrai-se uma unidade deste nó, $x_k = x_k - 1$ e desenvolve-se um novo ramo com o novo valor de x_k, neste exemplo tomando $k = 1$ e $x_1 = 3-1 = 2$ determinou-se $x_2 = 1$ e $x_3 = x_4 = 0$. Após a realização desse novo ramo, novamente é aplicada a etapa de *backtracking* e *branching*, o processo é encerrado quando $k = 1$ e $x_1 = 0$.

A geração da árvore com todas as soluções sensíveis possui um alto custo computacional para valores elevados de n, para um problema com n itens, no pior caso, será necessário o desenvolvimento de $2^n - 2$ ramos, o qual corresponde ao conjunto das partes de n, tornando-se praticamente inviável a utilização para valores altos de n. Para solucionar esse alto custo computacional é realizado o processo de *bounding* no decorrer da execução, com o objetivo de realizar o mínimo de esforço computacional. Para isso é necessário realizar estimativas a respeito do ramo em análise, antes de explorá-lo, para verificar qual o valor que este pode resultar na função objetivo.

Considere que um ramo arbitrário dado por $\hat{x}_1, \ldots, \hat{x}_k, \ldots, \hat{x}_n$ com $1 \leq k \leq n-1$ seja a melhor solução determinada até o momento $(\hat{z})$, realizando o processo de *backtracking* e $\hat{x}_k = \hat{x}_k - 1$ realizando-se uma nova ramificação do nó x_k, onde é criado um novo ramo com $\hat{x}_1, \ldots, \hat{x}_k - 1, \hat{x}_{k+1}, \ldots, \hat{x}_n$, com isso, busca-se estimar o valor $\overline{z}$ de modo que se o valor estimado de $\overline{z}$ for superior a melhor solução até o momento $\hat{z}$, desenvolva o ramo.

Para realizar a estimativa é necessário o cálculo de um limitante superior, denominado M, utilizando somente valores conhecidos, isto é $\hat{x}_{k+1}, \ldots, \hat{x}_n$. Se ocorrer que $M \leq \hat{z} \leq \overline{z}$ então realiza-se a poda do ramo interrompendo seu desenvolvimento, já que este não gerará uma solução melhor que a atual.

Um limitante superior para o ramo é proposto por Gilmore e Gomory [8]:

$$\begin{aligned} \bar{z} &= \sum_{i=1}^{n} u_i x_i \\ &= \underbrace{\sum_{i=1}^{k-1} u_i \hat{x}_i + u_k(\hat{x}_k - 1)}_{\text{Valores Conhecidos}} + \underbrace{\sum_{i=k+1}^{n} u_i \bar{x}_i}_{\text{Valores Desconhecidos}} \end{aligned} \tag{2.76}$$

Os valores que são conhecidos são $\hat{x}_1, \hat{x}_2, ..., \hat{x}_k - 1$ e os desconhecidos $\bar{x}_{k+1}, ..., \bar{x}_n$, onde o valor a ser estimado é $\sum_{i=k+1}^{n} u_i \bar{x}_i$. Como a restrição da mochila é uma restrição necessária no desenvolvimento do nó, tem-se $\sum_{i=1}^{n} l_i \bar{x}_i \leq L$. Utilizando esta restrição:

$$\sum_{i=1}^{n} l_i \bar{x}_i = \underbrace{\sum_{i=1}^{k-1} l_i \hat{x}_i + l_k(\hat{x}_k - 1)}_{\text{Valores Conhecidos}} + \underbrace{\sum_{i=k+1}^{n} l_i \bar{x}_i}_{\text{Valores Desconhecidos}} \leq L \tag{2.77}$$

De (2.76) e (2.77) tem-se:

$$\sum_{i=k+1}^{n} u_i\bar{x}_i = \sum_{i=k+1}^{n} \Big(\frac{l_i}{l_i}\Big)u_i\bar{x}_i = \sum_{i=k+1}^{n} \Big(\frac{u_i}{l_i}\Big)l_i\bar{x}_i \tag{2.78}$$

Note que:

$$\sum_{i=k+1}^{n} \Big(\frac{u_i}{l_i}\Big)l_i\bar{x}_i = \Big(\frac{u_{k+1}}{l_{k+1}}\Big)l_{k+1}\bar{x}_{k+1} + \Big(\frac{u_{k+2}}{l_{k+2}}\Big)l_{k+2}\bar{x}_{k+2} + \cdots + \Big(\frac{u_n}{l_n}\Big)l_n\bar{x}_n \leq \alpha \sum_{i=k+1}^{n} \Big(\frac{u_i}{l_i}\Big)l_i\bar{x}_i \tag{2.79}$$

Onde $\alpha = max\Big\{\frac{u_{k+1}}{l_{k+1}}, \ldots, \frac{u_n}{l_n}\Big\}$. Continuando (2.79) e utilizando (2.77):

$$\alpha \sum_{i=k+1}^{n} \Big(\frac{u_i}{l_i}\Big)l_i\bar{x}_i \leq \alpha\Big(L - \Big(\sum_{i=1}^{k-1} l_i\hat{x}_i - l_k(\hat{x}_k - 1)\Big)\Big) \tag{2.80}$$

Ou seja:

$$\begin{aligned} \bar{z} &= \sum_{i=1}^{n} u_i x_i \\ &= \sum_{i=1}^{k-1} u_i\hat{x}_i + u_k(\hat{x}_k - 1) + \underbrace{\sum_{i=k+1}^{n} u_i\bar{x}_i}_{\leq \alpha\left(L - \left(\sum_{i=1}^{k-1} l_i\hat{x}_i - l_k(\hat{x}_k - 1)\right)\right)} \\ &\leq M \end{aligned} \tag{2.81}$$

Onde M é um limitante superior para o ramo no qual não depende de nenhum valor desconhecido, dado por:

$$M = \sum_{i=1}^{k-1} u_i\hat{x}_i + u_k(\hat{x}_k - 1) + \alpha\Big(L - \Big(\sum_{i=1}^{k-1} l_i\hat{x}_i - l_k(\hat{x}_k - 1)\Big)\Big) \tag{2.82}$$

Com $\alpha = max\Big\{\frac{u_{k+1}}{l_{k+1}}, \ldots, \frac{u_n}{l_n}\Big\}$.

Ao calcular o limitante superior M para um nó, os nós sucessores a ele não necessitam o cálculo de seus limitantes posteriores. De fato, seja M_j e M_{j-1} onde j representa a posição do nó, com $\alpha = \frac{u_{k+1}}{l_{k+1}}$.

$$M_j = \sum_{i=1}^{k-1} u_i\hat{x}_i + u_k(\hat{x}_k - j) + \frac{u_{k+1}}{l_{k+1}}\Big(L - \Big(\sum_{i=1}^{k-1} l_i\hat{x}_i - l_k(\hat{x}_k - j)\Big)\Big) \tag{2.83}$$

$$M_{j-1} = \sum_{i=1}^{k-1} u_i \hat{x}_i + u_k(\hat{x}_k - (j-1)) + \frac{u_{k+1}}{l_{k+1}}\Big(L - \Big(\sum_{i=1}^{k-1} l_i \hat{x}_i - l_k(\hat{x}_k - (j-1))\Big)\Big) \quad (2.84)$$

Subtraindo (2.84) de (2.83), tem-se:

$$\begin{aligned} M_j - M_{j-1} &= u_k(\hat{x}_k - j) - u_k(\hat{x}_k - (j-1)) + \frac{u_{k+1}}{l_{k+1}}\Big(- l_k(\hat{x}_k - j) + l_k(\hat{x}_k - (j-1))\Big) \\ &= -u_k j + u_k j - u_k + \frac{u_{k+1}}{l_{k+1}}\Big(l_k j - l_k j + l_k\Big) \\ &= -u_k + \frac{u_{k+1}}{l_{k+1}}\Big(l_k\Big) \\ &= l_k\Big(\frac{u_{k+1}}{l_{k+1}}\Big) - u_k\Big(\frac{l_k}{l_k}\Big) \\ &= l_k\Big(\frac{u_{k+1}}{l_{k+1}} - \frac{u_k}{l_k}\Big) \\ &\leq 0 \end{aligned} \quad (2.85)$$

Isso advém do fato da ordenação dos itens ser em ordem decrescente. Com isso, conclui-se que:

$$M_j \leq M_{j-1} \quad (2.86)$$

Assim não é necessário a exploração dos demais ramos sucedentes ao ramo j. A Figura (2.5) apresenta o Exemplo 2 com a utilização do limitante superior durante a elaboração da árvore de soluções sensíveis, gerando a solução ótima com $x_1 = 2$, $x_2 = 1$, $x_3 = x_4 = 0$.

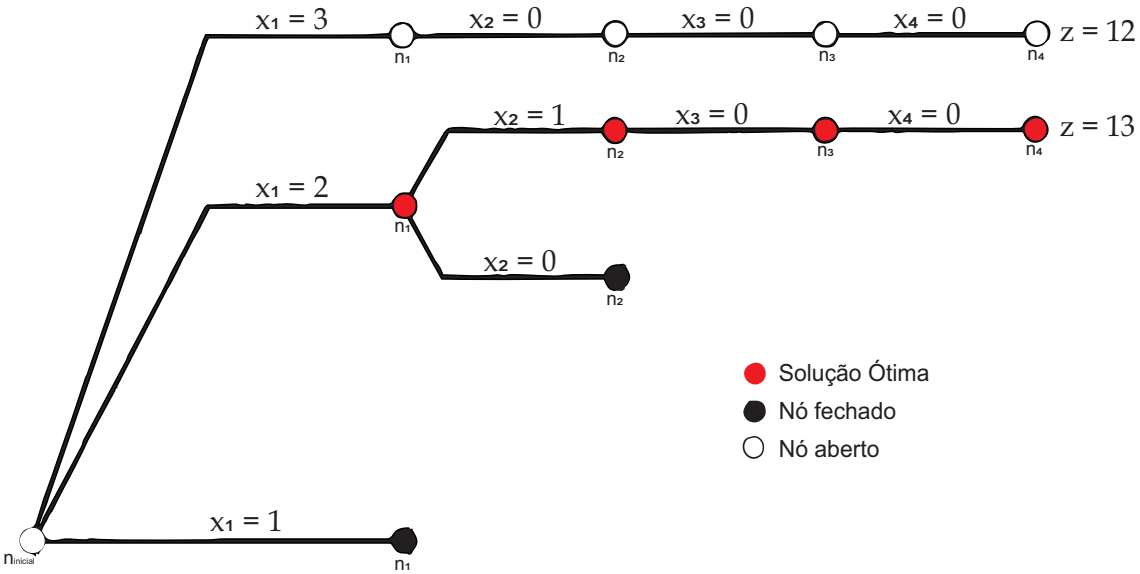

Figura 2.5: Árvore para o exemplo 2 após realização das Podas.

2.5 MÉTODO DE BRANCH AND BOUND DE MARTELLO E TOTH

O algoritmo 3 descreve um método *branch and bound* para resolução do problema da mochila. Esse algoritmo realiza uma busca pela solução ótima analisando os "ramos" da árvore de busca e realizando "podas" baseadas em limitantes. A elaboração inicial do algoritmo proposto por Martello e Toth [21] é para a resolução do seguinte problema de mochila irrestrito:

$$\text{Maximizar z} = \sum_{i=1}^{n} u_i xi \tag{2.87}$$

$$\text{Sujeito a:} \sum_{i=1}^{n} l_i x_i \leq L \tag{2.88}$$

$$x_i \geq 0 \text{ e inteiros, com } i \in N = \{1, \ldots, n\} \tag{2.89}$$

2.5.1 Limitantes Superiores

Para que um algoritmo do tipo *branch and bound* seja eficaz é necessário que exista um limitante superior igual ou minimamente maior que a solução ótima do problema, assim [21] apresenta os seguintes limitantes superiores levando em consideração que os itens estão organizados por ordem decrescente de eficiência.

A solução ótima para o problema (2.87) - (2.89) no caso de relaxação contínua, é $\overline{x}_1 = \frac{L}{l_1}$, $\overline{x}_j = 0$ para todo $j = 2, \ldots, n$ o que apresenta o limitante superior trivial

$$U_0 = \left\lfloor L \frac{u_1}{l_1} \right\rfloor$$

.

Supondo que $\overline{x}_1 \leq \left\lfloor \frac{L}{l_1} \right\rfloor$ pertence a uma solução inteira, então a solução contínua será dada por:

$$\overline{x}_1 = \left\lfloor \frac{L}{l_1} \right\rfloor \tag{2.90}$$

$$\overline{x}_j = 0, \forall j = 3, \ldots, n \tag{2.91}$$

$$\overline{x}_2 = \frac{\overline{L}}{l_2} \tag{2.92}$$

$$\text{onde } \overline{L} \text{ é dado por: } \overline{L} = L(\mod l_1) \tag{2.93}$$

Desta forma, um novo limitante é exposto $U_1 = \left\lfloor \frac{L}{l_1} \right\rfloor u_1 + \left\lfloor \overline{L} \frac{u_2}{l_2} \right\rfloor$ assim tem-se:

$$U_2 = max(U^0, U^1) \tag{2.94}$$

onde o valor do item crítico, que é o primeiro item onde não é mais possível inserir na mochila

baseado na ordem decrescente das eficiências, será sempre 2:

$$z' = \left\lfloor \frac{L}{l_1} \right\rfloor u_1 + \left\lfloor \frac{\overline{L}}{l_2} \right\rfloor u_2 \tag{2.95}$$

$$L' = \overline{L}(\mod l_2) \tag{2.96}$$

$$U^0 = z' + \left\lfloor L' \frac{u_3}{l_3} \right\rfloor \tag{2.97}$$

$$U^1 = z' + \left\lfloor u_2 - (l_2 - L') \frac{u_1}{l_1} \right\rfloor \tag{2.98}$$

Analisando o fato de que o item crítico será 2, e que U^1 é um limitante superior se pelo menos $\left\lfloor \frac{\overline{L}}{l_2} \right\rfloor + 1$ itens do tipo 2 são escolhidos, e isso só é possível se ao menos $\left\lceil \frac{(l_2 - L')}{l_1} \right\rceil$ itens do tipo 1 são removidos da solução correspondente a z' e então $L' + \left\lceil \frac{(l_2 - L')}{l_1} \right\rceil l_1$ unidades são liberadas para os itens do tipo 2, obtendo um novo limitante, quando realizada a mudança em U^1:

$$\overline{U}^1 = z' + \left\lfloor \left(L' + \left\lceil \frac{l_2 - L'}{l_1} \right\rceil l_1 \right) \frac{u_2}{l_2} - \left\lceil \frac{l_2 - L'}{l_1} \right\rceil u_1 \right\rfloor \tag{2.99}$$

Portanto, $\overline{U}^1 \leq U^1$, uma vez que "movendo" um número maior de itens do tipo 1 para itens do tipo 2, com menores eficiências, o que prova o seguinte teorema.

Teorema 2.9.

$$U_3 = max(U^0, \overline{U}^1) \tag{2.100}$$

é um limitante superior para o problema da mochila (2.87)-(2.89) e para qualquer instância $U_3 \leq U_2$, onde U^0 e $\overline{U}^1$ são definidos por (2.93), (2.95)-(2.97) e (2.99).

Para uma visualização do teorema apresentado, segue um exemplo presente em [21]:

Exemplo 3. Considere a seguinte instância do problema (2.87)-(2.89):

- n = 3;
- (u_j) = $(20, 5, 1)$;
- (l_j) = $(10, 5, 3)$;
- L = 39;

Os limitantes superiores serão:

$U_0 = 78$;
$U_1 = 60 + \left\lfloor 9\frac{5}{5} \right\rfloor = 69$;
$U^0 = 65 + \left\lfloor 4\frac{1}{3} \right\rfloor = 66$;

$U^1 = 65 + \left\lfloor 5 - 1\frac{20}{10} \right\rfloor = 68;$
$U_2 = 68;$
$\overline{U}^1 = 65 + \left\lfloor \left(4 + \left\lceil \frac{1}{10} \right\rceil 10\right)\frac{5}{5} - \left\lceil \frac{1}{10} \right\rceil 20 \right\rfloor = 59;$
$U_3 = 66;$

2.5.2 O algoritmo MTU1

O algoritmo MTU1 consiste em um método *branch and bound* no qual a árvore de soluções é mais ágil [21], quando comparada com o método de Gilmore e Gomory [8]. Para a utilização do método MTU1 os itens devem ser classificados inicialmente por ordem decrescente de eficiência, isso faz com que os limitantes adotados pelo método sejam exatos e consequentemente o próprio método, sendo assim, é assumida tal ordenação.

A árvore de enumeração implícita do MTU1 possui $n + 1$ níveis de busca, sendo o nó raiz (ou nó inicial) o que representa todo o espaço de busca do método. O nível inicial contém $\left\lfloor \frac{L}{l_1} + 1 \right\rfloor$ nós, onde cada nó representa um espaço de busca no qual contém $\left\lfloor \frac{L}{l_1} - 1 \right\rfloor$ itens do tipo 1 sucessivamente até não conter nenhum item do tipo 1. No segundo nível, estão contidos os itens do tipo 2, e assim por diante até o nível n. A partir do segundo nível a quantidade de nós é variada, quando não há nenhum item do tipo 1 e a mochila esta vazia, existem $\left\lfloor \frac{L}{l_2} \right\rfloor$ itens do tipo 2, e quando há itens do tipo 1 disponíveis são $\left\lfloor \frac{L \mod l_1}{l_2} + 1 \right\rfloor$ nós do segundo nível que são combinados com os itens do tipo 1, do primeiro nível.

Para a elaboração da árvore, são analisados os nós iniciais (de maior eficiência) em direção aos nós de menor eficiência associada (nós finais). Esta ordenação proporciona um método mais eficiente, já que as melhores soluções estão associadas aos itens de maior eficiência.

Como é um método *branch and bound*, em cada nó visitado o método MTU1 calcula e analisa um limitante superior para os nós restantes abaixo do nó em análise, se esse limite superior for igual ou menor que o limite inferior global, o método MTU1 irá ignorar o restante da subárvore e voltará ao nó inicial. Esses limitantes superiores consistem em uma solução com os itens já visitados e um pseudo-item com largura igual ao gap de capacidade e eficiência igual a eficiência do próximo item [2]. A análise e atualização dos limites inferiores é constante.

Para reduzir o tamanho da árvore, se o espaço na capacidade deixada por um nó é menor que o $\bar{l}_{min}$ então este é um nó "folha", assim não há necessidade de explorar os demais itens. Cada nó representa uma solução única (que é o caminho entre o nó principal até o nó final), devido a ordenação utilizada, para cada nó (sendo folha ou não) é possível definir qual o espaço de busca que ainda falta visitar. Assim, não é necessária uma numeração explícita, baseado no nó/solução atual é possível saber quais soluções ainda não foram tentadas.

A seguir é apresentado um exemplo do funcionamento do método MTU1, que

está disponível em Martello e Toth [21].

Exemplo 4. Considere a seguinte instância do problema (2.87)-(2.89):

- n = 7;
- (u_j) = $(20, 39, 52, 58, 31, 4, 5)$;
- (l_j) = $(15, 30, 41, 46, 25, 4, 5)$;
- L = 101;

A figura (2.6) apresenta a árvore de decisão elaborada pelo método MTU1.

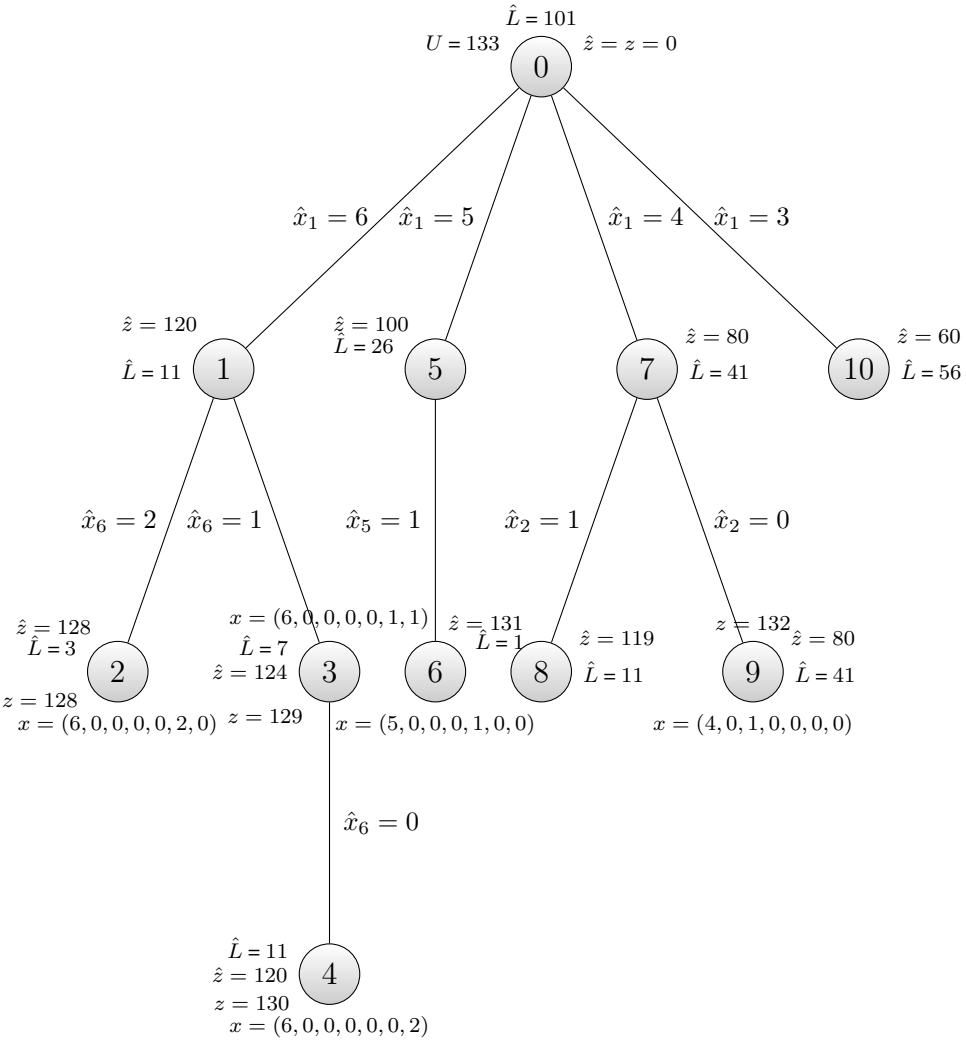

Figura 2.6: Exemplo de Árvore de decisão *Branch and Bound* do exemplo (4).

Os algoritmos 3, 4, 5 e 6 apresentam o método de *branch and bound* MTU1 proposto por Martello e Toth [21]. O algoritmo esta dividido para uma melhor visualização do

leitor.

Algoritmo 3: Algoritmo MTU1 (Etapas 1 e 2)

Entrada: l_i, u_i, L e n

Saída: x_j e Z

1. Inicio

z = 0;

$\hat{z}$ = 0;

$\hat{L}$ = L;

u_{n+1} = 0;

l_{n+1} = ∞;

para todo $k = 1 \ldots n$ **faça**
| $\hat{x}_k = 0$;

fim

Calcule o limitante superior $U = U_3$;

para todo *k = n... 1* **faça**
| Calcule $m_k = min\{l_i : i > k\}$;

fim

j = 1;

2. Construindo nova solução corrente

enquanto $l_j > \hat{L}$ **faça**
| **se** $z \geq \hat{z} + \lfloor \frac{\hat{L}u_{j+1}}{l_{j+1}} \rfloor$ **então**
| | Vá para 5;
| **fim**
| **senão**
| | j = j + 1;
| **fim**

fim

$y = \lfloor \frac{\hat{L}}{l_j} \rfloor$;

$v = \lfloor \frac{(\hat{L} - yl_j)u_{j+1}}{u_{j+1}} \rfloor$;

se $z \geq \hat{z} + yu_j + u$ **então**
| Vá para 5;

fim

se *v = 0* **então**
| Vá para 4;

fim

Fonte: Martello e Toth [21].

Algoritmo 4: Continuação (Etapas 3 e 4) - Algoritmo MTU1

3. Salve a solução corrente

$\hat{L} = \hat{L} - yl_j$;

$\hat{z} = \hat{z} + yu_j$;

$\hat{x}_j = y$;

$j = j + 1$;

se $\hat{L} \geq m_{j-1}$ **então**
| Vá para 2.

fim

se $z \geq \hat{z}$ **então**
| Vá para 5.

fim

y = 0;

4. Atualize a melhor solução até o momento

$z = \hat{z} + yu_j$;

para todo *k = 1 . . . j-1* **faça**
| $x_k = \hat{x}_k$;

fim

$x_j = y$; **para todo** *k = j+1. . . n* **faça**
| $x_k = 0$;

fim

se z= U **então**
| Retorne;

fim

Fonte: Martello e Toth [21].

Algoritmo 5: Continuação (Etapa 5) - Algoritmo MTU1

5. *Backtrack*

Determine i = max$\{k < j : \hat{x}_k > 0\}$;

se *Não encontrar* i **então**
  Retorne;

fim

$\hat{L} = \hat{L} + l_i$;

$\hat{z} = \hat{z} - u_i$;

$\hat{x}_i = \hat{x}_i - 1$;

se $z \geq \hat{z} + \lfloor \frac{\hat{L}u_{i+1}}{l_{i+1}} \rfloor$ **então**
  Removendo todos os itens do tipo i:
  $\hat{L} = \hat{L} + l_i\hat{x}_i$;
  $\hat{z} = \hat{z} - u_i\hat{x}_i$;
  $\hat{x}_i = 0$;
  $j = i$;
  Vá para 5.

fim

j = i + 1;

se $\hat{L} - l_i \geq m_i$ **então**
  Vá para 2.

fim

h = i;

Fonte: Martello e Toth [21].

Algoritmo 6: Continuação (Etapa 6) - Algoritmo MTU1

6. Tente trocar um item do tipo i com item do tipo h

$h = h + 1$; **se** $z \geq \hat{z} + \lfloor \frac{\hat{L}u_h}{l_h} \rfloor$ **então**
| Vá para 5.

fim

se $l_h = l_i$ **então**
| Vá para 6.

fim

se $l_h \geq l_i$ **então**
| **se** $w_h > \hat{L}$ *ou* $z \geq \hat{z} + u_h$ **então**
| | Vá para 6.
| **fim**
| $z = \hat{z} + u_h$; **para todo** $k = 1 \ldots n$ **faça**
| | $x_k = \hat{x}_k$;
| **fim**
| x_h = 1; **se** $z = U$ **então**
| | Retorne;
| **fim**
| i = h;
| Vá para 6.

fim

senão
| **se** $\hat{L} - l_h < m_{h-1}$ **então**
| | Vá para 6.
| **fim**
| j = h;
| Vá para 2.

fim

Fonte: Martello e Toth [21].

2.5.3 O algoritmo MTU2

O algoritmo MTU2 também é proposto por Martello e Toth [21] e tem como objetivo melhorar o tempo de execução do algoritmo MTU1 para exemplares grandes (por exemplo, exemplares com mais de 250.000 itens disponíveis para a mochila). O algoritmo MTU2 utiliza o algoritmo MTU1 internamente, porém com os itens de maior eficiência. A elaboração do método MTU2 foi motivada após a observação que a maioria das soluções dos exemplares utilizados por [21] continham apenas os itens de maior eficiência, e que para algumas instâncias o custo computacional para organizar todos os itens disponíveis em ordem decrescente de eficiência exigia um esforço computacional desnecessário.

A explicação vem do fato de que na apresentação do MTU1 na seção (2.5.2) os itens são classificados através de sua eficiência e são utilizados os itens de maior eficiência inicialmente, caso a solução utilize somente os primeiros itens (de maior eficiência) não é necessário uma exploração de todos os nós restantes, Martello e Toth [21] conclui que o tempo classificando quaisquer outros itens era desnecessário.

Para solucionar esse esforço computacional desnecessário e resolver instâncias ainda maiores, Martello e Toth [21] propõem o algoritmo MTU2, no qual é baseado no conceito de "problema central".

Suponha sem perda de generalidade que os itens estão ordenados por ordem decrescente de eficiência, $\frac{u_j}{l_j} > \frac{u_{j+1}}{l_{j+1}}$ para todo $j = 1, \ldots, n-1$, define-se o "centro" como:

$$C = \{1, 2, \ldots, \overline{n} \equiv \max\{j : x_j > 0\}\}$$

E o "Problema Central" é definido como:

$$\text{Maximizar z} = \sum_{j \in C} u_j xj \tag{2.101}$$

$$\text{Sujeito a:} \sum_{j \in C} l_j x_j \leq L \tag{2.102}$$

$$x_j \geq 0 \text{ e inteiros, com } j \in C \tag{2.103}$$

Se o valor de $\overline{n}$ é conhecido inicialmente, facilmente resolve-se o problema (2.87)-(2.89) atribuindo $x_j = 0$ para todo j tal que $\frac{u_j}{l_j} < \frac{u_{\overline{n}}}{l_{\overline{n}}}$, onde $C = \{j : \frac{u_j}{l_j} \geq \frac{u_{\overline{n}}}{l_{\overline{n}}}\}$, com isto basta resolver (2.101)- (2.103) com os itens de C utilizando o algoritmo MTU1, por exemplo.

O valor $\overline{n}$ usualmente não é conhecido inicialmente, porém pode ser determinado assumindo um valor inicial de $\overline{n}$ e resolvendo o "Problema Central" associado a este valor $\overline{n}$, caso a solução obtida seja igual a um limitante superior, o ótimo foi encontrado, caso contrário, acrescenta-se itens a C e realiza-se uma redução dos valores, até que o valor seja igual a um limitante superior ou C seja igual a seu complementar (os itens que não estão em C).

Os limitantes do método MTU2, são análogos aos da seção (2.5.1), com isso, assuma $U_q(j)$ um limitante superior com $q = 1, 2$ ou 3, com o adicional do item j estar incluso no cálculo deste limitante, onde j pertence ao complementar do conjunto C em análise. Se $U_q(j) \leq z$ (z é a solução do "Problema Central") então o $x_j = 0$. Este processo é realizado até que o valor seja igual a um limitante superior ou C seja igual a seu complementar (os itens que não estão em C).

Um outro fator que condensa a quantidade de itens que são selecionados para a resolução do "Problema Central" é a aplicação do conceito de *itens dominados* por [21].

Definição 2.10 (Itens Dominados)**.** *Dada uma instância do problema (2.87)-(2.89) e o conjunto de índices N relativos aos itens, o item do tipo $k \in N$ é dito dominado se o valor ótimo não é*

alterado removendo-o de N.

O seguinte teorema é apresentado:

Teorema 2.11. *Dada uma instância do problema (2.87)-(2.89) e um item do tipo* k*, se existir um item do tipo* j *tal que*

$$\left\lfloor \frac{l_k}{l_j} \right\rfloor u_j \geq u_k \tag{2.104}$$

então k *é dominado.*

Demonstração. Dada uma solução viável com $x_k = \alpha > 0$ e $x_j = \beta$, uma solução melhor pode ser obtida fazendo $x_k = 0$ e $x_j = \beta + \left\lfloor \frac{l_k}{l_j} \right\rfloor \alpha$. De fato, a nova solução é viável, desde que $\left\lfloor \frac{l_k}{l_j} \right\rfloor \alpha l_j \leq \alpha l_k$ e a utilidade gerada pelo item do tipo j na nova solução não é menor que a produzida pelos itens do tipo j e k na solução viável, desde que, (2.104), $\left\lfloor \frac{l_k}{l_j} \right\rfloor \alpha u_j \geq \alpha u_k$. □

Corolario 2.12. *Todos os itens dominados podem ser eliminados do "Problema Central" seguindo:*

1. *Ordene os itens em ordem decrescente de eficiência com* $l_j \leq l_{j+1}$*;*

2. ***para:*** $j = 1$ *até* $|C| - 1$ ***faça****:*
   para: $k = j + 1$ *até* $|C|$ ***faça****:*
   se: *(2.104) ocorre* ***então****:* $C = C \backslash \{k\}$

Demonstração. A condição (2.104) nunca ocorrerá se $\frac{u_j}{l_j} < \frac{u_k}{l_k}$ ou $\frac{l_k}{l_j}$. □

O mesmo exemplo (4) é apresentado e com ele a nova árvore de decisão, agora após realizar o corolário (2.12).

Tomando $\nu = 4$, o "Problema Central" é dado por:

(u_j) = $(20, 39, 52, 58)$;

(l_j) = $(15, 30, 41, 46)$;

Aplicando o corolário (2.12) temos que os itens do tipo 2 e 4 são itens dominados, assim, aplica-se o método MTU1 nos itens restante, sendo:

(u_j) = $(20, 52)$;

(l_j) = $(15, 41)$;

Obtendo a seguinte árvore de decisão, dada pela figura 2.7:

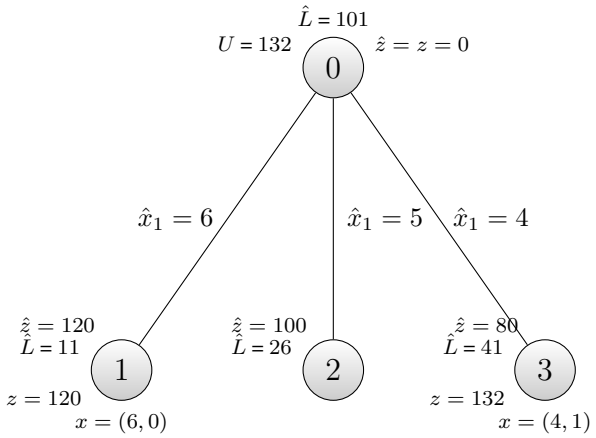

Figura 2.7: Exemplo de Árvore de decisão *Branch and Bound* do exemplo (4) utilizando o MTU2.

O valor obtido após a resolução do "Problema Central" não é igual ao limitante superior U_3 relativo ao problema original (sem a remoção dos itens dominados), onde: $U_3 = max(120 + \left\lfloor 11\frac{31}{25} \right\rfloor, 120 + \left\lfloor \left(11 + \left\lceil \frac{30}{15} \right\rceil 15\right)\frac{52}{41} - \left\lceil \frac{30}{15} \right\rceil 20 \right\rfloor) = 133$, aplicando a etapa de redução nos itens que não estão sendo utilizados no "Problema Central" obtém-se:

$j = 5 := U_1(5) = 31 + \left(100 + \left\lfloor 1\frac{52}{41} \right\rfloor\right) = 132 \leq z$;

$j = 6 := U_1(6) = 4 + \left(120 + \left\lfloor 7\frac{52}{41} \right\rfloor\right) = 132 \leq z$;

$j = 7 := U_1(7) = 5 + \left(120 + \left\lfloor 6\frac{52}{41} \right\rfloor\right) = 132 \leq z$;

Isto é, todos os itens que não estão no "Problema Central" são itens cujos os valores de seus limitante são inferiores ao limitante superior obtido com o "Problema Central", o que conclui que os itens selecionados para o "Problema Central" produz a solução ótima do problema, isto é: $z = 132$ e $(x_j) = (4, 0, 1, 0, 0, 0, 0)$.

O Algoritmo 7 apresenta o método MTU2 proposto por Martello e Toth [21].

Algoritmo 7: Algoritmo MTU2

Entrada: l_i, u_i, L e n

Saída: x_j e Z

$\nu = \max\left(100, \lfloor \frac{n}{100} \rfloor\right)$

$k = 0$;

$\overline{N} = \{1, 2, \ldots, n\}$;

repita

- $k = \min(k + \nu, |\overline{N}|)$;
- Determine o *k-ésimo* maior valor r em $\{\frac{u_j}{l_j} : j \in \overline{N}\}$;
- $G = \{j \in \overline{N} : \frac{u_j}{l_j} > r\}$:
- $E = \{j \in \overline{N} : \frac{u_j}{l_j} = r\}$:
- $\overline{E} =$ qualquer subconjunto de E tal que $|\overline{E}| = k - |G|$;
- $C = G \cup \overline{E}$;
- Ordene os itens de C por ordem decrescente de sua eficiência;
- Resolva o corolário (2.12);
- Resolva o problema central utilizando o algoritmo MTU1 e armazene z e (x_j);
- **se** $k = v$ **então**
  - *Primeira iteração*
  - Calcule o limitante superior U_3 utilizando (2.100);
- **fim**
- **se** $z < U_3$ **então**
  - *Redução*
  - **para todo** $j \in \overline{N} \backslash C$ **faça**
    - $v = U_1(j)$;
    - **se** $v > z$ **então**
      - $v = U_3(j)$;
    - **fim**
    - **se** $v \leq z$ **então**
      - $\overline{N} = \overline{N} \backslash \{j\}$;
    - **fim**
  - **fim**
- **fim**

até $z = U_3$ *ou* $\overline{N} = C$;

para todo $j \in \{1, \ldots, n\} \backslash C$ **faça**

- $x_j = 0$

fim

Fonte: Martello e Toth [21].

3 HEURÍSTICAS PROPOSTAS PARA O PROBLEMA DA MOCHILA COMPARTIMENTADA

Neste capítulo são apresentadas três novas heurísticas para a resolução do problema de preenchimento de uma mochila compartimentada. As heurísticas são denominadas: p_kX, $p_kGULOSO$ e $p_kMTComp$ e são formuladas para o problema da mochila compartimentada utilizando o modelo linear proposto por Inarejos [12]. As heurísticas são fundamentadas na inserção de compartimentos construtivos de maior eficiência no preenchimento da mochila.

As três heurísticas propostas compartilham um mesmo estágio inicial, definido como Processo Inicial.

3.1 PROCESSO INICIAL

O Processo Inicial consiste na determinação dos compartimentos construtivos e na ordenação das classes, compartimentos e itens em ordem decrescente de eficiência. A definição de compartimento construtivo é a seguinte [11]:

Definição 3.1. *Um compartimento é dito construtivo quando existe uma combinação linear inteira não negativa dos pesos dos itens da classe associada, que é igual a capacidade do compartimento em questão.*

A propriedade seguinte apresenta uma forma de verificação dos compartimentos construtivos de uma classe [11]:

Propriedade. Considere o compartimento j da classe k de capacidade $w_j > l^k_{min}$ e os pesos associados a essa classe N_k. Se existe algum item $r \in N_k$, tal que o compartimento de capacidade $w_j - l_r$ é construtivo, então o compartimento de capacidade w_j é construtivo.

Demonstração. Da hipótese de que $w_j - l_r$ é construtivo, segue que $w_j - l_r = \sum_{i \in N_k} l_i a_i$, disto tem-se $w_j = l_r(a_r + 1) + \sum_{\substack{i \in N_k \\ i \neq r}} l_i i a_i$ o que prova a propriedade. □

O Algoritmo 8 apresenta a obtenção dos compartimentos construtivos. Este algoritmo inclui um método para obtenção das utilidades associadas a cada compartimento construtivo. Em seguida, são selecionados os p_k compartimentos mais eficientes[1] de cada classe para a resolução do problema (3.1) - (3.4) e assim obter a melhor utilidade associada a cada

[1] A eficiência associada sera o quociente $\frac{\sum U_j}{\sum W_j}$ de cada compartimento construtivo.

compartimento construtivo selecionado.

Algoritmo 8: Obtenção dos Compartimentos Construtivos e Utilidades iniciais

Entrada: l_i, u_i, l_{max} e l_{min}

Saída: W e U

Seja CL^k conjunto de todos as larguras disponíveis para a classe k.

Inicialização: Para cada classe k realize:

Organize as larguras dos itens em ordem crescente.

enquanto $w_j \leq l_{max}$ **faça**

  se $w_j \in CL$ **então**

    $w_j \in C$

    $ut_j = u_j$

    $w_j = w_j + 1$

  fim

  se $w_j - l_i \in CL$ **então**

    $C := C \cup w_j$

    $i = i + 1$

    $ut_j = u_j + ut_i$

    $w_j = w_j + 1$

  fim

fim

para todo $w_j \in [l_{min}, l_{max}]$ **faça**

  $W_k = \cup w_j$

  $U_k = \cup ut_j$

fim

Fonte: Hoto [11] e adaptado pelo autor.

A modificação realizada na elaboração do Algoritmo 8 ocorre na criação das utilidades iniciais de cada compartimento construtivo. Neste algoritmo a inclusão da utilidade associada ao compartimento ocorre já na criação do compartimento construtivo, são os valores ut_j e U_k.

Após a geração de todos os compartimentos viáveis, será formulado um problema de mochila restrito para a obtenção das melhores utilidades associadas a cada compartimento construído. A forma de obtenção destes valores ótimos das utilidades é obtida através da resolução do problema (3.1) - (3.4):

$$\text{Maximizar } U_j = \sum_{i \in N_k} u_i a_{ij} \tag{3.1}$$

$$\text{Sujeito a: } \sum_{i \in N_k} l_i a_{ij} \leq w_j \tag{3.2}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2 \tag{3.3}$$

$$a_{ij} \geq 0 \text{ e inteiro}, i \in N_k \tag{3.4}$$

Após a obtenção das melhores utilidades associadas a cada um dos p_k compartimentos construtivos de cada classe k, U_j e W_j serão os novos valores para a resolução do problema de programação inteira (3.5) - (3.8), o que resultará na solução viável para o problema da mochila compartimentada. W_k são os índices associados a cada compartimento construtivo.

$$\text{Maximizar Z} = \sum_{i \in W_k} U_i y_{ij} \tag{3.5}$$

$$\text{Sujeito a: } \sum_{i \in W_k} W_i y_{ij} \leq L \tag{3.6}$$

$$\sum_{i \in W_k} y_{ij} \leq F_1 \tag{3.7}$$

$$y_{ij} \geq 0 \text{ e inteiro}, i \in W_k \tag{3.8}$$

As próximas seções apresentam o detalhamento das heurísticas propostas neste trabalho, que são denominadas p_kX, $p_kGULOSO$ e $p_kMTComp$, todas são desenvolvidas para a resolução do Problema da Mochila Compartimentada. A denominação de cada heurística é baseada na forma de resolução dos problemas (3.1) - (3.4) e (3.5) - (3.5). A Figura 3.1 apresenta o diagrama da elaboração das heurísticas.

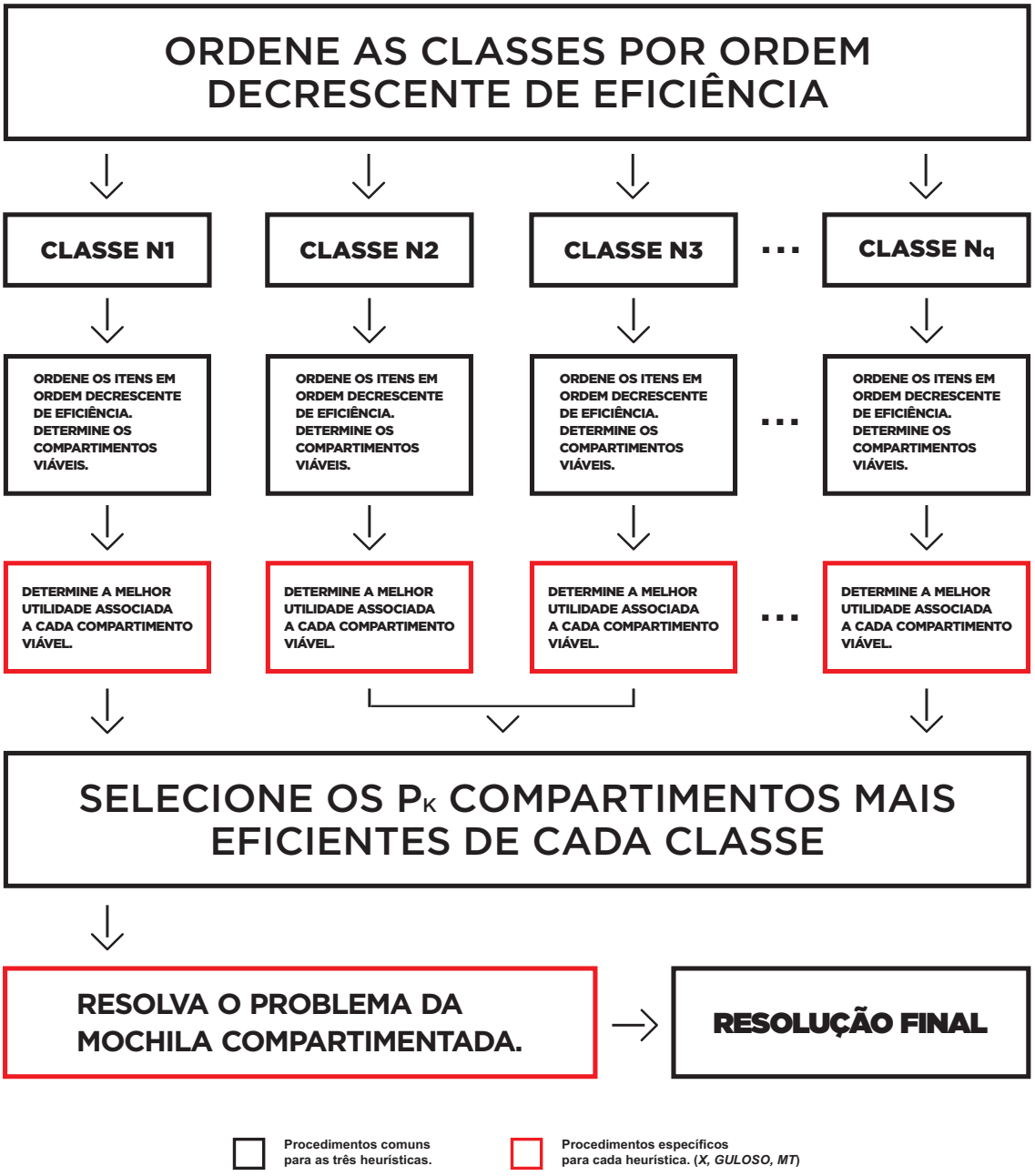

Figura 3.1: Diagrama com o esquema de resolução do Problema da Mochila Compartimentada através das Heurísticas.

3.2 A HEURÍSTICA p_kX

A fim de obter uma solução viável para os problemas (3.1) - (3.4) e (3.5) - (3.8) a heurística denominada p_kX utiliza-se do *software* FICO® Xpress com seu pacote de otimização FICO® Xpress *Optimizer*.

O pacote de otimização é do tipo "caixa-preta", não sendo possível o detalhamento dos procedimentos adotados pela função *maximize* do pacote de otimização. Assim, a

heurística é apresentada pelo algoritmo (9):

Algoritmo 9: Heurística p_kX

Entrada: l_i, W_j, U_j, u_i, l_{max}, l_{min}, L, q, p_k e n

Saída: x_j e Z

para todo k *em* $1, \ldots, q$ **faça**

Ordene as larguras dos itens em ordem crescente.

Crie os compartimentos construtivos executando-se o algoritmo 8.

Determine a eficiência de cada compartimento construtivo.

Selecione os p_k compartimentos mais eficientes.

Resolva o problema (3.1) - (3.4) através da função *maximize*.

fim

Resolva o problema (3.5) - (3.8) através da função *maximize*.

3.3 A HEURÍSTICA $p_kGULOSO$

Em busca de um tempo de execução mais rápido foi desenvolvida a heurística denominada $p_kGULOSO$, a qual não utiliza *software* para a resolução dos problemas (3.1) - (3.4) e (3.5) - (3.8).

Para a resolução dos problemas (3.1) - (3.4) e (3.5) - (3.8) a heurística denominada como $p_kGULOSO$ insere o maior número permitido de itens de maior eficiência no interior da mochila, caso ainda seja possível a inserção de outro item, é disposto o próximo item de maior eficiência, até o preenchimento da capacidade da mochila, isto é, apenas um ramo da árvore de enumeração é gerado. Isto é, $x_1 = \left\lfloor \frac{L}{l_1} \right\rfloor$ e a partir de x_2 são inseridos $\left\lfloor \frac{L - \sum_{i=1}^{j-1} l_i x_i}{l_j} \right\rfloor$ itens, realizando-se sucessivamente até x_n, formando somente o primeiro ramo. Na Figura 3.2 tem-se o ramo desenvolvido com o método guloso, quanto mais escuro o nó, maior a quantidade de itens do tipo j inseridos na mochila.

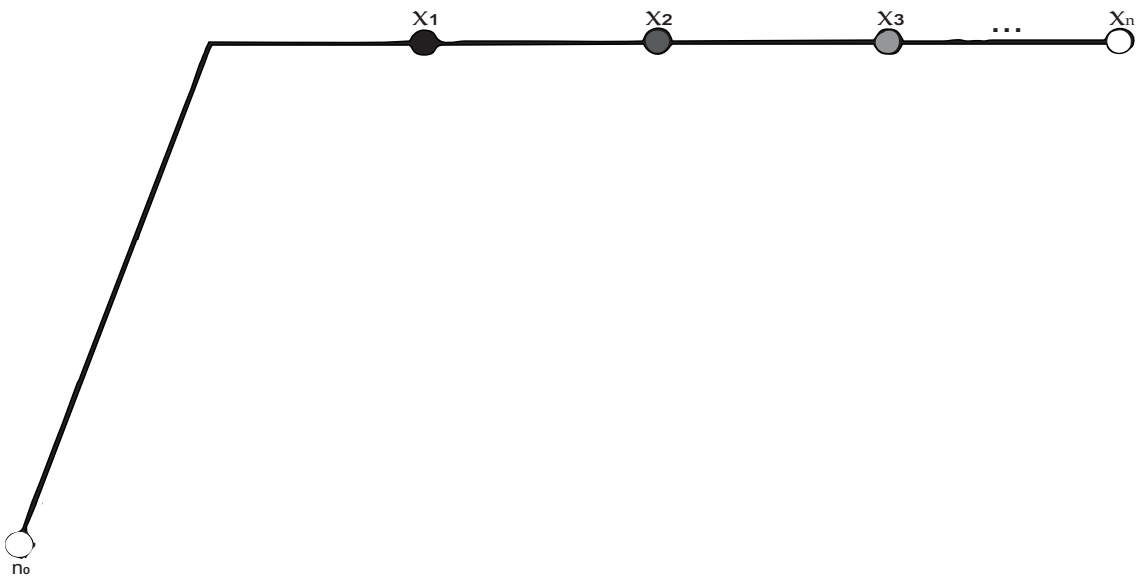

Figura 3.2: Ramo do método $p_kGULOSO$.

O algoritmo 10 descreve a heurística $p_kGULOSO$:

Algoritmo 10: Heurística $p_kGULOSO$

Entrada: l_i, W_j, U_j, u_i, l_{max}, l_{min}, L, q, p_k e n

Saída: x_j e Z

para todo k *em* $1, \ldots, q$ **faça**

Ordene as larguras dos itens em ordem crescente.

Crie os compartimentos construtivos executando-se o algoritmo (8).

Determine a eficiência de cada compartimento construtivo.

Selecione os p_k compartimentos mais eficientes.

Resolva o problema (3.1) - (3.4) através do método guloso.

fim

Resolva o problema (3.5) - (3.8) através do método guloso.

3.4 A HEURÍSTICA $p_kMTComp$

Esta heurística consiste na resolução dos problemas (3.1) - (3.4) e (3.5) - (3.8) utilizando o procedimento MTU2 que é proposto por Martello e Toth [21].

Para a resolução dos problemas (3.1) - (3.4) e (3.5) - (3.8) é necessário realizar uma modificação do algoritmo MTU2, apresentado por Martello e Toth [21], para a inclusão das facas no modelo (restrições (3.3) e (3.7)). A modificação consiste na etapa 2 do método MTU1 onde o novo valor de y a ser considerado é:

$$y = min\{\lfloor\frac{\hat{L}}{l_j}\rfloor, F^*\},$$

onde F^* representa a faca a ser considerada, neste caso ou $F1$ ou $F2$. O Algoritmo 3 em sua

etapa 2 passa a considerar este novo valor de y.

Para a implementação do método MTU1 na linguagem de programação C, foi proposto a utilização do modelo de máquina de estados finitos, eliminando-se as chamadas *goto* do código original, apresentado nos Algoritmos 3, 4, 5 e 6. Na utilização do modelo de máquina de estados finitos, o número de estados é finito, neste caso, cada estado será representado por uma etapa do algoritmo, e a máquina só poderá estar em um único estado por vez, o qual é denominado estado atual.

O estado atual armazena informações dos estados anteriores, refletindo as mudanças desde a entrada dos dados até a saída do resultado, a transição entre estados da máquina ocorre quando uma ação é executada, alterando o estado da máquina.

O uso de *multithreading* também foi possível, onde cada classe na resolução do problema (3.1) - (3.4) é atribuida a uma *thread*, proporcionando uma melhor utilização do recurso computacional.

Deste modo a heurística $p_kMTComp$ é apresentada pelo Algoritmo 11:

Algoritmo 11: Heurística $p_kMTComp$

Entrada: l_i, W_j, U_j, u_i, l_{max}, l_{min}, L, q, p_k e n

Saída: x_j e Z

para todo k *em* $1, \ldots, q$ **faça**

Ordene as larguras dos itens em ordem crescente.

Crie os compartimentos construtivos através do algoritmo (8).

Determine a eficiência de cada compartimento construtivo.

Selecione os p_k compartimentos mais eficientes.

Resolva o problema (3.1) - (3.4) através do algoritmo (7).

fim

Resolva o problema (3.5) - (3.8) através do algoritmo (7).

4 SIMULAÇÕES NUMÉRICAS

Este capítulo apresenta os resultados das simulações numéricas realizadas comparando as heurísticas desenvolvidas, e também é realizada uma comparação entre a heurística $p_kMTComp$ com a heurística das *w capacidades* proposta por Marques [18].

A heurística $w - capacidades$ foi elaborada para o caso irrestrito da mochila compartimentada, que é adotado neste trabalho. Assim, a restrição de demanda não foi considerada na implementação da heurística $w - capacidades$ e sua implementação foi baseada no artigo de Marques [18].

Para as simulações foram utilizados os seguintes *software* e *hardware*: para a implementação, execução e experimentos computacionais do modelo linear e da heurística p_kX foi utilizado a solução do *software* FICO® Xpress para arquitetura 64 bits, que é composta pela interface gráfica (IDE) FICO® Xpress IVE versão $1.24.22$, linguagem FICO® Xpress Mosel versão $4.8.2$ e pacotes de otimização com FICO® Xpress *Optimizer* versão $32.01.07$. As implementações foram compiladas pelo FICO® Xpress e executadas através de sua própria ferramenta de execução dos modelos.

Para a implementação, execução e experimentos computacionais das heurísticas $p_kGULOSO$, $p_kMTComp$, $w - capacidades$ e para o ordenador dos itens e classes por eficiência, foi utilizada a linguagem de programação C e o compilador *gcc* versão 5.1 para arquitetura 32 bits no sistema operacional Microsoft Windows®. Para a compilação foi utilizada somente a *flag* $-o$ do compilador *gcc*.

O *hardware* utilizado em todas as simulações e experimentos computacionais é composto por um processador Intel® Inside™ Xeon® CPU W3520, quatro núcleos com frequência baseada no processador de $2,67$ GHz e oito *threads*, cache de 8 MB e memória RAM de 8 GB executando o sistema operacional Microsoft Windows® Server 2012 R2 Standard, que encontra-se no Simulab, laboratório de simulação do Departamento de Matemática da Universidade Estadual de Londrina.

Os dados para realizar as simulações foram organizados em cinco classes com tamanhos definidos em $q = 5, 10, 20, 50$ e 100, e para cada categoria foram criadas mais cinco sub-categorias, que representam a quantidade de itens disponíveis para cada classe, sendo $n = 10, 50, 100, 1000$ e 10000. A Tabela 4.1 apresenta um resumo da organização das classes e itens.

		Categoria dos Exemplares				
	q	5	10	20	50	100
Sub-Categorias	q/n	5/10	10/10	20/10	50/10	100/10
		5/50	10/50	20/50	50/50	100/50
		5/100	10/100	20/100	50/100	100/100
		5/1000	10/1000	20/1000	50/1000	100/1000
		5/10000	10/10000	20/10000	50/10000	100/10000

Tabela 4.1: Categorias e sub-categorias definidas para os exemplares.

Para a criação dos exemplares foram utilizados os seguintes dados, baseados em problemas reais de corte de bobinas de aço em duas etapas [11]. Para a capacidade total da mochila, foi designado $L = 1100mm$, os valores para as facas 01 e 02 foram respectivamente $F1 = 7$ e $F2 = 12$, a capacidade dos compartimentos de cada classe estão relacionados entre os valores $L^k_{min} = 154mm$ e $L^k_{max} = 456mm$, as larguras dos itens serão geradas com valores distribuídas entre $53mm$ e $230mm$, já as utilidades dos itens serão valores estritamente positivos, tendo como limitante superior o valor de 100.

Para a geração dos exemplares, foi utilizado um gerador aleatório, baseado em [7], onde para cada categoria q/n os valores da largura e utilidade são gerados de forma aleatória. Dessa maneira, foram gerados 100 exemplares para cada categoria, resultando em 2500 exemplares.

Cada um desses 2500 exemplares foi submetido a um processo de organização inicialmente, de modo que as classes e os itens foram organizados através da sua eficiência, como descrito no Capítulo 3.

Após a organização inicial das classes e itens (veja tabela 4.2), foram executadas as simulações para o modelo linear e as heurísticas p_kX, $p_kGULOSO$ e $p_kMTComp$ com os exemplares. Com isso foram elaboradas as Tabelas 4.3, 4.4 e 4.5 em que são detalhados os resultados obtidos e o tempo máximo de execução para cada exemplar foi fixado em 86400 segundos.

A comparação entre os tempos de execução é apenas um indicativo do comportamento entre os métodos. Todas as simulações ocorreram no mesmo equipamento, dentro das mesmas especificações, não ocorrendo nenhum tipo de mudanças durante a execução de todos os exemplares, o equipamento estava dedicado a execução das simulações, realizando apenas este processo.

Nas Tabelas 4.3, 4.4 e 4.5, são apresentadas as seguintes informações: $\overline{T}$ é o tempo médio de execução em segundos, $\sigma(T)$ é o desvio padrão do tempo de execução em segundos, $\overline{gap}$ é o gap médio da heurística em análise, onde o gap é dado por $\frac{z_{linear} - z_{heurstica}}{z_{linear}}$, o gap_{min} e o gap_{max}, são respectivamente o menor e o maior gap obtido entre os exemplares executados na análise da heurística.

q	n	Redução de Compartimentos	Ordenação	
			$\overline{T}$	$\sigma(T)$
5	10	5,18%	0	0
	50	14,22%	0	0
	100	16,60%	0	0
	1000	16,96%	0,001	0,004
	10000	18,84%	0,009	0,007
10	10	5,17%	0	0
	50	14,48%	0,0001	0,001
	100	16,69%	0	0
	1000	16,95%	0,001	0,004
	10000	18,52%	0,018	0,006
20	10	5,63%	0	0
	50	14,25%	0,0001	0,001
	100	16,80%	0,0004	0,002
	1000	16,95%	0,003	0,006
	10000	18,92%	0,0436	0,006
50	10	5,36%	0,0003	0,002
	50	14,25%	0,0009	0,003
	100	16,80%	0,001	0,004
	1000	17,03%	0,013	0,005
	10000	19,03%	0,14	0,008
100	10	5,39%	0,0009	0,003
	50	14,36%	0,003	0,006
	100	16,70%	0,005	0,007
	1000	17,03%	0,037	0,008
	10000	19,05%	0,43	0,01

Tabela 4.2: Redução dos Compartimentos, tempo médio $\overline{T}$ e desvio padrão do tempo $\sigma(T)$ do Processo Inicial.

Ao observar a Tabela 4.2 sobre a redução do número de compartimentos viáveis, nota-se que as porcentagens de redução dos compartimentos são próximas independentemente do número de classes , quando as classe possuem o mesmo número de itens n, isto é devido à forma de criação e seleção dos compartimentos viáveis. Quanto aos valores dos tempos relacionados à ordenação, em diversos casos esses valores foram inferiores a microssegundos[1].

[1] 1 microssegundo $= 10^{-6}$ segundos.

A Tabela 4.3 apresenta as medidas estatísticas relacionadas a heurística p_kX em comparação ao modelo linear. Para os exemplares das classes $50/10000$, $100/1000$ e $100/10000$ o tempo de execução de cada exemplar é superior ao tempo máximo de 86400 segundos, estes resultados estão sendo representados por $*$.

q	**n**	$Linear$		p_kX					
		$\overline{T}$	$\sigma(T)$	$\overline{T}$	$\sigma(T)$	$\overline{gap}$	$\sigma(gap)$	gap_{min}	gap_{max}
5	10	217,6535	658,0430	1,0658	0,042	0,03%	0,24%	0%	2,23%
	50	43,7511	420,2747	2,6918	0,096	0,51%	1,26%	0%	5%
	100	0,0601	0,0321	3,0362	0,4554	3,04%	2,98%	0%	11,10%
	1000	0,1930	0,0443	13,1726	0,1672	0,78%	1,25%	0%	3,99%
	10000	3,2295	0,1215	8276,237	32,3817	0,01%	0,02%	0%	0,11%
10	10	1,5687	9,5373	2,6798	0,4554	0,11%	0,26%	0%	0,98%
	50	1,5936	15,3062	12,5866	0,6542	0,84%	1%	0%	5,56%
	100	0,1157	0,0493	13,4914	0,2359	1,98%	2,11%	0%	10,53%
	1000	0,4233	0,0464	40,0924	0,3678	0,64%	1,18%	0%	3,20%
	10000	12,1816	0,2539	16624,74	59,062	0,01%	0,01%	0%	0,05%
20	10	29,9757	181,3964	14,949	1,953	0,04%	0,36%	0%	3,57%
	50	0,1521	0,2808	71,3141	1,6496	0,36%	1,04%	0%	3,92%
	100	18,9978	187,6012	76,1120	0,7393	1,19%	1,57%	0%	6,68%
	1000	1,0560	0,0962	185,2088	0,969	0,46%	1,09%	0%	3,13%
	10000	100,902	0,8131	34152,2	241,51	0%	0%	0%	0%
50	10	0,1070	0,2117	202,236	40,2187	0,02%	0,08%	0%	0,38%
	50	0,4731	1,1416	900,1507	9,3217	0,19%	0,59%	0%	2,31%
	100	26,4013	257,8968	961,2703	6,1443	0,35%	0,69%	0%	3,25%
	1000	4,8865	0,3422	2420,67	32,6762	0,01%	0,06%	0%	0,57%
	10000	1058,958	14,3281	*	*	*	*	*	*
100	10	23,7795	235,5939	1560,647	267,4781	0,10%	0,74%	0%	6,18%
	50	26,6162	257,8439	6698,394	53,293	0,06%	0,37%	0%	2,22%
	100	26,6663	252,8972	7216,887	32,2862	0,14%	0,44%	0%	2,84%
	1000	30,9031	1,3370	*	*	*	*	*	*
	10000	5226,72	64,4448	*	*	*	*	*	*

Tabela 4.3: Medidas Estatística da Heurística p_kX.

q	n	$Linear$		$p_kGULOSO$					
		$\overline{T}$	$\sigma(T)$	$\overline{T}$	$\sigma(T)$	$\overline{gap}$	$\sigma(gap)$	gap_{min}	gap_{max}
5	10	217,6535	658,0430	0,0009	0,0035	14,33%	4,34%	1,11%	26,53%
	50	43,7511	420,2747	0,0156	0,0049	10,64%	4,31%	0%	20,33%
	100	0,0601	0,0321	0,0241	0,0079	10,77%	6,50%	1,14%	30,59%
	1000	0,1930	0,0443	0,1011	0,008	14,85%	9,15%	0,51%	30,93%
	10000	3,2295	0,1215	8,4886	0,0154	15,03%	7,47%	0,63%	31,15%
10	10	1,5687	9,5373	0,0036	0,0064	9,74%	5,98%	1,17%	20%
	50	1,5936	15,3062	0,0314	0,0081	11,13%	6,05%	0%	31,53%
	100	0,1157	0,0493	0,0506	0,0082	10,64%	6,94%	1,14%	34,33%
	1000	0,4233	0,0464	0,2023	0,0058	12,92%	8,98%	0,51%	30,51%
	10000	12,1816	0,2539	16,9356	0,0174	15,41%	6,71%	1,38%	31,18%
20	10	29,9757	181,3964	0,0058	0,0073	8,30%	8,71%	0%	27,27%
	50	0,1521	0,2808	0,0631	0,0099	10,10%	5,26%	0%	37,86%
	100	18,9978	187,6012	0,1021	0,0126	8,63%	6,53%	0%	24,36%
	1000	1,0560	0,0962	0,4021	0,0081	12,97%	8,55%	0%	30,84%
	10000	100,902	0,8131	33,9172	0,0235	14,51%	8,89%	1,69%	31,22%
50	10	0,1070	0,2117	0,015	0,0044	8,06%	8,30%	0%	25,96%
	50	0,4731	1,1416	0,1563	0,0162	10,75%	6,61%	0%	34,69%
	100	26,4013	257,8968	0,2528	0,0215	9,50%	6,25%	0%	30%
	1000	4,8865	0,3422	1,0054	0,0086	14,89%	7,26%	0,40%	31,69%
	10000	1058,958	14,3281	84,6561	0,0599	14,52%	9,83%	1,74%	31,22%
100	10	23,7795	235,5939	0,0297	0,0053	9,11%	7,45%	0%	31,37%
	50	26,6162	257,8439	0,3148	0,0269	9,58%	5,60%	0%	30%
	100	26,6663	252,8972	0,4994	0,0298	7,77%	6,54%	0%	25%
	1000	30,9031	1,3370	2,0107	0,0079	16,14%	8,24%	0,41%	32,18%
	10000	5226,72	64,4448	169,2361	0,0922	13,73%	9,93%	1,74%	31,22%

Tabela 4.4: Medidas Estatística da Heurística $p_kGULOSO$.

A Tabela 4.4 detalha os resultados das medidas estatísticas obtidas através dos experimentos computacionais da heurística $p_kGULOSO$.

q	n	$Linear$		$p_kMTComp$					
		$\overline{T}$	$\sigma(T)$	$\overline{T}$	$\sigma(T)$	$\overline{gap}$	$\sigma(gap)$	gap_{min}	gap_{max}
5	10	217,6535	658,0430	0,00183	0,0048	1,21%	1,78%	0%	7,14%
	50	43,7511	420,2747	0,01081	0,0055	1,64%	2,55%	0%	11,42%
	100	0,0601	0,0321	0,01125	0,0063	2,03%	1,73%	0%	5,82%
	1000	0,1930	0,0443	0,0575	0,012	1,12%	1,54%	0%	6,12%
	10000	3,2295	0,1215	1,897	0,6076	1,10%	1,42%	0%	3,84%
10	10	1,5687	9,5373	0,0045	0,0069	1,14%	1,48%	0%	10,20%
	50	1,5936	15,3062	0,01155	0,0068	1,58%	1,85%	0%	5,56%
	100	0,1157	0,0493	0,0063	0,0082	1,58%	1,54%	0%	6,15%
	1000	0,4233	0,0464	0,0909	0,0107	0,99%	1,25%	0%	3,61%
	10000	12,1816	0,2539	3,4362	0,8654	0,35%	0,63%	0%	3,54%
20	10	29,9757	181,3964	0,007	0,0075	0,66%	1,77%	0%	6,32%
	50	0,1521	0,2808	0,0214	0,009	0,60%	1,24%	0%	5,26%
	100	18,9978	187,6012	0,03504	0,0118	1,28%	1,45%	0%	6,68%
	1000	1,0560	0,0962	0,1624	0,0195	0,99%	1,21%	0%	3,55%
	10000	100,902	0,8131	6,7815	1,1933	0,18%	0,09%	0,05%	0,48%
50	10	0,1070	0,2117	0,0111	0,0071	0,22%	0,73%	0%	4,21%
	50	0,4731	1,1416	0,0487	0,0157	0,41%	0,89%	0%	5,26%
	100	26,4013	257,8968	0,0792	0,0225	0,66%	1,06%	0%	5,48%
	1000	4,8865	0,3422	0,403	0,0224	0,96%	1,62%	0%	7,39%
	10000	1058,958	14,3281	17,308	6,2321	0,11%	0,06%	0,05%	0,26%
100	10	23,7795	235,5939	0,027	0,0071	0,64%	1,50%	0%	6,90%
	50	26,6162	257,8439	0,1028	0,0232	0,33%	0,83%	0%	5,26%
	100	26,6663	252,8972	0,1582	0,03	0,63%	0,98%	0%	3,28%
	1000	30,9031	1,3370	0,7968	0,0388	0,96%	1,25%	0%	6,09%
	10000	5226,72	64,4448	35,049	8,2867	0,09%	0,05%	0%	0,21%

Tabela 4.5: Medidas Estatística da Heurística $p_kMTComp$.

Por fim, a Tabela 4.5 apresenta as medidas estatísticas relacionadas à heurística $p_kMTComp$ em comparação com o modelo linear proposto por [12].

A Figura 4.1 mostra um gráfico que permite a visualização do comportamento das heurísticas em relação ao $\overline{gap}$.

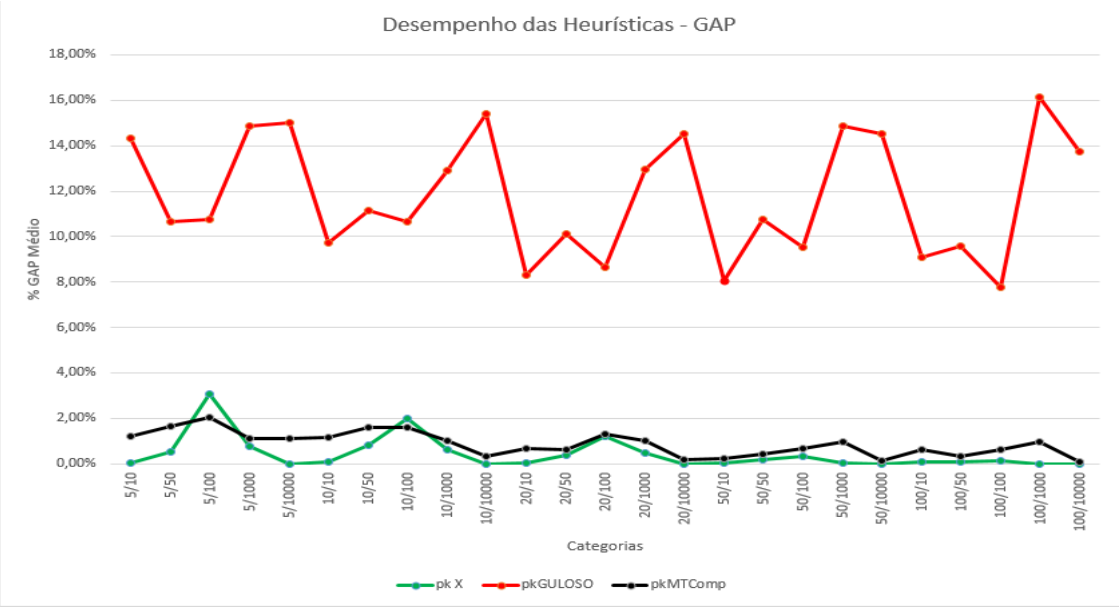

Figura 4.1: Desempenho das Heurísticas - $\overline{gap}$

A Tabela 4.6 apresenta as comparações entre a heurística $p_kMTComp$ e a heurística das $w - capacidades$ proposta por Marques [18]. Para as categorias com $n = 1000$ e $n = 10000$ a heurística $w - capacidades$ não obteve soluções, isto ocorre devido à sua elaboração com a utilização do método de Gilmore e Gomory [8] para a resolução dos subproblemas internos da heurística, que resolve problemas com até centenas de itens. O uso do método de Gilmore e Gomory por Marques [18] na resolução dos subproblemas é justificado pela ausência de categorias com centenas de itens na indústria de corte de bobinas. Essas categorias estão representadas por $*$ na Tabela 4.6.

A Figura 4.2 mostra o comportamento das heurísticas $p_kMTComp$ e $w - capacidades$ em relação ao $\overline{gap}$.

q	n	$w - Capacidades$				$p_k MTComp$			
		$\overline{gap}$	$\sigma(gap)$	gap_{min}	gap_{max}	$\overline{gap}$	$\sigma(gap)$	gap_{min}	gap_{max}
5	10	5,32%	7,67%	0%	28,11%	1,21%	1,78%	0%	7,14%
	50	16,82%	8,58%	0%	30,00%	1,64%	2,55%	0%	11,42%
	100	18,76%	4,28%	0,39%	22,73%	2,03%	1,73%	0%	5,82%
	1000	*	*	*	*	1,12%	1,54%	0%	6,12%
	10000	*	*	*	*	1,10%	1,42%	0%	3,84%
10	10	4,55%	7,50%	0%	26%	1,14%	1,48%	0%	10,20%
	50	11,40%	9,75%	0%	28,76%	1,58%	1,85%	0%	5,56%
	100	19,02%	3,22%	3,16%	23,00%	1,58%	1,54%	0%	6,15%
	1000	*	*	*	*	0,99%	1,25%	0%	3,61%
	10000	*	*	*	*	0,35%	0,63%	0%	3,54%
20	10	1,81%	2,26%	0%	8,24%	0,66%	1,77%	0%	6,32%
	50	12,72%	9,88%	0%	30,00%	0,60%	1,24%	0%	5,26%
	100	17,65%	5,32%	1%	22,73%	1,28%	1,45%	0%	6,68%
	1000	*	*	*	*	0,99%	1,21%	0%	3,55%
	10000	*	*	*	*	0,18%	0,09%	0,05%	0,48%
50	10	2,68%	5,01%	0%	33,68%	0,22%	0,73%	0%	4,21%
	50	12,09%	9,64%	0%	31,32%	0,41%	0,89%	0%	5,26%
	100	16,86%	6,42%	0%	23%	0,66%	1,06%	0%	5,48%
	1000	*	*	*	*	0,96%	1,62%	0%	7,39%
	10000	*	*	*	*	0,11%	0,06%	0,05%	0,26%
100	10	3,92%	6,24%	0%	37,27%	0,64%	1,50%	0%	6,90%
	50	11,32%	9,70%	0%	31%	0,33%	0,83%	0%	5,26%
	100	16,40%	6,70%	0%	23%	0,63%	0,98%	0%	3,28%
	1000	*	*	*	*	0,96%	1,25%	0%	6,09%
	10000	*	*	*	*	0,09%	0,05%	0%	0,21%

Tabela 4.6: Comparação entre as Heurísticas $p_k MTComp$ e $w - capacidades$.

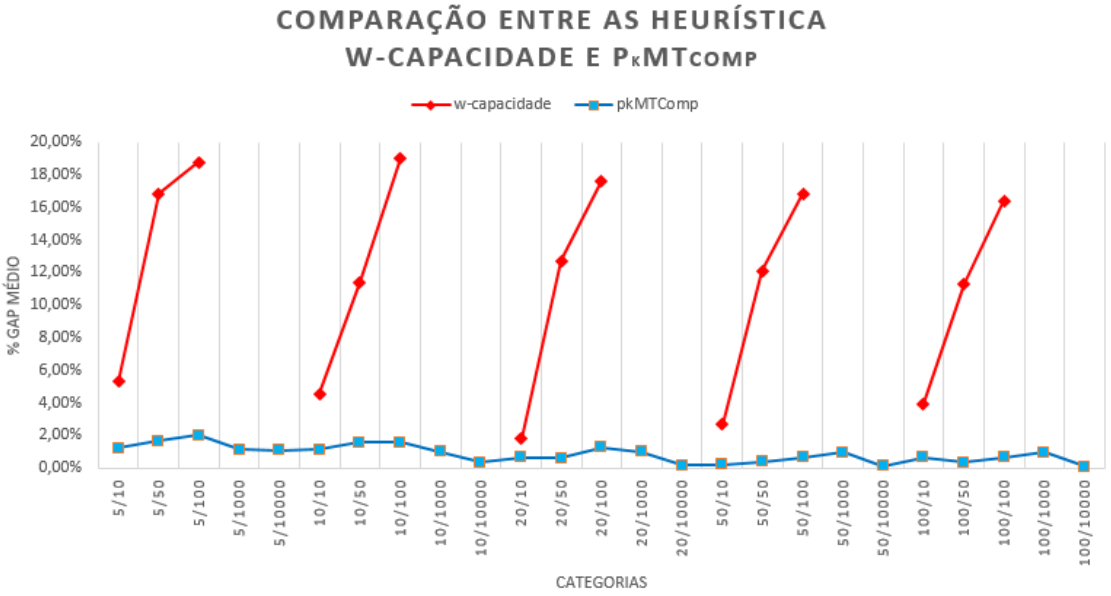

Figura 4.2: Comparação entre as Heurísticas $w - capacidades$ e $p_kMTComp$ - $\overline{gap}$

4.1 Análise dos Resultados Numéricos

A execução dos exemplares de cada categoria forneceu resultados que permitem analisar com detalhes o comportamento das três heurísticas propostas. É discutido o resultado de cada heurística e analisado o seu desempenho.

Dentre as três heurísticas propostas a heurística p_kX mostrou pior desempenho quanto ao tempo de execução. Em alguns casos apresentando tempos médios superiores a um dia de execução para cada exemplar, como pode ser visto na Tabela 4.3. Em relação a qualidade das soluções obtidas, 100% das categorias obtiveram soluções que estavam a menos de 5% do valor ótimo, mostrando soluções confiáveis em relação ao ótimo.

Um destaque em relação à qualidade das soluções obtidas é para a categoria $20/10000$, em que todos os 100 exemplares solucionados apresentaram solução igual a ótima. Outro ponto a ser notado é o baixo desvio padrão das soluções, mostrando que a discrepância entre os valores é baixa.

O valores gap_{min} e gap_{max} denotam a amplitude dos intervalos gap obtidos. No caso da heurística p_kX, em $91,3\%$ este intervalo foi inferior a 10%, denotando intervalos pequenos. Apesar do tempo de execução indicar valores altos, a heurística p_kX demonstra possuir qualidade nas soluções obtidas como pode ser visto na Figura 4.1.

A heurística $p_kGULOSO$ preenche a mochila com os itens de maior eficiência Essa característica possibilita indicativos de tempo com maior desempenho em relação a heurística p_kX. Por outro lado, devido à esta característica, as soluções apresentam gap superiores às heurísticas em comparação, p_kX e $p_kMTComp$. Dos resultados obtidos através dos experimentos realizados, apenas 32% apresentam gap médio inferior a 10%, sendo que nenhuma das categorias apresentou gap médio inferior a 5%.

Outra situação desfavorável a esta heurística são as amplitudes dos intervalos gap possuindo intervalos superiores a 20% em todas as categorias simuladas. Outro fator a ser notado é que em 52% das categorias simuladas, em nenhum dos 100 exemplares de cada categoria o ótimo foi encontrado como solução, apresentando o pior desempenho em relação as soluções obtidas em comparação com as heurísticas p_kX e $p_kMTComp$.

Por fim, a heurística $p_kMTComp$ dentre as três heurísticas propostas, apresentou resultados que indicam o melhor desempenho para os exemplares considerados. O fato desta heurística não depender de *softwares* proprietários com soluções embarcadas para a resolução dos subproblemas (3.1) - (3.4) e (3.5) - (3.8) é um diferencial quando comparada com a heurística p_kX. A utilização do método de Martello e Toth [21] possibilitou soluções com valores gap baixos, 100% dos valores médios obtidos ficaram a menos de 5%, sendo que 96% dos valores médios obtidos ficaram a menos de 2% da solução ótima e o desvio padrão apresenta que 96% dos valores estão abaixo de 2%, como pode ser visto na Tabela 4.5.

Sobre os intervalos gap obtidos, 92% ficaram abaixo de 10%, indicando variações pequenas dos resultados obtidos. Outro fator a ser notado é para categorias com $n = 10000$, as quais apresentam melhores indicativos de desempenho das soluções obtidas.

Em relação ao tempo de execução, os resultados realizados nestes exemplares indicam um tempo de execução final inferior a heurística p_kX e em algumas categorias valores inferiores à heurística $p_kGULOSO$.

Como apresentado nas Tabelas 4.3, 4.4 e 4.5, a heurística $p_kMTComp$ possui um desempenho superior em relação às heurísticas p_kX e $p_kGULOSO$, cujos resultados indicam que é uma heurística promissora entre as três desenvolvidas. Com o objetivo de verificar o desempenho das soluções obtidas pela heurística $p_kMTComp$ em relação à outra heurística disponível na literatura, realizou-se a comparação com a heurística $w - capacidades$ [18].

A Tabela 4.6 apresenta os resultados obtidos através dos experimentos realizados com a heurística $p_kMTComp$ e $w - capacidades$ que indicam um desempenho superior da heurística $p_kMTComp$ em relação à solução obtida, produzindo soluções de menor gap em comparação com a heurística $w - capacidade$. Outro ponto em que a heurística $p_kMTComp$ é superior à heurística $w - capacidades$ é em solucionar exemplares com milhares de itens, algo não realizado pela heurística $w - capacidades$. A utilização do método de Martello e Toth [21] possibilita a execução de exemplares com milhares de itens, já a heurística $w - capacidades$ ao utilizar o método de Gilmore e Gomory [8] não é capaz de solucioná-los.

As soluções obtidas através da execução da heurística $w - capacidades$ apresentam para os exemplares com $n = 10$ melhores soluções quando comparadas com as soluções obtidas em exemplares com $n = 50$ e $n = 100$, quando comparadas as soluções, em exemplares com $n = 10$ o gap fica próximo a 5% no caso de 5 classes e abaixo dos 5% de gap para as categorias $20/10$, $50/10$ e $100/10$. Observando-se os resultados obtidos por Marques [18] é possível verificar que quando o número de itens aumenta, são obtidas soluções de maior gap como pode ser visto na Figura 4.2.

5 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho aborda o Problema da Mochila Compartimentada, e apresenta como proposta a elaboração de três novas heurísticas para a sua resolução. No desenvolvimento de duas das três heurísticas, utilizou-se a linguagem de programação C, realizando a implementação com o método de máquina de estados finitos, como alternativa para a eliminação das chamadas *goto* do algoritmo MTU1 e MTU2 inicial, proposto por [21]. Também fez-se o uso de *threads* na implementação da heurística $p_kMTComp$.

Uma modificação foi realizada no algoritmo MTU1, para a inclusão das facas que estão presentes no modelo linear proposto por [12], deixando a modelagem do algoritmo fiel à realidade, também realizou-se uma modificação no Algoritmo de elaboração dos compartimentos viáveis, incluindo a utilidade associada a cada compartimento em sua criação.

Como contribuições, pode-se destacar a heurística $p_kMTComp$, cujos experimentos preliminares indicam que o método é promissor. O indicativo em relação ao tempo mostra que quando comparada com as heurísticas p_kX e $p_kGULOSO$ o método com a utilização do algoritmo de Martello e Toth [21], proporciona resultados com o gap menor, resultando em soluções mais próximas ao ótimo.

Quando comparada com a heurística $w - capacidades$, os testes realizados indicam que a heurística $p_kMTComp$ produz resultados mais próximos ao ótimo, isto é gap inferiores. A heurística $p_kMTComp$ é capaz de solucionar categorias com um número superior de itens em cada classe, o que não ocorre com a heurística $w - capacidades$ [18]. Outro ponto é em relação ao gap obtido com a heurística $p_kMTComp$ sendo inferior ao da heurística $w - capacidades$. Assim, a execução das simulações numérica e análise dos resultados indicam que a heurística $p_kMTComp$ é superior para a resolução do problema da mochila compartimentada, quando comparada com a heurística $w - capacidades$.

A continuação deste trabalho envolve o desenvolvimento da heurística para o caso restrito da mochila, quando ocorre a limitação de itens disponíveis para o preenchimento da mochila. Para a resolução dos subproblemas (3.1) - (3.4) e (3.5) - (3.8) será utilizado o método proposto por Pisinger [23], utilizando programação dinâmica. Esse método proposto para o caso restrito do problema da mochila é superior ao proposto por Martello e Toth em [21]. Após o desenvolvimento para o caso restrito será realizada uma comparação com a heurística $w - capacidades$ para o caso restrito, proposto em [18].

Outro ponto a ser considerado na continuação desta pesquisa é considerar apenas uma parcela dos compartimentos construtivos para a resolução do problema (3.1) - (3.4), considerando inicialmente apenas compartimentos de classes com maior eficiência. Eventualmente pode-se considerar uma parcela de compartimentos de classes com maior eficiência e gradativamente ir reduzindo o número de compartimentos a serem considerados em classes com menor eficiência.

Como aplicação, a proposta consiste na utilização da heurística com o objetivo de otimizar a reprodução de vídeos através do protocolo DASH [1]. Serão realizados testes em um cenário simulado através do *software* NS-3 e incorporado em um algoritmo a heurística.

Em resumo, a principal contribuição desta dissertação consiste na elaboração de novas heurísticas para o problema da mochila compartimentada e este desenvolvimento é possível devido à prova da linearidade do problema da mochila compartimentada, realizada por Inarejos [13]. Novos problemas que apresentem estruturas análogas a este trabalho podem ser beneficiados com as ideias apresentadas.

REFERÊNCIAS

[1] 23009-1.2, I. D. Information technology — dynamic adaptive streaming over http (dash) — part 1: Media presentation description and segment formats. Tech. rep., ISO/IEC, 2014.

[2] BECKER;, H. The unbounded knapsack problem: a critical review. Master thesis, Universidade Federal do Rio Grande do Sul, Mar. 2017.

[3] BRITO, A., NETO, J. C., SANTOS, P., AND SOUZA, S. A relaxed projection method for solving multiobjective optimization problems. *European Journal of Operational Research 256*, 1 (2017), 17 – 23.

[4] CHVATAL, V. *Linear Programming*. Series of books in the mathematical sciences. BEDFORD BOOKS, 2016.

[5] CRUZ, E. P. D. Uma abordagem heurística linear para mochilas compartimentadas restritas. Dissertação de mestrado, UEL, Londrina, 2010.

[6] FERREIRA, J. S., NEVES, M., AND CASTRO, P. A two–phase roll cutting problem. *European Journal of Operational Research 44* (1990), 185–196.

[7] GAU, T., AND WÄSHER, G. Cutgen1: A problem generator for the standar one-dimensional cutting stock problem. *European Jornal of Operational Research 84*, 3 (1995), 572–579.

[8] GILMORE, P. C., AND GOMORY, R. E. A linear programming approach to the cutting stock problem - part ii. *Operations Research* (May 1963).

[9] HOTO, R., FENATO, A., YANNASSE, H., MACULAN, N., AND SPOLADOR, F. Uma nova abordagem para o problema da mochila compartimentada. In *Anel do XXXVIII Simpósio brasileiro de Pesquisa Operacional* (2006).

[10] HOTO, R., MACULAN, N., MARQUES, F., AND ARENALES, M. Um problema de corte com padrões compartimentados. *Pesquisa Operacional 23*, 1 (jan 2003), 169–187.

[11] HOTO, R. S. V. *O Problema da Mochila Compartimentada Aplicado no Corte de Bobinas de Aço*. Tese de doutorado, UFRJ, Rio de Janeiro, 2001.

[12] INAREJOS, O. Sobre a não-linearidade do problema da mochila compartimentada. Dissertação de mestrado, UEL, Londrina, 2015.

[13] INAREJOS, O., HOTO, R., AND MACULAN, N. A integer linear optimization model to the compartimentalized knapsack problem. *International Transactions in Operational Research 00*, 1 (out 2017), 1–20.

[14] KELLERER, H., PFERSCHY, U., AND PISINGER, D. *Knapsack Problems*. 01 2004.

[15] LEAO, A. A. DE S.; SANTOS, M. O. A. M. N., AND HOTO, R. S. V. Uma heurística para o problema da mochila compartimentada. In *XL SBPO* (2008).

[16] LEÃO, A. A. D. S. Geração de colunas para problemas de cortes em duas fases. Dissertação de mestrado, ICMC - USP, São Carlos, 2009.

[17] MACÊDO, L. D. D. Análise do comportamento inovativo da indústria nacional no período de 2003-2014 sob a influência das políticas industriais de inovação. Dissertação, Universidade Federal de Alagoas, Maceio, June 2018.

[18] MARQUES, F., AND ARENALES, M. N. The constrained compartmentalised knapsack problem. *Computers & Operations Research 34*, 7 (2007), 2109 – 2129.

[19] MARQUES, F. P. O problema da mochila compartimentada. Dissertação de mestrado, ICMC - USP, 2000.

[20] MARQUES, F. P., AND ARENALES, M. N. O problema da mochila compartimentada e aplicações. *Pesquisa Operacional 22*, 3 (Dec. 2002), 285–304.

[21] MARTELLO, S., AND TOTH, P. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[22] OROZCO, J. J. Q. Um método *Branch and Bound* para o problema da compartimentação das mochilas. Dissertação de mestrado, Universidade Estadual de Londrina, Londrina, 2018.

[23] PISINGER, D. A minimal algorithm for the bounded knapsack problem. *Computer Science* (1995), 95–109.

[24] QUIROGA-OROZCO, J. J., DE CARVALHO, J. M. V., AND HOTO, R. S. V. A strong integer linear optimization model to the compartmentalized knapsack problem. *International Transactions in Operational Research* (2018).

[25] SILVA, G. N., SANTOS, P. S. M., AND SOUZA, S. S. Extended newton-type method for nonlinear functions with values in a cone. *Computational and Applied Mathematics 37*, 4 (Sep 2018), 5082–5097.

[26] SOBHANI, A., YASSINE, A., AND SHIRMOHAMMADI, S. Qoe-driven optimization for dash service in wireless networks. In *2016 IEEE International Symposium on Multimedia (ISM)* (Dec 2016), pp. 232–237.

[27] THANG, T. C. ; HO, Q.-D. K. J. W., AND PHAM, A. T. Adaptive streaming of audiovisual content using mpeg dash. *IEEE Transactions on Consumer Electronics 58*, 1 (2012), 78–85.

[28] XAVIER, E., AND MIYAZAWA, F. The class constrained bin packing problem with applications to video-on-demand. *Theoretical Computer Science 393*, 1-3 (mar 2008), 240–259.