UNIVERSIDADE ESTADUAL DE LONDRINA

OSVALDO INAREJOS FILHO

SOBRE A NÃO-LINEARIDADE DO PROBLEMA DA MOCHILA COMPARTIMENTADA

Londrina
2015

OSVALDO INAREJOS FILHO

SOBRE A NÃO-LINEARIDADE DO PROBLEMA DA MOCHILA COMPARTIMENTADA

Dissertação de mestrado apresentada ao Departamento de Matemática da Universidade Estadual de Londrina, como parte dos requisitos para obtenção do título de Mestre em Matemática Aplicada e Computacional.

Orientador: Prof. Dr. Robinson Samuel Vieira Hoto

Londrina
2015

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Inarejos Filho, Osvaldo.
Sobre a Não-linearidade do Problema da Mochila Compartimentada / Osvaldo Inarejos Filho. - Londrina, 2015.
145 f. : il.

Orientador: Robinson Samuel Vieira Hoto.
Dissertação (Mestrado em Matemática Aplicada e Computacional) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Matemática Aplicada e Computacional, 2015.
Inclui bibliografia.

1. Problema da Mochila Compartimentada - Teses. 2. Otimização Discreta - Teses. I. Hoto, Robinson Samuel Vieira. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Matemática Aplicada e Computacional. III. Título.

OSVALDO INAREJOS FILHO

SOBRE A NÃO-LINEARIDADE DO PROBLEMA DA MOCHILA COMPARTIMENTADA

Dissertação de mestrado apresentada ao Departamento de Matemática da Universidade Estadual de Londrina, como parte dos requisitos para obtenção do título de Mestre em Matemática Aplicada e Computacional.

BANCA EXAMINADORA

_____

Orientador: Prof. Dr. Robinson Samuel Vieira Hoto
Universidade Estadual de Londrina - UEL

_____

Prof. Dr. André Luís Machado Martinez
Universidade Tecnológica Federal do Paraná - UTFPR

_____

Prof. Dr. Naresh Kumar Sharma
Universidade Estadual de Londrina - UEL

_____

Prof. Dr. Silvio Alexandre de Araujo
Universidade Estadual Paulista - UNESP

Londrina, 18 de dezembro de 2015.

AGRADECIMENTOS

Agradeço a meus pais por terem me dado a oportunidade e o incentivo a estudar sempre mais. À minha noiva por ter me apoiado e ajudado como pôde durante a realização do mestrado.

Aos professores do PGMAC, que disponibilizaram tempo e dedicação a lecionar e orientar. Ao Robinson por conduzir o trabalho de maneira dinâmica, me permitindo pesquisar com liberdade e assim desenvolver minhas ideias.

Agradeço também a todos os alunos do PGMAC pelo companheirismo e aprendizado cooperativo.

À Capes e ao CNPq pelo apoio financeiro, também à Fico® por ceder o Xpress Optmizer.

INAREJOS FILHO, Osvaldo. **Sobre A não-linearidade do problema da mochila compartimentada**. 2015. 145. Dissertação de Mestrado (Matemática Aplicada e Computacional) – Universidade Estadual de Londrina, Londrina, 2015.

RESUMO

O Problema da Mochila Compartimentada surge de problemas de corte em duas fases, especialmente no corte de bobinas de aço. Em sua formulação original, trata-se de um problema de otimização inteira não-linear, e até então este problema tem sido resolvido por meio de heurísticas de decomposição. Esta dissertação tem por objetivo mostrar que o Problema da Mochila Compartimentada Restrito é um problema de Otimização Linear, justificando novos estudos voltados a uma nova abordagem do problema. Para isto, faz-se uma revisão dos problemas de mochila e suas aplicações (em especial nos problemas de corte de estoque), analisa-se o Problema da Mochila Compartimentada Restrito em suas formulações anteriores a este trabalho, e apresenta-se um novo modelo linear no qual organiza-se ensaios numéricos e prova-se ser um modelo equivalente ao original.

Palavras-chave: Corte de Estoque em Duas Fases. Otimização Discreta. Otimização Linear. Problema da Mochila Compartimentada. Problema de Corte de Estoque Unidimensional.

INAREJOS FILHO, Osvaldo. **About the non-linearity of the compartmentalized knapsack problem.** 2015. 145. Dissertação de Mestrado (Matemática Aplicada e Computacional) – Universidade Estadual de Londrina, Londrina, 2015.

ABSTRACT

The Compartmentalized Knapsack Problem arises from problems of cutting into two phases, especially in cutting of steel rolls. In its original formulation, it is a non-linear integer programming problem, and this problem has been solved by decomposition heuristics. This paper aims to show that the Restricted Compartmentalised Knapsack Problem is a linear optimization problem, justifying further studies aimed at a new approach to the problem. For this, contains a review of the knapsack problems and its applications (especially in cutting stock problems), a analysis of the Restricted Compartmentalised Knapsack Problem in previous formulations to this work, and a new linear model to which is organized numerical essays and a proof that it be equivalent to the original model.

Key words: Compartmentalized Knapsack Problem. Discrete Optimization. Linear Optimization. Two-fase Cutting Stock Problem. One-dimensional Cutting Stock Problem.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BF	*Best-Fit*
BFD	*Best-Fit Decreasing*
FF	*First-Fit*
FFD	*First-Fit Decreasing*
NF	*Next-fit*
NFD	*Next-Fit Decreasing*
PCBA	Problema de Corte das Bobinas de Aço
PCE	Problema de Corte de Estoque Unidimensional
PMC	Problema da Mochila Compartimentada
PMCP	Problema da Mochila Compartimentada Particular
PMCR	Problema da Mochila Compartimentada Restrito

SUMÁRIO

INTRODUÇÃO

Este trabalho apresenta um avanço teórico no estudo do Problema da Mochila Compartimentada. Problemas de mochila envolvem a escolha de itens a serem incluídos em mochilas, e surgem como subproblema gerador de colunas em problemas de corte de estoque. O Problema da Mochila Compartimentada surge no Problema de Corte das Bobinas de Aço.

O Problema de Corte das Bobinas de Aço é parte de um conjunto mais abrangente de problemas, os problemas de corte em duas fases (HAESSLER, 1979; JOHNSTON, KHAN, 1995; ZAK, 2000; CORREIA *et al*, 2004). Nestes problemas, o corte de um determinado material é feito em duas fases por diferentes condições práticas impostas sobre o processo de corte. As bobinas de aço são sujeitas a uma laminação a frio entre a primeira e a segunda etapa de corte, sendo que a máquina laminadora possui restrições na largura da sub-bobina a ser laminada, o que requer um processo de corte em duas fases.

Alguns estudos (FERREIRA *et al,* 1990; VALÉRIO DE CARVALHO, RODRIGUES, 1994; HOTO, 1996; MARQUES, 2000) propuseram modelos e heurísticas para gerar padrões de corte no Problema de Corte das Bobinas de Aço. Em Hoto (1996) é apresentado um modelo com a ideia de incluir “compartimentos” no padrão, que representam as sub-bobinas a serem laminadas. Estes compartimentos são formados por itens que deverão sofrer o mesmo processo de laminação a frio.

Sendo assim, o Problema da Mochila Compartimentada pode ser entendido como o de escolher itens a serem incluídos em uma mochila com a necessidade de se compartimentar a mochila, de forma a inserir itens que possuem determinada peculiaridade num mesmo compartimento.

No Capítulo 1, são apresentados problemas clássicos de mochila, com o propósito de criar uma base para o conteúdo a ser trabalhado. É discutida, neste capítulo, a complexidade destes problemas e alguns métodos de solução.

No Capítulo 2 são enunciadas várias aplicações desses problemas e associados, com o objetivo de justificar o estudo desta classe de problemas. Dá-se ênfase, nas aplicações, ao Problema de Corte de Estoque Unidimensional com Estoque Homogêneo, considerado uma base para o entendimento de problemas de corte em duas fases.

No Capítulo 3, o Problema de Corte das Bobinas de Aço (um caso especial de corte em duas fases) é utilizado para a formulação do modelo original (não-linear) do Problema da Mochila Compartimentada e do Problema da Mochila Compartimentada Restrito.

No Capítulo 4, apresenta-se o primeiro modelo linear proposto por Hoto *et al* (2006) para o Problema da Mochila Compartimentada Restrito e prova-se que o mesmo possui falhas. Por meio de um contra-exemplo, justifica-se que a formulação não é equivalente à original, de forma que o próprio contra-exemplo aponta a mudança que deve ser realizada no modelo.

No Capítulo 5, apresenta-se um novo modelo linear para o problema e uma demonstração de que ele é equivalente ao modelo original, juntamente com o resultado de ensaios numéricos.

A revisão teórica é disposta no trabalho de acordo com seu conteúdo específico. A seção 1.3 do Capítulo 1 apresenta uma breve revisão bibliográfica da resolução de alguns problemas de mochila e do *bin-packing*, juntamente com algoritmos para resolvê-los, onde um algoritmo *branch-and-bound* é apresentado de maneira detalhada. No Capitulo 2 apresenta-se uma revisão bibliográfica de aplicações de problemas de mochila. No Capítulo 3, seção 3.2, há uma revisão dos trabalhos relacionados ao Problema da Mochila Compartimentada, e de trabalhos anteriores que motivaram esta versão de mochila.

O trabalho foi escrito objetivando-se preencher os pré-requisitos para a abordagem principal (o Problema da Mochila Compartimentada), ou seja, procurando-se deixá-lo completo e didático para facilitar o entendimento de um aluno de pós-graduação. Com este propósito, problemas mais simples de mochila são descritos no Capítulo 1, o método de Geração de Colunas é descrito no Capítulo 2, seção 2.2.1, e complementado no Apêndice A, além de ser feita uma revisão básica a respeito de complexidade computacional no Capítulo 1, seção 1.2, e descritas heurísticas com detalhes ao longo do trabalho.

O Apêndice B apresenta uma tentativa de remodelagem no problema, por meio da teoria de grafos. O modelo foi construído durante a realização deste trabalho, e seu estudo não foi levado adiante por ter gerado tempos computacionais inferiores ao modelo linear proposto no Capítulo 5 em alguns ensaios numéricos realizados.

O Apêndice C contém ensaios de pesquisas futuras.

Durante a realização deste trabalho, um artigo foi escrito e à língua inglesa, com o propósito de ser submetido a uma revista internacional. A ênfase do artigo está na prova de que o Problema da Mochila Compartimentada é Linear. O esboço (em português) deste artigo encontra-se no Anexo A, formatado com as normas da revista Mathematics of Operations Research, do instituto Informs®.

CAPÍTULO 1

PROBLEMAS DE MOCHILA

Uma mochila é uma espécie de saco que serve para uma pessoa carregar itens pessoais. Problemas de mochila envolvem a escolha de itens a serem colocados em uma ou mais mochilas, buscando-se maximizar uma função que representa determinada utilidade (ARENALES *et al*, 2007).

Possivelmente, a primeira menção ao Problema da Mochila foi feita por Dantzig (1957), que apresenta o seguinte exemplo: uma pessoa está planejando uma viagem e decidiu não carregar mais que 70 *libras* de diferentes itens, tais como roupas de cama, latas de comida, etc. Cada item possui um peso e um valor de utilidade determinado. O objetivo da pessoa é levar consigo itens numa mochila que deverá ser a “mais útil possível” (soma linear das utilidades dos itens selecionados). Esta ideia ilustra o Problema da Mochila 0-1 que será formalizado na subseção 1.1.1.

1.1 Alguns Problemas de Mochila e Afins

Conforme a ideia de problema de mochila apresentada por Dantzig (1957), a qual cada item possui um valor e um peso, neste trabalho serão usadas as palavras *utilidade* e *peso*, que representam as quantidades na qual cada item irá contribuir na função-objetivo e ocupar no interior da mochila, respectivamente.

Alguns problemas clássicos serão abordados nesta seção, com o objetivo de introduzir o leitor ao Problema da Mochila Compartimentada, que pode ser considerado um problema desta “família” de problemas. O Problema de Múltiplas Mochilas 0-1 e o Problema da Designação Generalizada serão explorados para servir a este propósito.

Na seção 1.3 e no Capítulo 2 estarão em evidência os outros problemas aqui enunciados: O Problema da Mochila 0-1, o Problema da Mochila Restrito (e Irrestrito), e o *Bin-packing*.

1.1.1 O Problema da Mochila 0-1

Conforme modelou Dantzig (1957), considere n itens, sendo l_j o peso (em libras) do j-ésimo item e u_j o valor (utilidade) de tal item, e considere a

variável de decisão x_j, onde $x_j = 1$ se o j-ésimo item é selecionado, e $x_j = 0$ no caso contrário. Considere também L sendo a capacidade da mochila (que no exemplo de Dantzig (1957) vale 70 *libras*). Assim, de acordo com a terminologia apresentada em Martello e Toth (1997), o *Problema da Mochila 0-1* consiste no modelo (1.1)-(1.3):

Maximizar:

$$\sum_{j=1}^{n} u_j x_j \tag{1.1}$$

sujeito a:

$$\sum_{j=1}^{n} l_j x_j \leq L \tag{1.2}$$

$$x_j \in \{0,1\}. \tag{1.3}$$

Uma vez que o objetivo do problema é maximizar a utilidade dos itens a serem introduzidos na mochila, a equação (1.1) representa a soma das utilidades de tais itens.

A equação (1.2) é conhecida como *restrição física* ou *de mochila*, pois garante que a capacidade física da mochila não seja desrespeitada, ou seja, a soma dos pesos dos itens a compor a mochila não deve ultrapassar o valor L.

Observe que se $\sum_{j=1}^{n} l_j \leq L$, então a solução é trivial, $x_j = 1$ para todo $j = 1,\ldots,n$. Por isso é válido supor que $\sum_{j=1}^{n} l_j > L$. Outra suposição óbvia feita em problemas de mochila é que o peso de cada item não ultrapassa a capacidade da mochila, $l_j \leq L$ para todo $j = 1,\ldots,n$.

O caso particular em que $u_j = l_j$ para todo $j = 1,\ldots,n$ é chamado de *Problema da Soma de Subconjuntos*.

Segundo Martello e Toth (1997), o Problema da Mochila 0-1 é importante porque pode ser considerado o problema mais simples de programação inteira, além de surgir como subproblema para problemas mais complexos, como o PCE (GILMORE; GOMORY, 1961) e o *bin-packing* (MARTELLO; TOTH, 1997), e representar diversas situações práticas, as quais algumas serão descritas no Capítulo 2.

O Problema da Mochila 0-1 é um caso particular do Problema da Mochila Restrito (que será enunciado a seguir na subseção 1.1.2). Assim, se o problema binário for tratado como um problema de mochila restrito, este pode ser resolvido utilizando-se o algoritmo *Backtracking,* a ser apresentado na subseção 1.3.1.

1.1.2 O Problema da Mochila Restrito (e Irrestrito)

O Problema da Mochila 0-1 pode ser generalizado assumindo que para cada $j = 1,\ldots,n$, d_j itens de utilidade u_j e peso l_j são considerados. A letra d remete à demanda, uma vez que em algumas situações práticas, há uma demanda de cada item e é importante não escolher mais itens do que este valor para não causar superprodução. Assim, o *Problema da Mochila Restrito* é definido por:

Maximizar:

$$\sum_{j=1}^{n} u_j x_j \tag{1.4}$$

sujeito a:

$$\sum_{j=1}^{n} l_j x_j \leq L \tag{1.5}$$

$$0 \leq x_j \leq d_j, \text{ para } j = 1,\ldots,n, \tag{1.6}$$

$$x_j \text{ inteiro, para } j = 1,\ldots,n. \tag{1.7}$$

A diferença para o Problema da Mochila 0-1 está na restrição (1.6), juntamente com a definição das variáveis em □, o que limita o valor de x_j a valores inteiros entre 0 e d_j, não mais entre 0 e 1, logo a região viável é mais abrangente.

Caso não haja uma limitação para a quantidade de cada item, ou seja, podem ser inseridos quantos itens forem desejados de cada índice $j = 1,\ldots,n$, que também seria dizer que $d_j = \infty$, tem-se o *Problema da Mochila Irrestrito*, expresso por:

Maximizar:

$$\sum_{j=1}^{n} u_j x_j \tag{1.8}$$

sujeito a:

$$\sum_{j=1}^{n} l_j x_j \leq L \tag{1.9}$$

$$x_j \geq 0 \text{, para } j = 1,\ldots,n, \tag{1.10}$$

$$x_j \text{ inteiro, para } j = 1,\ldots,n. \tag{1.11}$$

Este caso é considerado quando não há limitação para itens em estoque, desde que respeitada a capacidade da mochila, ou seja, poderia ser considerado $d_j = \left\lfloor L/l_j \right\rfloor$*, pois a restrição de mochila implica que não podem ser inseridos itens de peso l_j em quantidade maior a esse valor.

O Problema da Mochila Irrestrito será usado na construção do algoritmo *Backtracking*, que pode ser modificado para resolver o Problema da Mochila Restrito e também o Problema da Mochila 0-1.

1.1.3 O Problema com Múltiplas Mochilas

Outra possível generalização do Problema da Mochila 0-1 ocorre quando considera-se mais de uma mochila para alocar os itens. Nos problemas de corte de estoque, é comum as barras disponíveis a serem cortadas em itens menores terem larguras diferentes. Neste caso, um problema que gera “padrões de corte” deve considerar padrões em diferentes barras, ao tempo que, um problema de múltiplas mochilas deve considerar alocações de itens em diferentes mochilas.

Se consideradas m mochilas de capacidade L_i $(i = 1,\ldots,m)$, e n itens de peso l_j e utilidade u_j $(j = 1,\ldots,n)$, o *Problema de Múltiplas Mochilas 0-1* consiste em maximizar a soma das utilidades dos itens a serem escolhidos, respeitando as capacidades das mochilas. Para isso, considere as variáveis de decisão x_{ij} $(i = 1,\ldots,m; j = 1,\ldots,n)$ com valor 1 se o item j for incluído na mochila i, e 0 no caso contrário.

---

* $\lfloor x \rfloor$ denota o maior número inteiro menor ou igual a x, que no caso de x ser positivo pode ser entendido como a parte inteira de x.

A soma das utilidades dos itens que serão incluídos nas mochilas é dada pela expressão $\sum_{i=1}^{m}\sum_{j=1}^{n} u_j x_{ij}$. Para cada mochila, há uma restrição de capacidade, pois na mochila de índice i a soma dos pesos dos itens que irão compô-la não pode ultrapassar L_i, logo $\sum_{j=1}^{n} l_j x_{ij} \leq L_i$, para cada $i = 1,\ldots,m$. Também um item pode compor apenas uma mochila, então para cada $j = 1,\ldots,n$ deve valer $\sum_{i=1}^{m} x_{ij} \leq 1$.

Sendo assim, o modelo para o Problema de Múltiplas Mochilas 0-1 é o seguinte:

Maximizar:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} u_j x_{ij} \tag{1.12}$$

sujeito a:

$$\sum_{j=1}^{n} l_j x_{ij} \leq L_i, \text{ para } i = 1,\ldots,m, \tag{1.13}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \text{ para } j = 1,\ldots,n, \tag{1.14}$$

$$x_j \in \{0,1\}, \text{ para } i = 1,\ldots,m,\ j = 1,\ldots,n. \tag{1.15}$$

Assim como no Problema da Mochila 0-1, algumas suposições são feitas no Problema de Múltiplas Mochilas 0-1. Primeiramente, $\max_{j=1,\ldots,n}\{l_j\} \leq \max_{i=1,\ldots,m}\{L_i\}$, pois todos os itens devem possuir peso menor ou igual que a capacidade de pelo menos uma mochila.

Outra suposição feita é que $\min_{j=1,\ldots,n}\{l_j\} \leq \min_{i=1,\ldots,m}\{L_i\}$, isso garante que todas as mochilas recebam ao menos um item. De fato, se alguma mochila possui capacidade menor que o peso do menor item, e dessa forma possui capacidade menor que o peso de cada item, nenhum item poderá compor essa mochila, e ela pode ser trivialmente descartada do problema.

1.1.4 O Problema da Designação Generalizada

Embora o Problema da Designação Generalizada não seja propriamente um problema de mochila, este incorpora similaridades com os problemas de mochila, tal como será discutido sua relação com o problema de múltiplas mochilas. Além disso, conforme coloca Martello e Toth (1997), problemas de mochila compõem uma chave central para algoritmos que o resolvem. E ainda o Problema da Designação Generalizada pode ser descrito com a terminologia adotada neste trabalho para os problemas de mochila, como será modelado a seguir.

Em uma possível formulação para o problema, suponha que n tarefas precisam ser realizadas, indexadas em $\{1,\ldots,n\}$, e há m pessoas disponíveis para realizar as tarefas, indexadas em $\{1,\ldots,m\}$. Precisa-se designar quais pessoas farão quais tarefas. São conhecidos valores que representam as vantagens de se designar cada pessoa a cada tarefa, e o tempo necessário que cada pessoa precisa para realizar cada tarefa. Considere u_{ij} a vantagem e l_{ij} o tempo que a pessoa de índice i precisa para realizar a tarefa de índice j, e L_i o tempo máximo disponível à pessoa de índice i para trabalhar nas tarefas, onde $i \in \{1,\ldots,m\}$ e $j \in \{1,\ldots,n\}$.

As incógnitas do problema são x_{ij} $(i = 1,\ldots,m,\ j = 1,\ldots,n)$, onde $x_{ij} = 1$ se a tarefa de índice j foi designada à pessoa de índice i, e $x_{ij} = 0$ no caso contrário. Como cada tarefa deve ser designada a uma e somente uma pessoa, dado $j^* \in \{1,\ldots,n\}$ deve-se ter $\sum_{i=1}^{m} x_{ij^*} = 1$, garantindo assim que para algum $i^* \in \{1,\ldots,m\}$ vale $x_{i^* j^*} = 1$ e $x_{ij^*} = 0$ para todo $i \neq i^*$. Além disso, o tempo máximo disponível de cada pessoa deve ser respeitado, impondo que a soma dos tempos gastos para realizar as tarefas designadas a determinada pessoa não extrapole o tempo disponível da pessoa, assim $\sum_{j=1}^{n} l_{ij} x_{ij} \leq L_i$, para todo $i = 1,\ldots,m$.

A melhor designação é a que possui mais vantagens. Logo, deve-se maximizar $\sum_{i=1}^{m}\sum_{j=1}^{n} u_{ij} x_{ij}$, a soma linear das vantagens de cada pessoa ao realizar as

tarefas que lhe foram designadas. O modelo do Problema da Designação Generalizada é estabelecido como:

Maximizar:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} u_{ij}x_{ij} \tag{1.16}$$

sujeito a:

$$\sum_{j=1}^{n} l_{ij}x_{ij} \le L_i, \text{ para } i=1,\ldots,m, \tag{1.17}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \text{ para } j=1,\ldots,n, \tag{1.18}$$

$$x_j \in \{0,1\}, \text{ para } i=1,\ldots,m,\ j=1,\ldots,n. \tag{1.19}$$

Observe a similaridade com o Problema de Múltiplas Mochilas 0-1. O Problema da Designação Generalizada pode ser entendido como um problema de múltiplas mochilas da seguinte forma: suponha que cada item (no problema de múltiplas mochilas) possua utilidades e pesos variáveis de acordo com a mochila na qual será inserido. Dessa forma, ao invés do item de índice j ($j=1,\ldots,n$) possuir um peso l_j independente da mochila a qual for inserido, ele terá um peso l_{ij} para cada mochila de índice $i=1,\ldots,m$. Isto modifica cada restrição de mochila de $\sum_{j=1}^{n} l_j x_{ij} \le L_i$ para $\sum_{j=1}^{n} l_{ij}x_{ij} \le L_i$ e a função-objetivo de $\sum_{i=1}^{m}\sum_{j=1}^{n} u_j x_{ij}$ para $\sum_{i=1}^{m}\sum_{j=1}^{n} u_{ij}x_{ij}$. Sendo assim, para que o problema com múltiplas mochilas torne-se idêntico ao Problema da Designação Generalizada, basta exigir que todos os itens sejam escolhidos e supor que as capacidades das mochilas sejam suficientes para isso, exigindo assim que a restrição $\sum_{i=1}^{m} x_{ij} \le 1$ seja satisfeita com igualdade.

O Problema da Designação Generalizada pode ser interpretado para dividir n trabalhos a m agentes de uma empresa, cuja realização de um trabalho gera um lucro e um recurso deve ser aplicado para determinado agente desenvolver determinado trabalho, sendo que cada agente dispõe de um recurso máximo a ser utilizado. O mesmo modelo aplica-se à situação de uma máquina que deve dividir n tarefas a m processadores, e assim por diante.

Um caso similar ao Problema da Designação Generalizada é o *Problema da Alocação de Tarefas*, que pode ser encontrado em Anton e Rorres

(2008). Neste caso, a quantidade de tarefas é igual à quantidade de agentes, e as utilidades u_{ij} valem o oposto dos pesos l_{ij}, ou seja, o objetivo é minimizar a soma dos pesos, e, além de cada tarefa dever se realizada por um único agente, cada agente deve realizar uma única tarefa. Uma aplicação pode ser dada pela seguinte situação: uma empresa dispõe de escavadeiras que estão localizadas em n cidades e devem ser utilizadas em outras n cidades. Os valores l_{ij} correspondem ao custo de transporte da escavadeira de índice i à cidade de índice j. O objetivo é minimizar o custo dos transportes, sendo que cada cidade deve ser atendida por uma escavadeira, e cada escavadeira deve atender uma única cidade.

1.1.5 O *Bin-packing*

Nesta seção será descrita a modelagem clássica do *Bin-packing Problem,* problema que será chamado de *Bin-packing* neste trabalho. Também será relacionado o *Bin-packing* com o Problema de Corte de Estoque Unidimensional, que será mais bem detalhado no Capítulo 2, seção 2.1.

O *Bin-packing* consiste em empacotar itens em mochilas (ou pacotes), respeitando sua capacidade, porém com o objetivo de alocar todos os itens, utilizando o menor número de mochilas, diferente dos problemas de mochila, que possuem o objetivo de maximizar uma utilidade que dependerá de cada item a ser escolhido.

Sejam n itens e n mochilas, onde devem ser incluídos todos os itens. Seja L a capacidade das mochilas (todas devem ter a mesma capacidade no *Bin-packing*). As incógnitas x_{ij} assumirão valor 1 se o item j for colocado na mochila i, e 0 no caso contrário.

Considere $y_i = 1$ se a mochila i for utilizada, e $y_i = 0$ no caso contrário (MARTELLO; TOTH, 1997, p. 221). Como deseja-se a quantidade mínima de mochilas a serem utilizadas, deve-se minimizar $\sum_{i=1}^{n} y_i$. Cada item deve ser incluído em uma e apenas uma mochila, logo $\sum_{i=1}^{n} x_{ij} = 1$ para cada item $j = 1,\ldots,n$.

Para cada pacote $i \in \{1,\ldots,n\}$ utilizado deve-se satisfazer a restrição de mochila $\sum_{j=1}^{n} l_j x_{ij} \leq L$. Porém, se o pacote i não é utilizado, os elementos x_{ij} devem ser nulos, pois nenhum item irá compor a mochila i. Pode-se combinar essas duas exigências com a restrição $\sum_{j=1}^{n} l_j x_{ij} \leq L y_j$, para cada $i \in \{1,\ldots,n\}$.

O modelo apresentado em Martello e Toth (1997) é:

Minimizar:

$$\sum_{i=1}^{n} y_i \tag{1.20}$$

sujeito a:

$$\sum_{j=1}^{n} l_j x_{ij} \leq L y_i, \text{ para } i = 1,\ldots,n, \tag{1.21}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \text{ para } j = 1,\ldots,n, \tag{1.22}$$

$$y_j \in \{0,1\}, \text{ para } j = 1,\ldots,n. \tag{1.23}$$

$$x_{ij} \in \{0,1\}, \text{ para } i, j = 1,\ldots,n. \tag{1.24}$$

onde

$$y_i = \begin{cases} 1, & \text{se a mochila } i \text{ é usada;} \\ 0, & \text{caso contrário.} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{se o item } j \text{ é colocado na mochila } i; \\ 0, & \text{caso contrário.} \end{cases}$$

Deve-se assumir que $l_j \leq L$ para todo $j = 1,\ldots,n$, pois caso algum item de índice j viole esta desigualdade, o problema é infactível, uma vez que a restrição de mochila irá impor $x_{ij} = 0$ para todo i (o que significa que o item em questão não entrará em nenhuma mochila), e torna-se impossível $\sum_{i=1}^{n} x_{ij}$ assumir valor 1 (que deveria ocorrer porque no *Bin-packing* todo item deve ser incluído a alguma mochila).

O *Bin-packing* pode ser visto como um caso particular do modelo de Kantorovich para o problema de corte de estoque, que será desenvolvido e melhor

delineado no Capítulo 2. O modelo atribuído à Kantorovich (1997 apud VALÉRIO DE CARVALHO, 2002, p. 255) para o PCE é como exibido em (1.25)-(1.29).

Minimizar:

$$\sum_{j=1}^{n} y_j \tag{1.25}$$

sujeito a:

$$\sum_{i=1}^{m} l_i x_{ij} \leq L y_j, \text{ para } j = 1,\ldots,n, \tag{1.26}$$

$$\sum_{j=1}^{n} x_{ij} \geq d_i, \text{ para } i = 1,\ldots,m, \tag{1.27}$$

$$y_j \in \{0,1\}, \text{ para } j = 1,\ldots,n, \tag{1.28}$$

$$x_{ij} \geq 0 \text{ e inteiro, para } i = 1,\ldots,m,\ j = 1,\ldots,n. \tag{1.29}$$

As constantes d_i indicam a demanda de um item no problema de corte, e podem ser consideradas com valor 1 num caso particular. Além disso, tome $m = n$. Observe também que substituir a restrição (1.27) por $\sum_{j=1}^{n} x_{ij} = d_i$ não altera a solução do problema, pois zerar elementos x_{ij} não invalida (1.26) e não acrescenta valores na função-objetivo (que deve ser minimizada). Isso transforma a restrição (1.27) em $\sum_{j=1}^{n} x_{ij} = 1$, fazendo com que $x_{ij} \in \{0,1\}$. Basta então inverter a ordem da indexação nas variáveis x_{ij} para que o caso particular do modelo de Kantorovich construído dessa maneira torne-se exatamente o *Bin-packing.*

1.2 Complexidade dos Problemas de Mochila

Nesta seção faz-se um breve comentário a respeito da complexidade computacional dos problemas apresentados, começando por uma revisão básica dos conceitos utilizados para classificar os problemas, e posteriormente apresentando a classificação. Para mais detalhes da teoria de complexidade computacional recomenda-se a leitura de Nemhauser e Wolsey (1988, p. 114-142), apenas no conceito de fortemente NP-difícil utilizou-se como referência Goodrich (2002).

O tempo de execução de um algoritmo é medido considerando-se o número de operações elementares necessárias para sua execução na pior hipótese (dependendo dos exemplares do problema) e o tamanho da entrada de dados (quantidade de *bits* requeridos para descrever a entrada), assim a máquina usada não interfere na avaliação do algoritmo. Um algoritmo tem tempo $O(f(n))$ quando existe uma constante positiva k e um instante $n_0 \in \square$, tal que para todo inteiro $n > n_0$ tem-se $k \cdot f(n) \geq T(n)$, onde T é uma função que relaciona a cada tamanho de entrada n o pior caso de tempo $T(n)$ para determinado algoritmo. Essa definição considera o comportamento da função apenas para valores com $n \to \infty$ e desconsidera constantes. Por exemplo, se o tempo requisitado por um algoritmo em todas as entradas de tamanho n é no máximo $4n^3 + 5n^2 + n$, diz-se que o algoritmo tem tempo $O(n^3)$.

Se o tempo de um algoritmo é $O(f(n))$ e $f(n)$ é um polinômio, diz-se que o algoritmo possui tempo polinomial, se $f(n)$ representa uma função exponencial, o algoritmo tem tempo exponencial, e analogamente, para outras classes de funções. A rigor, um algoritmo dito de tempo exponencial possui tempo controlado ao menos exponencialmente. Analogamente, um algoritmo dito de tempo polinomial é controlado ao menos polinomialmente, logo também é controlado exponencialmente e pode ser dito de tempo exponencial.

O conjunto P é formado pelos problemas que possuem um algoritmo de tempo polinomial que o resolve ou verifica que não há solução.

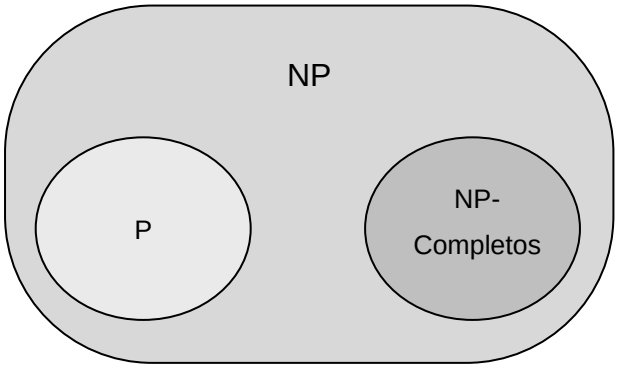
Definindo o conjunto NP de maneira breve, trata-se do conjunto de problemas na qual a factibilidade de uma possível solução de um exemplar pode ser verificada em tempo polinomial. Supondo que um problema esteja em NP, mas não em P, sua solução ótima não pode ser encontrada em tempo polinomial, embora se possa verificar se uma solução é factível em tempo polinomial. O nome NP refere-se a *Nondeterministic Polynomial,* sendo que um algoritmo não-determinístico gera uma solução num primeiro estágio e verifica se ela é factível no segundo estágio, se for, a saída do algoritmo diz que o problema é factível. Este algoritmo será polinomial se o tempo de verificação no segundo estágio for polinomial.

O conjunto NP é importante porque constitui a maior parte dos problemas comumente estudados, de fato, todos os problemas enunciados neste trabalho pertencem à NP. Por exemplo, no Problema da Mochila Irrestrito, dados

valores gerados aleatoriamente de $x_j \in \mathbb{R}_+$, para $j = 1,\ldots,n,$ verificar se tal solução é factível (se satisfaz a restrição de mochila, neste caso) requer n operações de multiplicação, $n-1$ operações de soma e a comparação do resultado com L, logo isso pode ser feito em tempo polinomial (e linear) $O(n)$. A prova de que os outros problemas mencionados estão em NP é análoga.

Todo problema em P pertence à NP, pois o algoritmo polinomial que resolve um problema ou retorna que não há solução pode ser usado como não-determinístico. Não há uma resposta satisfatória na literatura para a questão se P=NP. Woeginger (2015) fez uma lista de 107 trabalhos na literatura que tentaram (e falharam em) mostrar a prova de que P=NP, P≠NP ou que a questão é injustificável. Diante dessa indagação, surge a ideia de pensar na classe dos problemas mais difíceis de NP, os NP-completos, que possuem a propriedade de que se ao menos um problema NP-completo for polinomial (pertence à P), então P=NP. Caso P≠NP, o diagrama de Venn da Figura 1 ilustra a relação entre os conjuntos.

Figura 1: Caso P não contenha NP



Fonte: Nemhauser e Wolsey (1988, p. 138)

Para definir os problemas NP-completos é preciso o conceito de redução polinomial. Um problema X_1 é *polinomialmente redutível* a outro problema X_2 se existe uma função g que relaciona a cada exemplar de X_1 um exemplar de X_2, sendo que $d \in D(g)$ [†] é factível em X_1 se, e somente se $g(d)$ é factível em X_2, e g deve ser computável em tempo polinomial. Isto significa que o problema X_1 pode ser

[†] $D(g)$ denota o domínio da função g.

resolvido resolvendo-se X_2, exigindo-se um tempo polinomial para a conversão, o que dá a noção intuitiva de que o problema X_1 não é mais difícil que X_2.

Um problema X é NP-difícil se todos os problemas de NP são polinomialmente redutíveis a X, intuitivamente, todos os problemas de NP possuem a propriedade de não serem mais difíceis do que X. O problema X é NP-completo se pertence a NP e é NP-difícil. Há ainda o caso particular dos problemas que são fortemente NP-difíceis, as quais permanecem NP-difíceis ainda que o valor de cada entrada passe a ser limitado a um polinômio do tamanho da entrada.

A ideia intuitiva da redução polinomial diz que os problemas em P são os mais “fáceis” de NP, os NP-completos mais “difíceis”, e os NP-completos que são também fortemente NP-difíceis são ainda mais. A menos que P=NP.

Considere o problema da partição: dados n inteiros positivos $l_1,\ldots,l_n$, existe um subconjunto $S \subseteq N = \{1,\ldots,n\}$ tal que $\sum_{j\in S} l_j = \sum_{j\in N-S} l_j$? Este é um problema básico NP-completo, e que Martello e Toth (1997, p. 6) prova ser polinomialmente redutível ao Problema da Soma de Subconjuntos, caso particular do Problema da Mochila 0-1 e do Problema da Mochila Restrito, logo esses problemas são todos NP-completos.

Analogamente, Martello e Toth (1997, p. 8) usa um problema de partição diferente, a qual é fortemente NP-difícil, e que prova ser pseudo-polinomialmente redutível ao Problema de Múltiplas Mochilas 0-1, para provar que este é também fortemente NP-difícil, juntamente com o caso mais geral do Problema da Designação Generalizada. O problema de partição em questão é chamado de 3-PARTITION, e pode ser descrito como: dados $n = 3m$ inteiros positivos $l_1,\ldots,l_n$, satisfazendo $\sum_{j=1}^{n} l_j / m = B$ inteiro e $B/4 < l_j < B/2$ para todo $j = 1,\ldots,n$, existe uma partição de $N = \{1,\ldots,n\}$ em m subconjuntos $S_1,\ldots,S_m$ (cada um contendo três elementos de N) tal que $\sum_{j\in S_i} l_j = B$ para todo $i = 1,\ldots,m$?

Foi usado que se um problema é fortemente NP-difícil e pode ser reduzido pseudo-polinomialmente a outro, este outro também é fortemente NP-difícil, sendo que um algoritmo pseudo-polinomial é um algoritmo executado com tempo polinomial com relação ao valor das entradas, e não ao número de bits requeridos para descrevê-la.

Para provar que o *Bin-packing* é fortemente NP-difícil, Martello e Toth (1997, p. 9) considera o problema de decisão R(*Bin-packing*), descrito por: dados $n+2$ inteiros positivos $l_1,\ldots,l_n,c$ e a, existe uma partição de $N=\{1,\ldots,n\}$ em a subconjuntos $S_1,\ldots,S_a$ tal que $\sum_{j\in S_i} l_j \le c$ para todo $i=1,\ldots,a$? É fácil ver que a resolução do *Bin-packing* (problema de otimização) resolve o R(*Bin-packing*) (problema de decisão), basta verificar se a é maior ou igual ao valor ótimo da solução ótima do *Bin-packing*. Agora, observe que um exemplar qualquer do 3-PARTITION pode ser transformado pseudo-polinomialmente em um equivalente do R(*Bin-packing*) simplesmente escolhendo $c=B$ e $a=m$, pois exigir que $\sum_{j\in S_i} l_j \le c=B$ para todo $i=1,\ldots,a=m$ é o mesmo que exigir que $\sum_{j\in S_i} l_j = B$ para todo $i=1,\ldots,m$, uma vez que se a desigualdade for estrita para algum $i\in\{1,\ldots,m\}$, tem-se $\sum_{j=1}^{n} l_j = \sum_{i=1}^{m}\sum_{j\in S_i} l_j < mB$, o que é um absurdo, pois $\sum_{j=1}^{n} l_j = mB$. Portanto o *Bin-packing* é um problema fortemente NP-difícil.

Sendo o *Bin-packing* um caso particular do problema de corte de estoque, tal problema também é fortemente NP-difícil.

Considerando a complexidade dos problemas abordados, todos NP-completos, é inesperado que haja algoritmos exatos que os resolvam em tempo polinomial. Isso justifica alguns deles, tal como o *Bin-packing*, serem tão estudados por meio de heurísticas que buscam encontrar uma solução satisfatoriamente próxima da ótima, em um tempo viável de execução. O mesmo se aplica ao Problema da Mochila Compartimentada, a ser enunciado no Capítulo 3, comumente abordado por heurísticas.

1.3 SOLUÇÕES E ALGORÍTIMOS

Nesta seção serão abordadas algumas maneiras comumente utilizadas para resolver o Problema da Mochila 0-1, o Problema da Mochila Restrito, e o Problema da Mochila Irrestrito, e citadas heurísticas que se apresentam como alternativa para encontrar soluções aproximadas do *Bin-packing.* Ao discutir a resolução de cada problema, será feita uma revisão teórica sobre seu estudo.

Primeiramente observe que alguns desses problemas podem ser resolvidos sendo reduzidos a outros, apesar de ser preciso considerar um tempo computacional para conversão. Embora o Problema da Mochila 0-1 seja claramente

um caso particular do Problema da Mochila Restrito, este segundo também pode ser resolvido a partir de uma solução do primeiro. De fato, dado um exemplar do Problema da Mochila Restrito, cada item i, que deve figurar no máximo d_i vezes na mochila, pode ser transformado em $\lceil \log_2(d_i+1) \rceil$‡ itens de utilidade e peso (u_i, l_i), $(2u_i, 2l_i)$, $(4u_i, 4l_i)$, ... , $(2^{\lfloor \log_2 d_i \rfloor - 1}u_i, 2^{\lfloor \log_2 d_i \rfloor - 1}l_i)$ e $((d_i - 2^{\lfloor \log_2 d_i \rfloor}+1)u_i, (d_i - 2^{\lfloor \log_2 d_i \rfloor}+1)l_i)$, todos com demanda igual a 1. A utilidade e peso do último item da lista completa a sequência de forma que a soma das utilidades e pesos dos itens gerados para o Problema da Mochila 0-1 seja igual à $u_i d_i$ e $l_i d_i$, respectivamente.

Agora, focando no Problema da Mochila 0-1, sua resolução foi inicialmente investigada por Dantzig (1957), que apresentou a solução do problema relaxado (em que substitui-se $x_j \in \{0,1\}$ por $0 \le x_j \le 1$) e um método que usa a solução do problema relaxado e acrescenta restrições para modificá-la e obter uma solução inteira.

O algoritmo de Dantzig (1957) não será apresentado aqui, mas quanto à solução do problema relaxado, foi apresentada uma resolução geométrica que é equivalente a reordenar os itens de forma a obter $\frac{u_1}{l_1} \ge \frac{u_2}{l_2} \ge \ldots \ge \frac{u_n}{l_n}$ (ver a subseção 1.3.1 para uma motivação da reordenação) e encontrar o “item crítico” de índice k tal que $k = \min\left\{ j \in \mathbb{N} \,/\, \sum_{i=1}^{j} l_i > L \right\}$, fazendo $x_j = 1$ para $j < k$, $x_j = 0$ para $n \ge j > k$ e $x_k = \frac{L - \sum_{i=1}^{k-1} l_i}{l_k}$, ou seja, inserem-se os $k-1$ primeiros itens na mochila e a completa com o que ainda for possível de x_k.

Kolesar (1967) propôs um algoritmo *branch-and-bound* para resolver o Problema da Mochila 0-1, que consiste em separar as possíveis soluções em classes organizadas na forma de uma árvore (gerando “ramos”), e a cada estágio dos ramos limitantes superiores são calculados com o objetivo de se fazer “podas” nos ramos onde não há esperança de encontrar uma solução melhor que as já

---

‡ $\lceil x \rceil$ denota o menor número inteiro maior ou igual a x.

analisadas. Esta ideia será explorada no algoritmo *Backtracking* apresentado na seção 1.3.1 para resolver os problemas de mochila restrito e irrestrito.

Contudo, diferentemente do *Backtracking* a ser apresentado neste trabalho, o algoritmo de Kolesar (1967) inicialmente não ordena as variáveis e resolve um subproblema com algumas variáveis relaxadas para encontrar os limitantes inferiores. No subproblema as variáveis são ordenadas. A cada "nó" é avaliado se um item entra ou não na mochila, e não há o "backtrack" (volta) pelos nós, pois uma vez avaliado, um ramo é podado e o outro segue sendo desenvolvido na próxima iteração, ou seja, a busca é em largura e não em profundidade.

A poda num algoritmo evidencia a importância do estudo de limitantes superiores. A solução do problema relaxado dada em Dantzig (1957) é obviamente um limitante superior para o problema, uma vez que o conjunto de soluções factíveis da relaxação é maior e o problema é de maximização. Martello e Toth (1977) apresentou o primeiro limitante superior melhor ou igual ao de Dantzig.

Balas e Zemel (1980) exploraram a observação de Dantzig (1957) de que a solução do problema relaxado é, em geral, próxima da solução ótima, e elaboraram um algoritmo que busca em apenas alguns itens, com índices próximos ao do item crítico, encontrar a solução ótima após obter a solução do problema relaxado. Ao problema de otimização reduzido a alguns itens foi dado o nome de *core problem*.

Mais detalhadamente, considere que o problema esteja posto de forma que $\frac{u_1}{l_1} \geq \frac{u_2}{l_2} \geq \ldots \geq \frac{u_n}{l_n}$, e seja $C = \{j_1, \ldots, j_2\}$ o conjunto nomeado de *core* do problema, onde $j_1 = \min\{j \mid x_j = 0\}$ e $j_2 = \max\{j \mid x_j = 1\}$. Dessa forma, na solução ótima do problema, se $j < j_1$ então $x_j = 1$ e se $j > j_2$ então $x_j = 0$, logo basta decidir o valor das variáveis entre j_1 e j_2, o que significa resolver o problema:

Maximizar:

$$\sum_{j \in C} u_j x_j \tag{1.30}$$

sujeito a:

$$\sum_{j \in C} l_j x_j \leq L - \sum_{j=1}^{j_1 - 1} l_j \tag{1.31}$$

$$x_j \in \{0,1\} \text{ para } j \in C . \quad (1.32)$$

Assim reduz-se o problema inicial a um com menos variáveis. Na verdade, para problemas de muitas variáveis, o *core problem* possui como quantidade de variáveis uma parcela muito pequena de n (BALAS; ZEMEL, 1980, p. 1131; MARTELLO; TOTH, 1997, p. 58). Embora o conjunto C só possa ser determinado com a resolução do problema, Balas e Zemel (1980) afirmou que uma aproximação satisfatória pode ser encontrada resolvendo a relaxação linear, e desenvolveu um algoritmo que encontra a solução do problema relaxado em tempo linear, ou seja, mais rápido que a resolução de Dantzig (1957).

Uma vez que o *core* está centrado no item crítico, a resolução do problema relaxado, que encontra o item crítico, é usada para determinar uma aproximação do conjunto *core* relacionado ao problema, por meio de algum critério adotado. Então o *core problem* é resolvido. Balas e Zemel (1980) o resolve com um algoritmo que usa procedimentos heurísticos e de redução (que procura reduzir mais o número de variáveis do problema).

Fayard e Plateau (1982, apud MARTELLO; TOTH, 1997, p. 22-23, 60-61) também desenvolveu um algoritmo para o *core problem*, além de propor um limitante superior melhor ou igual que o de Martello e Toth (1977, apud MARTELLO; TOTH, 1997, p. 20), obtido como o máximo entre dois casos da solução do problema relaxado, um com a restrição adicional $x_k = 1$ e o outro com a restrição adicional $x_k = 0$, onde k é o índice do item crítico do problema original relaxado.

Muitos outros estudos importantes foram realizados para o Problema da Mochila 0-1. Pode ser citado, por exemplo, Ingargiola e Korsh (1973, apud MARTELLO; TOTH, 1997, p. 14, 45), que apresentou o primeiro procedimento de redução, a qual busca reduzir o número de variáveis do problema, para isso procura determinar itens que obrigatoriamente são inseridos na mochila na solução ótima, e itens que obrigatoriamente não são inseridos. Esta ideia pode ser usada numa resolução pelo *core problem*.

Considerando agora o Problema da Mochila Restrito, foi visto que ele pode ser reduzido ao Problema de Mochila 0-1, todavia a literatura desses problemas não é exatamente a mesma.

Ingargiola e Korsh (1977, apud MARTELLO; TOTH, 1997, p. 88) apresentou um procedimento de redução para o Problema da Mochila Restrito tal como o apresentado pelos mesmos autores para o Problema da Mochila 0-1.

Na busca por uma solução exata para o Problema da Mochila Restrito e o Problema da Mochila Irrestrito, encontram-se estudos baseados em programação dinâmica (resolver sub-rotinas recursivamente) e em métodos do tipo *branch-and-bound*.

Para esboçar uma ideia da resolução do Problema da Mochila Restrito com programação dinâmica, seja $f_i(\tilde{L})$ o valor ótimo da sub-mochila definida pelos itens $1,\ldots,i$ e capacidade da mochila $\tilde{L}$, para $i=1,\ldots,n$ e $\tilde{L}=0,1,\ldots,L$ (os pesos dos itens devem ser números inteiros). A relação de recursão que justifica a resolução por programação dinâmica é:

$$f_i(\tilde{L}) = \max \begin{cases} f_{i-1}(\tilde{L}) & \text{se } \tilde{L} \geq 0; \\ f_{i-1}(\tilde{L}-l_i)+u_i & \text{se } \tilde{L}-l_i \geq 0; \\ \vdots & \\ f_{i-1}(\tilde{L}-d_i l_i)+d_i u_i & \text{se } \tilde{L}-d_i l_i \geq 0. \end{cases} \tag{1.33}$$

Sendo assim, encontrado o valor de $f_{i-1}(\tilde{L})$ para todo $\tilde{L}=0,1,\ldots,L$, a equação (1.33) fornece o valor de $f_i(\hat{L})$ para todo $\hat{L}=0,1,\ldots,L$ em tempo $O(Ld_i)$, lembrando-se que tem-se de início:

$$f_1(\tilde{L}) = \max \begin{cases} 0 & \text{para } \tilde{L}=0,\ldots,l_1-1; \\ u_1 & \text{para } \tilde{L}=l_1,\ldots,2l_1-1; \\ \vdots & \\ d_1 u_1 & \text{para } \tilde{L}=d_1 l_1,\ldots,L. \end{cases}$$

Este recurso foi aprimorado por Gilmore e Gomory (1966, apud MARTELLO; TOTH, 1997, p. 88) e por Nemhauser e Ullmann (1969, apud MARTELLO; TOTH, 1997, p. 88). No entanto ele não funciona para problemas de tamanho grande. Nemhauser e Ullmann (1969, apud PISINGER, 1995, p. 134) reportou que o exemplar de maior tamanho resolvido considerou $n=50$ e $d_i=2$ para todo $i=1,\ldots,n$.

Segundo Pisinger (1995), o problema desses algoritmos está em não haver uma estratégia de enumeração eficiente. Pisinger propôs uma recursão à qual são considerados apenas alguns itens próximos ao item crítico, como no *core problem*.

Horowitz e Sahni (1974), em um estudo de um problema de partição onde foram apresentadas aplicações no Problema da Mochila 0-1, apresentou um algoritmo que usa suas ideias para o problema de partição e programação dinâmica. Também apresentou um algoritmo do tipo *branch-and-bound*, à qual chamou de *branch and scarch*. Este algoritmo foi modificado por Martello e Toth (1977, apud HOTO, 2001, p. 53), e dentre as modificações está um diferente limitante superior.

Em outro artigo, Martello e Toth (1977, apud MARTELLO; TOTH; 1997, p. 88) fez uma adaptação de um algoritmo *branch-and-bound* da mochila 0-1 (que pode ser encontrado em Martello e Toth (1997, p. 34-36)) para o Problema da Mochila Restrito.

No caso do Problema da Mochila Irrestrito, o limitante superior trivial, dado pela resolução do problema relaxado, é dado por $L \cdot u_1/l_1$, caso o item de índice 1 obtiver o maior valor u_i/l_i para i entre 1 e n (ocorre no caso de uma ordenação das variáveis). Martello e Toth (1997) exibiu limitantes melhores ou iguais aos estudados pelos mesmos autores, juntamente com um algoritmo guloso, que insere itens na mochila, seguindo a ordem mencionada nesta seção, até que não caiba mais nenhum (qualquer item que for inserido ultrapassa a capacidade da mochila).

Pensando em programação dinâmica, uma recursão imediata para computar a função $f_i(\tilde{L})$ (valor ótimo da sub-mochila definida pelos itens $1,\ldots,i$ e capacidade da mochila $\tilde{L}$) no Problema da Mochila Irrestrito é:

$$f_1(\tilde{L}) = \left\lfloor \frac{\tilde{L}}{l_1} \right\rfloor u_1 \text{ para } \tilde{L} = 0,\ldots,L;$$

$$f_i(\tilde{L}) = \max \begin{cases} f_{i-1}(\tilde{L}), & \text{se } \tilde{L} \geq 0; \\ f_{i-1}(\tilde{L} - l_i) + u_i, & \text{se } \tilde{L} - l_i \geq 0; \\ \vdots & \\ f_{i-1}(\tilde{L} - \left\lfloor \frac{\tilde{L}}{l_i} \right\rfloor l_i) + \left\lfloor \frac{\tilde{L}}{l_i} \right\rfloor u_i, & \text{se } \tilde{L} - \left\lfloor \frac{\tilde{L}}{l_i} \right\rfloor l_i \geq 0. \end{cases}$$

Com essa relação recursiva, o tempo para calcular $f_n(L)$ é $O(nL^2)$. Gilmore e Gomory (1965) propôs que fosse usada como relação recursiva a equação (1.34):

$$f_i(\tilde{L}) = \max \begin{cases} f_{i-1}(\tilde{L}) \\ f_i(\tilde{L} - l_i) + u_i, \text{ se } \tilde{L} - l_i \geq 0 \end{cases}, \text{ para } i > 1. \qquad (1.34)$$

Para calcular $f_n(L)$ com a relação recursiva dada em (1.34) é necessário tempo $O(nL)$. Ainda, conforme enfatiza Martello e Toth (1997), a programação dinâmica é capaz de resolver apenas exemplares de tamanho pequeno.

Gilmore e Gomory (1963), em um estudo de problemas de corte e estoque, apresenta um algoritmo *branch-and-bound* para o Problema da Mochila Irrestrito, usado na geração de colunas do problema de corte. Segundo os autores, o novo procedimento mostra-se melhor que a programação dinâmica, principalmente com relação ao tempo, algo que os autores se preocuparam, por buscar uma melhoraria no algoritmo do problema de corte que o permita ser computado em máquinas menores, mais comuns em empresas da época.

O algoritmo de Gilmore e Gomory (1963) é similar ao de Kolesar (1967) para o Problema da Mochila 0-1, conforme cita o próprio Kolesar, ao qual também se influenciou por um trabalho de Little *et al* (1963, apud KOLESAR, 1967, p. 724) sobre o problema do caixeiro viajante.

Martello e Toth (1977, apud MARTELLO; TOTH; 1997, p. 96) adaptou seu algoritmo *branch-and-bound* do Problema da Mochila 0-1 para o irrestrito. Os mesmos autores (MARTELLO; TOTH, 1997) mostraram a extensão do conceito de *core problem* ao Problema da Mochila Irrestrito, permitindo assim que exemplares de tamanho grande possam ser resolvidos. Em alguns exemplares testados, o algoritmo *branch-and-bound* sem a ideia de *core problem* foi considerado impraticável com $n > 1.000$, em outros exemplares com 90.000. Com o *core problem,* exemplares com 250.000 variáveis foram resolvidos em segundos.

Pisinger (1995) estudou procedimentos de redução do Problema da Mochila Irrestrito. Utilizou o conceito de dominância entre itens, e diferentes formas de reordená-los para, assim, criar um algoritmo melhor que os anteriores em termos de tempo.

O *Bin-packing Problem* não possui algoritmo exato satisfatório (que resolva grandes exemplares) para obtenção da solução. Os primeiros algoritmos exatos expostos na literatura são capazes de resolver apenas exemplares de tamanho pequeno. Martello e Toth (1989, apud MARTELLO; TOTH, 1997) propôs um algoritmo exato chamado MTP, baseado na estratégia *First-Fit Decreasing* (ver subseção 1.3.2). Trata-se de um algoritmo *branch-and-bound* que, assim como na

heurística FFD, é ordenado pelos itens, colocando um a um na mochila escolhida, e que conta com procedimentos de poda e um critério de dominância. Os mesmos autores também apresentaram um procedimento de redução para o problema. Porém, o algoritmo não resolve grandes exemplares, e heurísticas são necessárias de serem incluídas para diminuir o número de *backtrackings.*

Scholl, Klein e Jürgens (1997, apud ALVIM, 2004) criou um algoritmo híbrido baseado nos procedimentos de redução de Martello e Toth, combinando heurísticas e novos procedimentos de redução, além de mudar a busca que, diferente do MTP, é ordenada pelas mochilas.

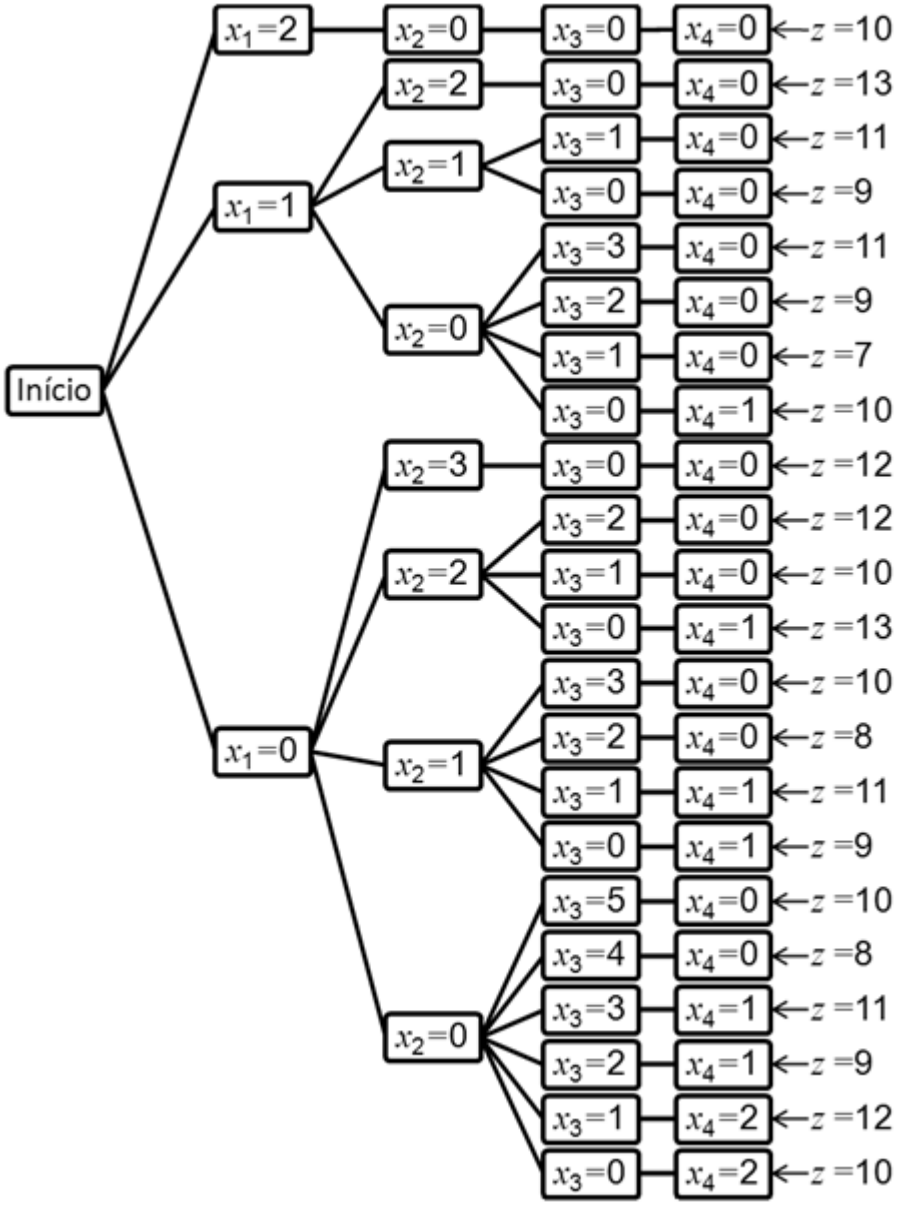
Quase toda a literatura do *Bin-packing* trata sobre heurísticas e suas performances. Algumas das mais famosas heurísticas desse problema são descritas na seção 1.3.2. Golden (1976, apud HOTO, 2001, p. 30) fez experimentos com heurísticas e constatou que na maioria dos casos encontra-se uma solução considerada satisfatória. Porém, em alguns exemplares a solução encontrada é próxima do pior caso.

Valério de Carvalho (1999) propôs um modelo para o *bin-packing* que utiliza grafos na sua formulação, e desenvolveu um algoritmo *branch-and-price* para resolvê-lo. Um modelo para o Problema da Mochila Compartimentada com esta ideia de fluxo de arcos foi desenvolvido para esta dissertação, e pode ser visto no Apêndice B. Não foi dada continuidade ao estudo desta formulação por não ter oferecido bons resultados nos ensaios numéricos, em termos de tempo computacional.

1.3.1 O Algoritmo *Backtracking*

Nesta subseção será descrito um algoritmo *branch-and-bound* para resolver o Problema da Mochila Irrestrito, e adaptado para resolver o Problema da Mochila Restrito, assim podendo também resolver o Problema da Mochila 0-1. Este algoritmo foi descrito por Gilmore e Gomory (1963). Pretendeu-se fazer uma descrição longa e detalhada para que a limitação que permite a poda de um ramo da árvore de soluções viáveis seja desenvolvida de maneira intuitiva.

Figura 2: Árvore sem podas.



Fonte: próprio autor.

A fim de introduzir a estratégia do *Backtracking*, considere o seguinte exemplo, extraído de (CHVÁTAL, 1983, p. 201):

Maximizar:

$$z = 5x_1 + 4x_2 + 2x_3 + 5x_4 \tag{1.35}$$

sujeito a:

$$51x_1 + 33x_2 + 22x_3 + 49x_4 \leq 120 \tag{1.36}$$

$$x_1, x_2, x_3, x_4 \in \mathbb{Z}_+^* \tag{1.37}$$

Se for aplicado ao exemplo (1.35)-(1.37) um algoritmo de busca que verifique (quase) todas as soluções viáveis, tem-se a árvore de enumeração esboçada na Figura 2.

Uma observação importante é a ordem no qual a busca é feita. Numa busca em profundidade, como esboça a Figura 2, os ramos são desenvolvidos na ordem que estão de cima para baixo. Primeiramente, x_1 é calculado como $\left\lfloor L/l_1 \right\rfloor$, no caso $\left\lfloor 120/51 \right\rfloor = 2$, e então para $j > 1$ calcula-se $x_j = \left\lfloor (L - \sum_{i=1}^{j-1} l_i x_i)/l_j \right\rfloor$, ou seja, seguindo a ordem das variáveis, cada item é inserido na mochila na quantidade máxima suportada.

Depois de desenvolvido o ramo, faz-se a "volta", escolhendo o maior índice k tal que $x_k \neq 0$ e colocando $x_k := x_k - 1$, no caso tem-se $k = 1$ e $x_1 := 2 - 1 = 1$, e o restante do ramo é desenvolvido da mesma maneira. Depois, a volta irá impor $k = 2$ e fazer $x_2 = 1$, e assim a terceira linha é desenvolvida. O processo continua até que a árvore esteja completa, quando $x_1 = 0$ e $k = 1$.

Determinada a ordem da busca, é necessário gravar a solução corrente, e substituí-la sempre que se encontrada uma que melhora a função-objetivo. Desta forma, o algoritmo trabalha na busca de apenas uma solução ótima para o problema, descartando as equivalentes, embora em algumas situações práticas possa ser mais interessante obter diferentes soluções para que a empresa ou indústria que lida com o problema possa tomar a decisão que lhe for conveniente. Também é o caso de, como subproblema, o problema de mochila precisar fornecer uma quantidade "z" de melhores soluções (ver a Heurística dos "z" Melhores Compartimentos no Capítulo 3, seção 3.1). Neste caso, no percorrer do algoritmo *branch-and-bound*, uma solução encontrada é gravada se for melhor que uma das "z" melhores já gravadas, e a pior dentre estas "z" que deve ser descartada.

Observe que nem todas as soluções viáveis foram construídas na Figura 2, pois nos ramos em que $x_4 > 0$ não foram exploradas outras opções para o valor desta variável. Por exemplo, quando $x_1 = 0$, $x_2 = 0$ e $x_3 = 1$, podem ser consideradas também as possibilidades $x_4 = 1$ e $x_4 = 0$. No entanto, estas possibilidades trivialmente não melhorarão o valor da função-objetivo. Por este motivo, não é necessária a redução gradativa dos valores da última variável.

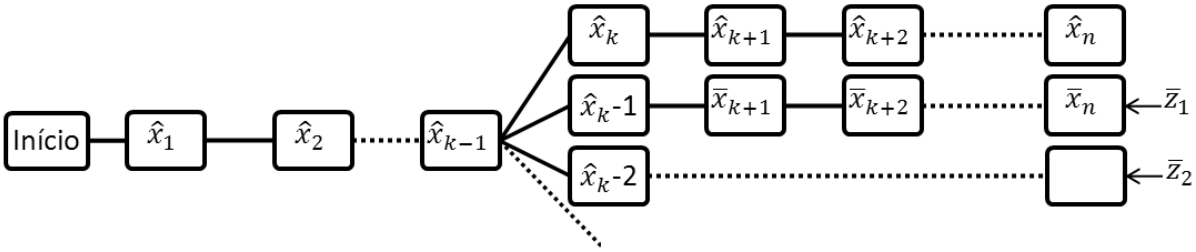
Ainda não construindo tais soluções, o custo deste processo pode ser muito alto, considerando um problema com muitas variáveis e também se a capacidade da mochila for grande.

Observação 1. *No pior caso, é necessário verificar no conjunto das partes dos itens qual conjunto possui a maior utilidade, logo devem ser desenvolvidos* $2^n - 1$ *ramos, deixando claro que qualquer tentativa de resolução exaustiva é inviável para grandes exemplares.*

Então, o objetivo é criar um algoritmo em que não seja necessário buscar na árvore toda, ou seja, alguns ramos são “podados” se não houver esperança de encontrar a solução em seu desenvolvimento.

Para isso ser possível, é necessário estimar o valor em que um ramo pode alcançar na função-objetivo. Considere na Figura 3 um ramo arbitrário $\hat{x}_1,\ldots,\hat{x}_n$, onde k é o maior índice entre 1 e $n-1$ maior que zero, $\hat{x}_k$ receberá o valor $\hat{x}_k - 1$ e um novo ramo seria desenvolvido com os valores $\hat{x}_1,\ldots,\hat{x}_{k-1},\hat{x}_k - 1,\overline{x}_{k+1},\ldots\overline{x}_n$. A solução corrente é $\hat{z}$ e a solução do novo ramo que seria construído será chamada de $\overline{z}_1$, tal como indica a Figura 3.

Figura 3: Novo ramo arbitrário



Fonte: Próprio autor.

Pretende-se estimar o valor de $\overline{z}_1$ sem desenvolver o ramo (sem calcular $\overline{x}_{k+1},\ldots\overline{x}_n$) para compará-lo à solução corrente $\hat{z}$ e verificar se o ramo pode ser podado e não desenvolvido. Isso acontecerá se for encontrado um limitante superior para $\overline{z}_1$, denotado por M_1, tal que $M_1 \leq \hat{z}$, e por transitividade isso implica $\overline{z}_1 \leq \hat{z}$, garantido que o ramo não precisa ser desenvolvido, pois não constrói uma solução melhor que a corrente.

Defina M_2 como um limitante superior para $\overline{z}_2$, que por sua vez é a solução para o ramo desenvolvido a partir de $x_k = \hat{x}_k - 2$ (Figura 3), e assim $M_3, M_4, \ldots, M_{x_k}$ são definidos de forma análoga. É interessante também que ocorra $M_1 \geq M_2 \geq \ldots \geq M_{x_k}$, pois assim, se $M_1 \leq \hat{z}$ pode-se podar não só o nó em que $x_k = \hat{x}_k - 1$, mas o nó em que $x_k = \hat{x}_k - 2$ e os outros também, podendo-se fazer a volta na árvore (reduzir o valor de k).

Pretendendo-se então limitar $\overline{z}_1$ sem calcular $\overline{x}_{k+1}, \ldots \overline{x}_n$, observe que:

$$\begin{aligned} \overline{z}_1 &= \sum_{i=1}^{k} u_i \overline{x}_i + \sum_{i=k+1}^{n} u_i \overline{x}_i \\ &= \underbrace{\sum_{i=1}^{k-1} u_i \hat{x}_i + u_k (\hat{x}_k - 1)}_{\text{valor conhecido}} + \underbrace{\sum_{i=k+1}^{n} u_i \overline{x}_i}_{\text{desconhecido}} . \end{aligned} \tag{1.38}$$

O valor que precisa ser estimado é $\sum_{i=k+1}^{n} u_i \overline{x}_i$. A única restrição aplicada à eventual construção do ramo, além das variáveis serem inteiras positivas, é a capacidade da mochila, $\sum_{i=1}^{n} l_i \overline{x}_i \leq L$, então é necessário usá-la para impor um limite à $\overline{z}_1$.

Observe que:

$$\begin{aligned} &\sum_{i=1}^{k} l_i \overline{x}_i + \sum_{i=k+1}^{n} l_i \overline{x}_i \leq L \\ &\Rightarrow \underbrace{\sum_{i=1}^{k-1} l_i \hat{x}_i + l_k (\hat{x}_k - 1)}_{\text{conhecido}} + \underbrace{\sum_{i=k+1}^{n} l_i \overline{x}_i}_{\text{desconhecido}} \leq L, \end{aligned}$$

então:

$$\begin{aligned} &\sum_{i=k+1}^{n} l_i \overline{x}_i \leq L - \sum_{i=1}^{k-1} l_i \hat{x}_i - l_k (\hat{x}_k - 1) \\ &\Rightarrow \sum_{i=k+1}^{n} \frac{l_i}{u_i} u_i \overline{x}_i \leq \underbrace{L - \sum_{i=1}^{k-1} l_i \hat{x}_i - l_k (\hat{x}_k - 1)}_{\text{valor conhecido}} . \end{aligned} \tag{1.39}$$

Agora, é necessário fazer uma análise. Para limitar o valor desconhecido da inequação (1.38), que é $\sum_{i=k+1}^{n} u_i \overline{x}_i$, é necessário “retirar” a fração $\frac{l_i}{u_i}$

na inequação (1.39). Uma maneira de se fazer isso é tomando o mínimo $\min_{i=k+1}^{n}\left\{\frac{l_i}{u_i}\right\}$, minorando assim os valores dessas frações e mantendo a desigualdade válida. Outra maneira é reordenar as variáveis, o que não modifica o problema em sua origem, e há duas maneiras diferentes de fazer a reordenação:

1) De forma que $\frac{l_1}{u_1} \geq \frac{l_2}{u_2} \geq \cdots \geq \frac{l_k}{u_k} \geq \frac{l_{k+1}}{u_{k+1}} \geq \cdots \geq \frac{l_n}{u_n}$.

2) De forma que $\frac{l_1}{u_1} \leq \frac{l_2}{u_2} \leq \cdots \leq \frac{l_k}{u_k} \leq \frac{l_{k+1}}{u_{k+1}} \leq \cdots \leq \frac{l_n}{u_n}$.

De imediato, a primeira maneira de reordenação não oferece vantagens, pois nessa ordem $\min_{i=k+1}^{n}\left\{\frac{l_i}{u_i}\right\} = \frac{l_n}{u_n}$ para todo $k = 1,\ldots,n$. Enquanto que na segunda maneira tem-se $\min_{i=k+1}^{n}\left\{\frac{l_i}{u_i}\right\} = \frac{l_{k+1}}{u_{k+1}}$.

De (1.39) e (1.38) segue que:

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

$$\Rightarrow \min_{i=k+1}^{n}\left\{\frac{l_i}{u_i}\right\}\sum_{i=k+1}^{n}u_i\bar{x}_i \leq L - \sum_{i=1}^{k-1}l_i\hat{x}_i - l_k(\hat{x}_k - 1)$$

$$\Rightarrow \min_{i=k+1}^{n}\left\{\frac{l_i}{u_i}\right\}\sum_{i=k+1}^{n}u_i\bar{x}_i \leq L - \sum_{i=1}^{k-1}l_i\hat{x}_i - l_k(\hat{x}_k - 1)$$

$$\Rightarrow \sum_{i=k+1}^{n}u_i\bar{x}_i \leq \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}\left(L - \sum_{i=1}^{k-1}l_i\hat{x}_i - l_k(\hat{x}_k - 1)\right)$$

$$\Rightarrow \begin{aligned}\bar{z}_1 &= \sum_{i=1}^{k-1}u_i\hat{x}_i + u_k(\hat{x}_k - 1) + \sum_{i=k+1}^{n}u_i\bar{x}_i \\ &\leq \sum_{i=1}^{k-1}u_i\hat{x}_i + u_k(\hat{x}_k - 1) + \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}\left(L - \sum_{i=1}^{k-1}l_i\hat{x}_i - l_k(\hat{x}_k - 1)\right).\end{aligned}$$

Defina $M_1 = \sum_{i=1}^{k-1}u_i\hat{x}_i + u_k(\hat{x}_k - 1) + \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}\left(L - \sum_{i=1}^{k-1}l_i\hat{x}_i - l_k(\hat{x}_k - 1)\right)$, que também pode ser definido como:

$$M_1 = \sum_{i=1}^{k}u_i\bar{x}_i + \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}\left(L - \sum_{i=1}^{k}l_i\bar{x}_i\right).$$

Analogamente, para $r = 2,\ldots, x_k$, defina:

$$M_r = \sum_{i=1}^{k-1} u_i \hat{x}_i + u_k(\hat{x}_k - r) + \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}\left(L - \sum_{i=1}^{k-1} l_i \hat{x}_i - l_k(\hat{x}_k - r)\right).$$

Segue que:

$$\begin{aligned}
\overline{z}_r &= \sum_{i=1}^{n} u_i \overline{\overline{x}}_i \\
&= \sum_{i=1}^{k-1} u_i \hat{x}_i + u_k(\hat{x}_k - r) + \sum_{i=k+1}^{n} u_i \overline{\overline{x}}_i \\
&= \sum_{i=1}^{k-1} u_i \hat{x}_i + u_k(\hat{x}_k - r) + \sum_{i=k+1}^{n} \frac{u_i}{l_i} l_i \overline{\overline{x}}_i \\
&\leq \sum_{i=1}^{k-1} u_i \hat{x}_i + u_k(\hat{x}_k - r) + \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}\left(\sum_{i=k+1}^{n} l_i \overline{\overline{x}}_i\right) \\
&\leq \sum_{i=1}^{k-1} u_i \hat{x}_i + u_k(\hat{x}_k - r) + \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}\left(L - \sum_{i=1}^{k} l_i \overline{\overline{x}}_i\right) \\
&= \sum_{i=1}^{k-1} u_i \hat{x}_i + u_k(\hat{x}_k - r) + \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}\left(L - \sum_{i=1}^{k-1} l_i \hat{x}_i - l_k(\hat{x}_k - r)\right) \\
&= M_r
\end{aligned}$$

Definidos os limitantes $M_1, M_2, \ldots, M_{x_k}$ de $\overline{z}_1, \overline{z}_2, \ldots, \overline{z}_{x_k}$, respectivamente, resta verificar a condição procurada de que $M_1 \geq M_2 \geq \ldots \geq M_{x_k}$. Para isso, seja r arbitrário entre 1 e x_{k-1}, observe que:

$$\begin{aligned}
M_r - M_{r+1} = &\sum_{i=1}^{k-1} u_i \hat{x}_i + u_k(\hat{x}_k - r) + \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}\left(L - \sum_{i=1}^{k-1} l_i \hat{x}_i - l_k(\hat{x}_k - r)\right) \\
&- \sum_{i=1}^{k-1} u_i \hat{x}_i + u_k(\hat{x}_k - r - 1)) + \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}\left(L - \sum_{i=1}^{k-1} l_i \hat{x}_i - l_k(\hat{x}_k - r - 1)\right) \\
= &\, u_k - \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\} l_k
\end{aligned}$$

Assim, tem-se que $M_r \geq M_{r+1}$ se, e somente se, $u_k - \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\} l_k \geq 0$, o que é equivalente a $\frac{u_k}{l_k} \geq \max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\}$, o que somente ocorrerá para todo $k = 1,\ldots,n$ se as variáveis estiverem ordenadas como em 2). Portanto, considere o problema com as variáveis ordenadas como em (1.40), conforme proposto por Dantzig (1957) para a resolução do problema relaxado:

$$\frac{u_1}{l_1} \geq \frac{u_2}{l_2} \geq \ldots \geq \frac{u_k}{l_k} \geq \ldots \geq \frac{u_n}{l_n}. \tag{1.40}$$

Com isso, $\min_{i=k+1}^{n}\left\{\frac{l_i}{u_i}\right\} = \frac{l_{k+1}}{u_{k+1}}$ e $\max_{i=k+1}^{n}\left\{\frac{u_i}{l_i}\right\} = \frac{u_{k+1}}{l_{k+1}}$.

A reordenação está em conforme com a noção intuitiva de que itens com utilidade maior em relação ao peso são preferíveis de serem escolhidos. Isso motiva a definição de *eficiência* do item i como $\frac{u_i}{l_i}$.

Visto que $\bar{z}_r \leq M_r$ para todo $r = 1,\ldots,x_k$ e $M_1 \geq M_2 \geq \ldots \geq M_{x_k}$, se ocorre $M_1 \leq \hat{z}$ então $\bar{z}_r \leq \hat{z}$ para todo $r = 1,\ldots,x_k$, e assim pode-se podar o ramo e fazer a volta (*backtrack*) na árvore. Sendo assim, defina:

$$M = M_1 = \sum_{i=1}^{k} u_i \bar{x}_i + \frac{u_{k+1}}{l_{k+1}}\left(L - \sum_{i=1}^{k} l_i \bar{x}_i\right), \tag{1.41}$$

como um limitante superior para a utilidade do ramo a ser construído, podendo ser utilizado como critério de poda. Na equação (1.41), $\bar{x}_i = \hat{x}_i$ para $i \in \{1,\ldots,k-1\}$ e $\bar{x}_k = \hat{x}_k - 1$.

Sintetizando, o Quadro 1 mostra os passos do algoritmo assim construído.

Quadro 1: Algoritmo *Backtracking*

Algoritmo *Backtracking*
Entrada: n, u_i, l_i, L.
Saída: x_i, z.
Inicialização: faça $z = 0$, $k = 0$;

Quadro 2: Algoritmo *Backtracking* (continuação)

Passo 1 (desenvolvimento de um ramo)**:**
Para $j = k+1,\ldots,n$ faça
$\hat{x}_j = \left\lfloor (L - \sum_{i=1}^{j-1} l_i \hat{x}_i) / l_j \right\rfloor$;
e faça $k = n$;
Passo 2 (salvar solução corrente)**:**
Se $\sum_{i=1}^{n} c_i \hat{x}_i > z$, então

faça $z = \sum_{i=1}^{n} c_i \hat{x}_i$;

e para $j = 1, \ldots, n$ faça

$x_i = \hat{x}_i$;

Passo 3 (*backtrack*)**:**

a) Se $k = 1$ então

pare!

caso contrário, faça

$k = k - 1$;

b) Se $\hat{x}_k = 0$ então

volte ao passo **a)**.

caso contrário, faça

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

Passo 4 (poda)**:**

Se $\sum_{i=1}^{k} u_i \hat{x}_i + \frac{u_{k+1}}{l_{k+1}} \left(L - \sum_{i=1}^{k} l_i \hat{x}_i \right) > z$ então

volte ao passo **1**.

caso contrário, volte ao passo **3**.

Fonte: Chvátal (1983, p. 206)

Aplicando o algoritmo *Backtracking* no exemplo (1.35)-(1.37), as variáveis são reordenadas de forma que o problema se resuma à:

Maximizar:

$$z = 4y_1 + 5y_2 + 5y_3 + 2y_4 \tag{1.42}$$

sujeito a:

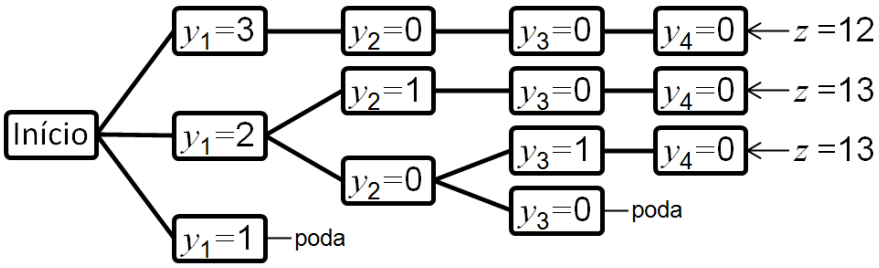
$$33y_1 + 49y_2 + 51y_3 + 22y_4 \leq 120 \tag{1.43}$$

$$y_1, y_2, y_3, y_4 \in \mathbb{Z}_+^*, \tag{1.44}$$

uma vez que $\frac{4}{33} \geq \frac{5}{49} \geq \frac{5}{51} \geq \frac{2}{22}$. Após efetuarem-se os cálculos, pode-se verificar que a árvore gerada pelo algoritmo é como na Figura 4.

A solução encontrada pelo algoritmo é $x_1 = y_3 = 0$, $x_2 = y_1 = 2$, $x_3 = y_4 = 0$, $x_4 = y_2 = 1$ e $z = 13$.

Figura 4: Árvore com podas



Fonte: Próprio autor.

Para resolver o Problema da Mochila Restrito com o *Backtracking*, basta modificar no Passo 1 o cálculo de $\hat{x}_j$ por $\hat{x}_j = \min\left\{\left\lfloor\left(L-\sum_{i=1}^{j-1} l_i\hat{x}_i\right)\Big/ l_j\right\rfloor, d_j\right\}$, e assim é garantido que a quantidade escolhida de cada item não ultrapassará sua demanda. Analogamente, para resolver o Problema da Mochila 0-1, o Passo 1 conta com o cálculo de $\hat{x}_j = \min\left\{\left\lfloor\left(L-\sum_{i=1}^{j-1} l_i\hat{x}_i\right)\Big/ l_j\right\rfloor, 1\right\}$, ou ainda $\hat{x}_j = 1$ se $L-\sum_{i=1}^{j-1} l_i\hat{x}_i \geq l_j$ e $\hat{x}_j = 0$ no caso contrário.

Observe que, ordenando os itens como em (1.40), o primeiro ramo da árvore construído coloca o máximo de cada item nessa ordem na mochila, assim como na resolução do problema relaxado, onde um último entra com a quantidade truncada. Sendo assim, a primeira solução viável encontrada pode ser vista como um arredondamento da resolução do problema relaxado.

1.3.2 Heurísticas para o *Bin-packing*

Algumas heurísticas utilizadas para resolver o *Bin-packing* podem passar, da forma mais simples possível, a ideia por trás de heurísticas que resolvem problemas de corte de estoque.

A heurística mais simples que aproxima a solução do *Bin-packing* é a *Next-Fit* (NF), no qual o primeiro item é colocado na primeira mochila, e a partir do segundo, insere-se o item corrente na mochila corrente se couber, caso não caiba toma-se a próxima mochila, onde o item é colocado. Para isso são necessários dois procedimentos básicos, um de incluir o item corrente k na mochila corrente q e outro de incluí-lo na próxima mochila, sendo p a quantidade de mochilas já utilizadas. Seja B_q o conjunto de índices dos itens que irão compor a mochila q, e

$c(B_q)$ a capacidade utilizada pela mochila q. Os dois procedimentos são descritos no Quadro 3.

Quadro 3: Procedimentos para heurísticas do Bin-packing

Procedimento Mochila Usada
Procedimento: MochilaUsada(B_q, $c(B_q)$, l_k).
$c(B_q) = c(B_q) + l_k$;
$B_q = B_q + \{k\}$;
fim-procedimento
Procedimento Mochila Nova
Procedimento: MochilaNova(p, l_k).
$p = p + 1$**;**
$c(B_p) = l_k$;
$B_p = \{k\}$;
fim-procedimento

Fonte: Hoto (2001, p. 27)

Outra heurística conhecida é a *First-Fit* (FF). Nela, cada item, seguindo a ordem de indexação, é colocado na primeira mochila que couber, e só se não puder ser colocado em nenhuma que entrará em uma nova mochila. Ou seja, diferente da *Next-Fit,* que verifica apenas se o item pode entrar na última mochila usada, a *First-Fit* verifica se ele pode entrar nas mochilas já usadas.

Dessa forma, a *First-Fit* coloca o item na primeira mochila que puder ser colocado, mas pode-se pensar em preencher melhor as mochilas, colocando o item na mochila que mais ficará cheia com a inclusão dele. É a ideia da heurística *Best-Fit*.

Nas notações anteriores, se $L - c(B_i) \geq l_k$ então o item k pode ser inserido na mochila i, e o espaço disponível na mochila passará a ser $L - c(B_i) - l_k \geq 0$ (antes de atualizar $c(B_i) = c(B_i) + l_k$). Uma vez que a *Best-Fit* escolhe a mochila que mais pode ficar cheia com a inclusão do item k, deve ser

escolhida a mochila r tal que $L - c(B_r) - l_k = \min_{1 \le i \le n}\{L - c(B_i) - l_k \ge 0\}$. O Quadro 4 mostra as heurísticas enunciadas.

Quadro 4: Heurísticas do Bin-packing

Heurística *Next-Fit*

Entrada: n, l_i, L.

Saída: B_i.

Inicialização: $p = 1$; $c(B_1) = l_1$; $B_1 = \{1\}$;

Para $k = 2, \ldots, n$ faça

Se $L - c(B_p) \ge l_k$ então

MochilaUsada (B_p, $c(B_p)$, l_k);

Caso contrário, faça MochilaNova(p, l_k);

Heurística *First-Fit*

Entrada: n, l_i, L.

Saída: B_i.

Inicialização: $p = 1$; $c(B_1) = l_1$; $B_1 = \{1\}$;

Para $k = 2, \ldots, n$ faça

$encontrou = 0$; $q = 1$;

Enquanto $q \le p$ e $encontrou = 0$ faça

Quadro 5: Heurísticas do Bin-packing (continuação)

Se $L - c(B_q) \ge l_k$ então

MochilaUsada(B_q, $c(B_q)$, l_k);

e $encontrou = 1$;

caso contrário, faça $q = q + 1$;

Se $encontrou = 0$ então

MochilaNova(p, l_k);

Heurística *Best-Fit*

Entrada: n, l_i, L.

Saída: B_i.

Inicialização: $p = 1$; $c(B_1) = l_1$; $B_1 = \{1\}$;

Para $k = 2, \ldots, n$ faça

$f_r = \min_{1 \le i \le p} \{ f_i = L - c(B_i) - l_k$ tal que $f_i \ge 0 \}$;

Se $f_r \ge 0$ então

MochilaUsada(B_r, $c(B_r)$, l_k);

caso contrário, faça MochilaNova(p, l_k);

Fonte: Próprio autor.

Uma ideia para melhorar essas heurísticas é considerar os itens em ordem decrescente de larguras, ou seja, $l_1 \ge l_2 \ge \cdots \ge l_n$. Considerando essa ordenação, as heurísticas NF, FF e BF recebem o nome de *Next-Fit Decreasing* (NFD), *First-Fit Decreasing* (FFD) e *Best-Fit Decreasing* (BFD), respectivamente.

O tempo computacional do algoritmo NF é $O(n)$, e dos outros cinco mencionados é $O(n \log n)$.

Johnson (1973) provou que, para qualquer exemplar I do *Bin-packing,* vale:

$$FFD(I) \le \frac{11}{9} I^* + 4,$$

onde $FFD(I)$ é a solução do exemplar I encontrada pela heurística FFD e I^* é a solução ótima do exemplar. Limitantes superiores para as outras heurísticas podem ser encontrados em Martello e Toth (1997, p. 223).

CAPÍTULO 2

APLICAÇÕES

Além da descrição clássica do problema da mochila, que traz uma hipotética aplicação em escolher itens para compor a mochila, Martello e Toth (1997) apresenta a seguinte aplicação para o Problema da Mochila 0-1: uma empresa dispõe de um capital de L dólares para fazer investimentos e possui n possíveis investimentos, sendo que o j-ésimo requer l_j dólares, e u_j é o lucro esperado com a aplicação. Pretende-se maximizar a soma dos lucros de cada investimento feito, investindo-se no máximo os L dólares disponíveis.

Antes mesmo de se estudar problemas de mochila, Lorie e Savage (1955) apresentou um estudo do problema de investimento descrito no parágrafo acima. Por ser um trabalho voltado à área de finanças, os autores chamam a atenção para a dificuldade de estimular o valor u_j, neles podem ser consideradas questões como o interesse da empresa no investimento e comparações com investimentos semelhantes efetuados com recorrência pela empresa. Em sua resolução, Lorie e Savage (1955) considerou os valores $u_j - pl_j$, como uma taxa do lucro líquido esperado com a aplicação, sendo p inicialmente um valor representativo tal como uma taxa de retorno mínima da empresa.

O valor de p é modificado de acordo com a necessidade do problema, por exemplo, se não há itens suficientes para completar a mochila (para utilizar o recurso total) com essas novas "utilidades" positivas, então p é diminuído. Encontrado o valor de p que não pode ser aumentado (se for o recurso necessário para os investimentos com utilidade positiva não é totalmente aproveitado) nem diminuído (assim os itens com utilidade positiva ultrapassam a capacidade de investimento), pode-se selecionar os investimentos que devem ser escolhidos. Essa heurística é equivalente ao arredondamento pela resolução do problema da mochila relaxado, uma vez que escolhe itens tal que $u_i - p \cdot l_i > 0 \Leftrightarrow {u_i}/{l_i} > p$, embora seja mais intuitiva numa abordagem econômica, pois o valor p pode simbolizar uma atratividade dos investimentos. Lorie e Savage (1955) considerou também o

esperado de cada investimento em diversos intervalos de tempo, o que modifica o valor de utilidade e requer uma adaptação na técnica de solução.

Vasquez e Hao (2001) modelou um problema de decidir quais fotos um satélite deve tirar dentre as fotografias candidatas. É um problema que surge quando um satélite que tira fotos diariamente (de 30 a 100 fotos aproximadamente) deve tirar as mais vantajosas considerando aspectos como sua trajetória e condições de fotografar. A formulação matemática proposta é um problema de mochila 0-1 generalizado, contando com restrições adicionais.

As possíveis fotografias (em alguns exemplares em torno de 2300) são os itens a compor a mochila, cuja capacidade é a memória disponível para armazenar as imagens. Os pesos das imagens são os tamanhos de arquivos, e as utilidades são calculadas a partir de critérios como importância dos clientes, urgência e previsões meteorológicas. As restrições adicionadas são lineares e consideram as três câmeras disponíveis, que fotos estéreos precisam de duas câmeras ao mesmo tempo, dentre outros fatores. Para resolver o problema gerado pelo modelo, Vasquez e Hao (2001) criou um algoritmo que procura sempre melhorar a solução, utilizando vários mecanismos acrescentados pelos autores, como de manuseios das restrições, intensificação (direcionar a busca onde parece promissor) e diversificação (buscar em soluções muito diferentes das já percorridas).

Florentino e Spadotto (2006) aplicaram o Problema da Mochila Restrito ao carregamento do palhiço da cana-de-açúcar.

A cana-de-açúcar sofreu uma grande mudança em sua produção nos últimos tempos. A colheita, que era realizada após uma queimada na plantação, passou a ser feita de forma mecanizada, produzindo biomassa residual (palhiço), que deve ser retirada do local da plantação, pois cria condições favoráveis para o aparecimento de parasitas e um atraso no broto da cana.

A legislação brasileira tem cada vez forçado mais agricultores a aderirem à colheita mecanizada. A queima é proibida em algumas cidades e possivelmente em torno de 2017 será totalmente proibida em alguns estados. A colheita mecanizada possui vantagens em questões ambientais, uma vez que a queima do canavial polui a atmosfera e pode causar incêndios descontrolados que queimam reservas e danificam instalações, ainda o palhiço pode ser usado como biomassa na geração de energia. A cana-de-açúcar tornou-se a maior fonte de

biomassa do país. Sendo assim, algumas usinas tem transportado o palhiço, em diversos tamanhos, diretamente do campo para o centro de processamento.

Um dos principais problemas neste processo é o custo do transporte do palhiço. Para ser retirado do campo, primeiro o palhiço é enleirado por máquinas enfardadoras que formam fardos cilíndricos ou prismáticos (em formato de prisma retangular). Depois os fardos são colocados em caminhões para transporte. A proposta de Florentino e Spadotto (2006) foi aplicar o Problema da Mochila Restrito para maximizar o uso dos caminhões no carregamento de fardos prismáticos, carregando o máximo de biomassa possível em cada viagem.

Considerando n máquinas enfardadoras, cuja i-ésima produz d_i fardos de volume $u_i = l_i$, e que a caçamba do caminhão possua volume L, maximizar o volume de palhiço ocupado no caminhão sujeito a respeitar a capacidade do caminhão e a produção de cada enfardadora é o mesmo que resolver o Problema da Mochila Restrito proposto neste trabalho com $u_i = l_i$.

Chajakis e Guignard (2003) aplicou problemas de mochila à distribuição de produtos em veículos com múltiplos compartimentos. Nesses veículos (geralmente navios ou caminhões) pretende-se transportar produtos em diferentes temperaturas e diferentes composições, e por isso devem ficar em compartimentos separados. Com relação às composições, podem ser admitidos produtos de origem do petróleo, comida, itens de mercearia para lojas de conveniência, etc. Após definir quais veículos efetuarão quais entregas, aplica-se um problema de caixeiro viajante para determinar as rotas. Os autores criaram uma heurística que, com a relaxação lagrangeana, decompõe o problema em problemas de mochila, procura uma solução factível em cada iteração e aperfeiçoa-a caso encontre-a.

Outra aplicação bastante considerada na literatura foi feita por Straszak *et al* (1993), no sistema de aprovação de votos restrito. A votação por aprovação difere-se do tradicional sistema, em que cada eleitor vota em um candidato (ou projeto) e os mais votados são eleitos, sendo que nesse sistema cada eleitor pode votar em quantos candidatos desejar, como se aprovasse alguns e desaprovasse os outros. Entre as vantagens de uma eleição com essa normatização enunciadas por Straszak *et al* (1993), há maior flexibilidade para as opiniões e aumenta-se a participação dos eleitores.

Acrescentam-se restrições ao modelo de votação por aprovação, tornando-o o chamado modelo de votação por aprovação restrito (possível tradução para *constrained approval voting*). As restrições em questão consideram que diferentes interesses devem ser representados no comitê eleito. Dessa forma, são criadas categorias de candidatos, sendo que cada candidato pertence a diversas categorias, e uma restrição matemática garante que o número de candidatos por categoria esteja na faixa determinada a ela (proporcional ao número de votos recebidos por candidatos desta categoria). Com estas condições, Straszak *et al* (1993) desenvolveu um modelo matemático que consiste num problema de mochila com restrições adicionais.

Para resolver esse modelo, os autores propõem um algoritmo *branch-and-bound* aproximativo, que tem como sub-rotina a resolução de problemas de mochila 0-1, e conta com algumas relaxações. Uma das preocupações para a formulação do algoritmo foi o tempo computacional. Em um exemplo da eleição de 50 membros para o *Committee for Organization and Management Sciences of the Polish Academy of Sciences* foram necessários 5,27 segundos de execução, considerado um tempo satisfatório.

NELIβEN (1995) aplicou um problema de mochila para encontrar um limitante superior para a área utilizável de um pallet, em um estudo de logística que visa encontrar o melhor arranjo de caixas retangulares em um pallet retangular, cuja importância econômica está em reduzir os custos de distribuição e transporte. Alocação de caixas em pallets, de itens em contêineres, de contêineres em navios, e problemas do tipo são comuns de conterem aplicações do *Bin-packing* (HOTO, 2001, p. 30).

Xavier e Miyazawa (2008) aplicou uma generalização do *Bin-packing* (*Class Constrained Bin Packing*) a sistemas de vídeo sob demanda (*video on demand*), que também possuem uma aplicação ao próprio *Bin-packing*, caso tenha-se por objetivo construir um *server* com o mínimo de discos tal que todos os pedidos são satisfeitos, conforme enuncia Xavier e Miyazawa (2008, p. 4).

Para melhor esclarecimento, um sistema de vídeo sob demanda armazena arquivos de vídeo em servidores para que estes possam ser assistidos *online*, sem a necessidade de efetuar-se o *download* do vídeo. Há alguns anos esse sistema basicamente se resumia ao YouTube, porém é um sistema cada vez mais popular e explorado por outras empresas. No Brasil, destaca-se a Netflix, que cobra

mensalidades dos clientes para que esses possam ter acesso ilimitado a séries e filmes *online*. Um limite para o crescimento desse mercado no Brasil tem sido a velocidade da internet, à qual muitos usuários reclamam de ser uma das mais caras e lentas do mundo. Mesmo assim este mercado tem crescido, oferecendo a cada vez acervos maiores de filmes, maratonas, e inclusive seriados exclusivos.

O problema estudado por Xavier e Miyazawa (2008) consiste em construir um *server* baseado nos pedidos esperados por filmes, que podem ser estipulados pela popularidade de cada um, utilizando discos iguais (de mesma capacidade de armazenamento e tempo de leitura). Vale mencionar que o autor faz uso de uma heurística FFD adaptada.

Van de Vel e Shijie (1991) aplicou o *Bin-packing* à resolução de um problema de programação de produção. No problema, n trabalhos devem ser designados a m máquinas, sendo que o tempo de processamento do trabalho i é l_i. Pretende-se designar os trabalhos às máquinas de forma a minimizar o tempo total para realizá-los, com as máquinas trabalhando juntas. Os autores propuseram uma resolução do problema de designação de trabalho por meio de um problema *Bin-packing* associado.

Ainda na programação de produção, Morihara *et al* (1983) enunciou dois possíveis problemas. Neles, n máquinas diferentes $M_1,\ldots,M_n$ são usadas para produzir n tipos de bens $I_1,\ldots,I_n$, respectivamente, sendo que a máquina M_i produz um bem I_i em um dia utilizando a_i trabalhadores. Ainda considerou que o número de máquinas disponíveis do tipo M_i é m_i, condição conhecida como *restrição de máquinas*.

O primeiro problema proposto por Morihara *et al* (1983) assume que a quantidade de trabalhadores por dia deve ser no máximo P, e busca minimizar a quantidade de dias necessários para produzir uma determinada demanda de bens. O segundo problema assume que a quantidade de dias para produzir os bens demandados é máximo T, e busca minimizar a quantidade de trabalhadores necessários por dia.

Sem a restrição de máquinas, o primeiro problema é o *Bin-packing*, onde as mochilas são os dias, os bens são os itens, o peso l_i vale a_i, e a capacidade da mochila é P. O segundo problema, sem a restrição de máquinas, é

um problema de programação de produção clássico. Porém, com a adição das restrições de máquinas, o problema se torna diferente, ainda que heurísticas similares às desses problemas clássicos sejam propostas por Morihara *et al* (1983).

A propósito, algumas variações do *Bin-packing* necessárias em certas aplicações são enumeradas por Malkevitch (2015), tais como: a) O número de itens por mochila deve ser limitado por certo valor. Essa restrição pode aparecer em problemas de carregamento de caminhão; b) Alguns itens não devem ser colocados na mesma mochila. Em certos carregamentos é necessário garantir que se uma carga tiver problemas para chegar ao seu destino, outra com produtos idênticos deve suprir a necessidade dos itens no local; c) Há uma ordenação que limita a forma como os itens são inseridos na mochila. Por exemplo, ao considerar uma coletânea de músicas que devem ser gravadas em cd's, algumas músicas "combinam" mais com umas do que com outras.

Como mencionado no Capítulo 1, seção 1.1.5, o *Bin-packing* também é um caso particular do problema de corte de estoque. Suponha que, ao invés de colocar itens em pacotes, utilizando-se do menor número de pacotes, barras de mesma largura devam ser cortadas em itens menores, satisfazendo a obtenção de todos os itens, com o objetivo de utilizar-se do menor número de barras possíveis. No corte de uma barra, o fato da soma das larguras dos itens terem de ser no máximo a largura total da barra torna-se a restrição de mochila, e sendo assim, essa formulação do *Bin-packing* é equivalente à exibida na seção 1.1.5.

Espera-se com essas aplicações ter-se ilustrado a diversidade de formas e situações em que os problemas de mochila podem aparecer na prática. Além das aqui evidenciadas, existe uma série de aplicações desses problemas que podem ser encontradas na literatura, tais como em criptografia, controle orçamentário, orçamento de capital, fluxo de rede, simulação de recozimento, seleção de jornais para uma livraria e compartilhamento de memória (SALKIN; DE KLUYVER, 1975; LAGOUDAKIS, 1996). A maior parte das aplicações desses problemas está em problemas mais complexos, tal como Problemas de Corte e Empacotamento.

2.1 Problemas de Corte e Empacotamento

Esta seção concentra-se nos Problemas de Corte e Empacotamento, que tanto constituem uma aplicação dos problemas de mochila

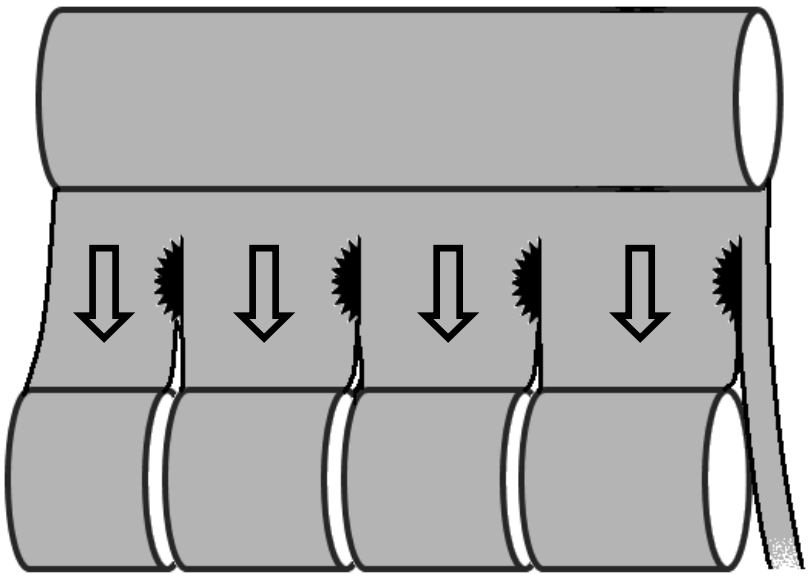
quanto são importantes na formulação do Problema da Mochila Compartimentada feita no Capítulo 3.

A classe de problemas conhecida como *Problemas de Corte e Empacotamento* envolve problemas de cortar unidades maiores em unidades menores ou empacotar unidades menores em unidades maiores. Problemas desses dois tipos possuem formulação matemática equivalente. Se, por exemplo, o objetivo é minimizar a quantidade de unidades maiores para cortar as menores, o problema tem a mesma formulação matemática de minimizar a quantidade de unidades maiores as quais serão empacotadas as unidades menores.

Os Problemas de Corte e Empacotamento podem ser classificados quanto à dimensão, à seleção e ao sortimento dos itens. No caso de problemas de corte de estoque são selecionados os itens maiores e demandados todos os itens menores, sendo que o estoque pode ser homogêneo (itens grandes de mesma largura, como abordado no *Bin-packing*) ou heterogêneo, e os itens pequenos são variados com muitos de cada largura (no *Bin-packing* há apenas um item de cada largura).

Quanto à dimensão, um problema pode ser unidimensional, como nos problemas de mochila apresentados. Um problema de corte de estoque unidimensional consiste em cortar barras à qual apenas uma dimensão é considerada, por exemplo no corte de tubos ou canos, e também de rolos ou bobinas de certo material (tecido, aço, alumínio, papel, entre outros), que são cortados em toda sua extensão, caracterizando um corte unidimensional, tal como ilustra a Figura 5.

Figura 5: Corte de uma bobina



Fonte: Rosa Neto (2015).

Em muitos processos de corte de bobinas, como em bobinas de aço (HOTO, 2001), a bobina grande é desenrolada e passa por facas que a cortam em várias tiras, enquanto tais tiras são novamente enroladas formando as bobinas menores, como mostra a Figura 5, representando uma bobina sendo cortada em quatro bobinas de largura menor e uma pequena sobra/perda de material.

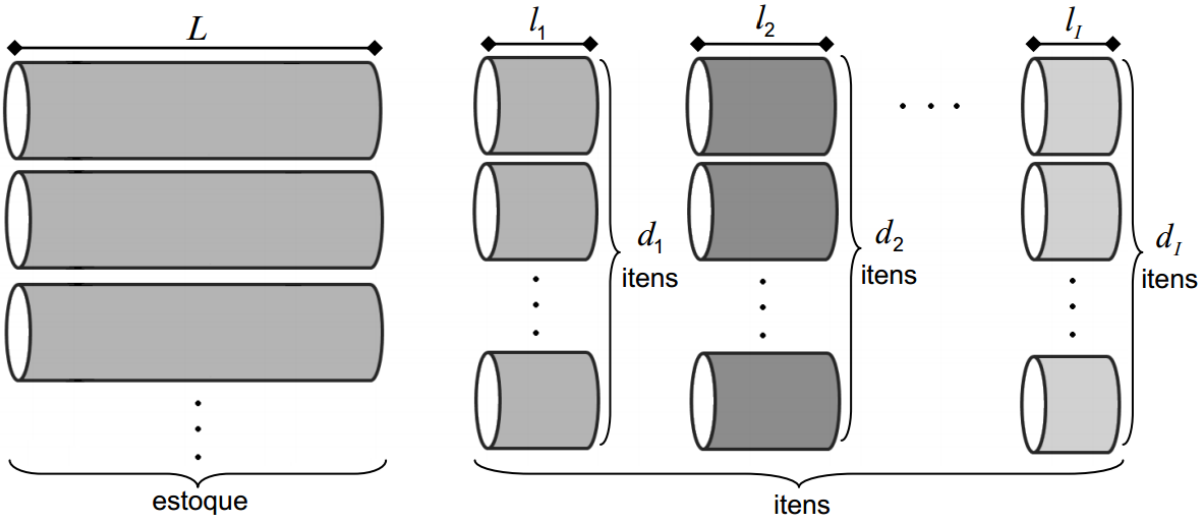
Essa sobra/perda constitui motivo de estudo de muitos trabalhos, que visam não só utilizar o mínimo de barras em estoque, como minimizar as “perdas” no processo, planejando os cortes de forma a concentrar “sobras” em um número menor de barras e gerando “retalhos” que poderão ser utilizados em futuras demandas de itens. Como neste trabalho não será abordado o Problema de Corte de Estoque com Aproveitamento de Sobras (Cherri *et al*, 2009), não haverá preocupação com essa terminologia de sobras, resíduos, retalhos e perdas. A sobra/perda representa neste trabalho apenas o espaço da mochila não ocupado pelos itens que foram escolhidos para compô-la. Para estudos do problema de reaproveitamento de sobras, além de Cherri *et al* (2009), recomenda-se Rosa Neto (2015), que descreveu o problema e as principais heurísticas estudadas por outros autores, incluindo uma reformulada pelo próprio autor, apresenta um gerador de problemas, faz simulações computacionais e comparações dessas heurísticas.

Um problema de corte de estoque pode também ser bidimensional, onde uma placa é cortada em itens menores, geralmente retangulares, por exemplo,

um compensado de madeira que será usado para produzir móveis. Enquanto isso, o problema de carregamento em um contêiner é tridimensional. Existem também classificações intermediárias com relação à dimensão de um problema, como o problema de corte de estoque 1.5-dimensional, onde são cortados itens retangulares, porém de um rolo em que não se considera limitação no comprimento, apenas de largura, como um rolo de tecido. Neste caso, procura-se minimizar o comprimento desenrolado para obter os itens demandados.

Nesta seção será mais enfatizado o *Problema de Corte de Estoque Unidimensional (PCE)* com estoque homogêneo. Neste problema, são consideradas barras de estoque de largura L e I tipos de itens, sendo que o item de índice i possui largura l_i e demanda d_i. A Figura 6 ilustra as barras de estoque e os itens, onde, por hipótese, existem barras suficientes para atender à demanda de itens, ou considera-se ilimitada a quantidade de barras de largura L. O objetivo é obter todos os itens cortando o mínimo possível de barras de estoque.

Figura 6: Estoque e itens.



Fonte: próprio autor.

Uma possível formulação para o problema é a de Kantorovich (1997 apud VALÉRIO DE CARVALHO, 2002, p. 255) exibida na seção 1.1.5. No entanto, essa formulação conta com muitas restrições de mochila, o que dificulta o arredondamento inteiro de uma solução. Será apresentada uma formulação mais explorada na literatura, principalmente por permitir a técnica de geração de colunas proposta por Gilmore e Gomory (1961), que gera padrões de corte (resolvendo um

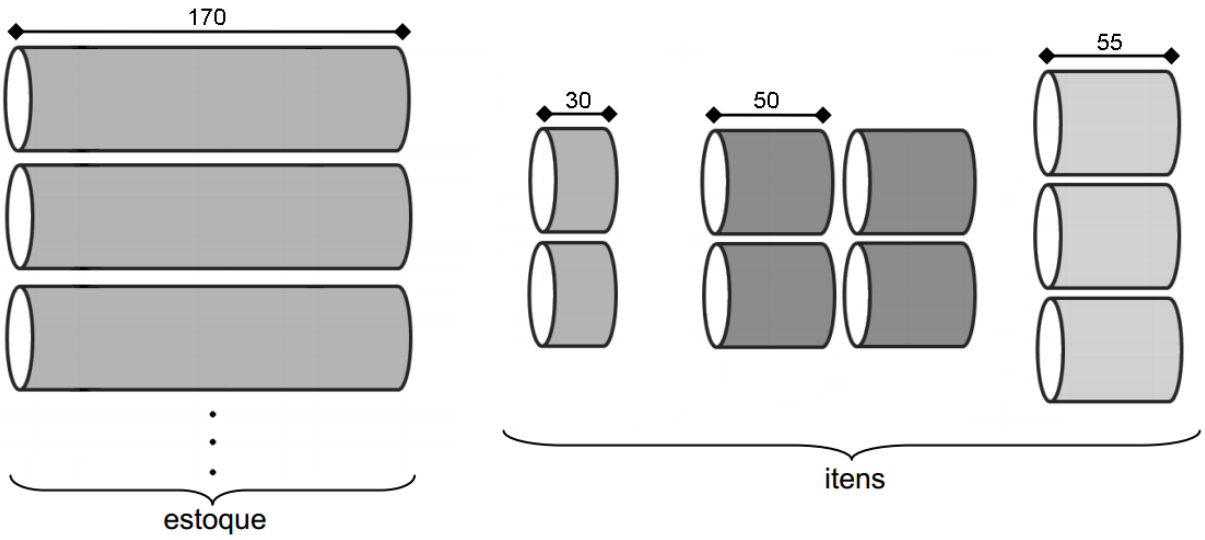
problema de mochila) com valores inteiros, o que facilita a resolução do problema mestre. Vance (1998) aplicou a decomposição de Dantzig-Wolf para provar que o modelo de Kantorovich é equivalente ao que será enunciado a seguir.

Defina um *padrão de corte* como uma I-upla $a_p = \left(a_{1p}, a_{2p}, \ldots, a_{Ip}\right) \in \mathbb{Z}_+^I$ tal que $l_1 a_{1p} + l_2 a_{2p} + \ldots + l_I a_{Ip} \leq L$. Esse nome é conveniente porque os valores $a_{1p}, a_{2p}, \ldots, a_{Ip}$ podem ser entendidos como as quantidades de itens de largura $l_1, l_2, \ldots, l_I$, respectivamente, a serem cortados de uma barra de estoque. Assim o PCE pode ser formulado em duas etapas:

1) Defina todos os possíveis padrões de corte. Suponha que existam P padrões.
2) Uma vez obtidos os padrões de corte, deve-se decidir a quantidade de vezes que cada padrão é utilizado. Como cada padrão de corte corta uma barra de estoque, o problema é minimizar a quantidade de padrões utilizados. Defina por x_p a quantidade de vezes que o padrão p deve ser utilizado.

Rosa Neto (2015) apresenta o seguinte exemplar: considere $L = 170$, $I = 3$, $l_1 = 30$, $l_2 = 50$, $l_3 = 55$, $d_1 = 2$, $d_2 = 4$, e $d_3 = 3$, como ilustra a Figura 7. Os possíveis padrões de corte para esse exemplo são descritos na Tabela 1.

Figura 7: Estoque e itens do exemplo.



Fonte: próprio autor.

Tabela 1: Padrões de corte do exemplo.

p	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a_{1p}	5	4	4	3	3	3	2	2	2	2	2	2	1	1
a_{2p}	0	1	0	1	0	0	2	1	1	0	0	0	2	1
a_{3p}	0	0	0	0	1	0	0	1	0	2	1	0	0	1
p	15	16	17	18	19	20	21	22	23	24	25	26	27	
a_{1p}	1	1	1	1	0	0	0	0	0	0	0	0	0	
a_{2p}	1	0	0	0	3	2	2	1	1	1	0	0	0	
a_{3p}	0	2	1	0	0	1	0	2	1	0	3	2	1	

Fonte: próprio autor.

Embora nem todos os padrões de corte da Tabela 1 precisam ser construídos se a demanda for considerada (por exemplo: (5,0,0), e apenas dois itens de índice 1 são requeridos), 27 é a quantidade de padrões que podem ser gerados com os dados a respeito das larguras das barras. Vale mencionar que os padrões (5,0,0), (0,3,0) e (0,0,3) são denominados *padrões de corte homogêneos maximais*, por serem formados por apenas um item inserido na mochila (a ser cortado de uma barra) na quantidade máxima possível.

Para que apenas duas barras de largura 30 sejam produzidas, a soma das quantidades em que cada item de largura 30 é cortado nos padrões utilizados deve ser igual a dois, ou seja, $\sum_{p=1}^{P} a_{1p} x_p = 2$. O mesmo vale para as demandas das barras de largura 50 e de 55, logo o problema da etapa 2) é:

Minimizar:

$$\sum_{p=1}^{27} x_p$$

sujeito a:

$$\sum_{p=1}^{P} a_{1p} x_p = 2$$

$$\sum_{p=1}^{P} a_{2p} x_p = 4$$

$$\sum_{p=1}^{P} a_{3p} x_p = 3$$

$x_p \in \mathbb{Z}_+$ para $p = 1, \ldots, P$.

A solução encontrada por um algoritmo *branch-and-bound* associado ao Método Simplex (inserindo-se restrições e ramificando novas possibilidades para o problema até encontrar solução inteira) é 3 para a função-objetivo (3 barras devem ser cortadas) com $x_9 = x_{19} = x_{25} = 1$ e as outras variáveis de decisão anuladas, ou seja, em uma barra cortam-se dois itens de 30 e um de 50, em outra cortam-se três itens de 50 e na outra três itens de 55. Outra solução equivalente é $x_7 = x_{22} = x_{23} = 1$, e as outras variáveis com valor nulo.

Em forma matricial, as restrições deste exemplo podem ser expressas por:

$$A_{3\times 27} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{27} \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 3 \end{pmatrix},$$

onde as colunas de A são os padrões de corte da Tabela 1.

Generalizando, o Problema de Corte e Estoque Unidimensional com estoque homogêneo e a satisfazer as demandas com igualdade possui a seguinte formulação:

Minimizar:

$$\sum_{p=1}^{P} x_p$$

sujeito a:

$$A_{I\times P} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_P \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_I \end{pmatrix}$$

$x_p \in \mathbb{Z}_+$ para $p = 1, \ldots, P$.

As colunas da matriz A são todos os possíveis padrões de corte do problema. A questão combinatória sobre o número de padrões de corte é a mesma do número de ramos na árvore de possibilidades do Problema da Mochila Restrito, inclusive na Tabela 1 os padrões estão ordenados seguindo a regra de ramificação descrita na seção 1.3.1. Logo, esse número pode crescer exponencialmente em relação ao tamanho do problema. A estratégia de geração de colunas descrita na

seção 2.1.1 tem por objetivo contornar essa situação, resolvendo o problema sem gerar todos os padrões de corte, apenas os mais “atrativos”.

Ainda sem gerar todos os padrões de corte possíveis, a solução do PCE pode não ser tão simples como no exemplo dado, que por ser pequeno pôde ser resolvido com um método *branch-and-bound* associado ao Método Simplex. A seção 2.1.2 descreve alguns algoritmos de aproximação para o problema, para serem utilizados no caso de exemplares grandes.

2.1.1 Geração de Colunas

Observe que, no exemplo dado pela Figura 7, apenas 3 variáveis geraram 27 possíveis padrões de corte (Tabela 1). Agora, imagine em exemplares de tamanho grande. De fato, se for considerado $d_i = 1$ para todo $i \in N$, escolher os padrões factíveis equivale a encontrar no conjunto das partes dos itens os subconjuntos de itens que não somam em largura mais do que a barra de estoque, logo, no pior caso, tem-se $2^n - 1$ possibilidades de padrões de corte. Isso significa que, para grandes exemplares, a quantidade de colunas da matriz A das restrições pode ser demasiadamente grande, dificultando a resolução do problema. Por isso, gerar todas as colunas da matriz A (padrões de corte) previamente pode ser inviável, o que torna interessante um método que gera colunas apenas quando necessário durante a resolução do problema.

No Método Simplex, as colunas da matriz A apenas são consultadas no passo em que decide-se qual coluna deve entrar na base. Como o problema é de minimização, deve-se escolher uma coluna $a_p = (a_{1p}, a_{2p}, \ldots, a_{Ip})^{\mathrm{T}}$ tal que $c_B B^{-1} a_p - c_p > 0$ (ver Apêndice A para mais detalhes), onde B é a base corrente, e c representa os coeficientes da função-objetivo. No PCE tem-se $c = (1,1,\ldots,1)$. Defina $y = (y_1, \ldots, y_I) = c_B B^{-1}$. Assim, deve-se escolher uma coluna de A (um padrão de corte) tal que $\sum_{i=1}^{I} y_i a_{ip} > 1$.

Geralmente, o critério utilizado no Método Simplex é escolher a coluna com maior valor de custo reduzido (para problema de minimização), ou seja,

maior valor $c_B B^{-1} a_p - c_p = \sum_{i=1}^{I} y_i a_{ip}$ tal que a_p é um padrão de corte, o que equivale a resolver o seguinte Problema da Mochila Irrestrito:

Maximizar:

$$y_1 a_{1p} + y_2 a_{2p} + \ldots + y_I a_{Ip} \tag{2.1}$$

sujeito a:

$$l_1 a_{1p} + l_2 a_{2p} + \ldots + l_I a_{Ip} \leq L \tag{2.2}$$

$$a_{ip} \in \mathbb{Z}_+ \text{ para } i = 1, \ldots, I\,. \tag{2.3}$$

Se o valor encontrado na função-objetivo for maior que 1, encontrou-se uma coluna atrativa, caso contrário, não há custo reduzido estritamente positivo e há garantia de solução ótima para o PCE relaxado.

A técnica de geração de colunas consiste, portanto, em a cada iteração do Método Simplex, resolver um problema de otimização, para construir a coluna que entrará na base, evitando a avaliação de todos os padrões de corte que compõem a matriz A. Para dar início ao método, pode ser considerada como base a matriz formada pelos padrões de corte homogêneos maximais. Um exemplo dessa rotina é dado no Apêndice A.

Segundo, Gilmore e Gomory (1961), não é necessário encontrar o máximo da função-objetivo para o problema de mochila, apenas uma coluna que satisfaça $\sum_{i=1}^{I} y_i a_{ip} > 1$ e $\sum_{i=1}^{I} l_i a_{ip} \leq L$, sendo assim, pode-se resolver o problema de mochila relaxado conforme faz Dantzig (1957), arredondar a solução e verificar se a coluna encontrada é atrativa, e apenas quando não se encontra colunas atrativas com esse método que resolve-se o problema de mochila com exatidão. Assim reduz-se o esforço computacional do processo.

Estudos computacionais apontam a uma conjectura de que o Problema de Corte de Estoque Unidimensional pode ser bem resolvido por relaxação e solução via geração de colunas seguida por um arredondamento da solução, se os itens tiverem demanda alta e não muitas larguras diferentes. Porém, se o problema tiver demanda baixa e muitos itens, o arredondamento pode não apresentar uma boa solução, sendo necessária a utilização de heurísticas para resolvê-lo (CHVÁTAL, 1983, p. 198; POLDI; ARENALES, 2006, p. 474). É o caso do

Bin-packing, que conta com um largo espectro de larguras de itens, ou seja, muitos itens diferentes e demanda baixa ($d_i = 1$).

2.1.2 Heurísticas para o PCE

Caso o problema seja grande o suficiente para não ser viável a solução por um método *brach-and-bound* que calcula a solução exata, heurísticas que fornecem uma solução aproximada podem dar uma resposta satisfatória. As heurísticas mais comuns se dividem entre construtivas e residuais. As heurísticas construtivas consistem em escolher “bons” padrões de corte e usá-los o possível, enquanto que as heurísticas residuais constituem em resolver o problema relaxado por meio da geração de colunas, para encontrar uma solução aproximada que pode não cumprir exatamente a demanda, atualizar a demanda e resolver um problema menor (residual). Como mencionado na seção 2.1.1, problemas com poucos itens e demanda alta costumam ter boa aproximação, podendo ser resolvidos por heurísticas residuais, enquanto que a problemas com demanda baixa e muitos itens é mais recomendada a tentativa por heurísticas construtivas.

Com relação às heurísticas construtivas, são famosas a FFD e a Gulosa. A heurística *FFD* para o PCE é como descrita para o *Bin-packing*, consistindo em cortar-se os itens em ordem de largura decrescente da primeira barra que for possível. Há duas maneiras de implementar essa heurística. Em uma delas, aloca-se o maior item num padrão de corte quantas vezes for possível (até que não haja mais espaço na barra para este item ou que sua demanda seja atendida), em seguida, aloca-se o segundo maior item nessa barra quantas vezes for possível, e assim por diante, formando um padrão de corte. Faz-se o mesmo barra por barra até que toda a demanda seja atendida. Na outra maneira, aloca-se o maior item na primeira barra o quanto possível e, se sua demanda não é atendida, parte-se para a segunda barra e continua-se inserindo o maior item em seu padrão de corte, e assim em quantas barras forem necessárias para atender sua demanda. Então, faz-se o mesmo para o segundo maior item, procurando alocá-lo no espaço restante das primeiras barras já utilizadas. Esta segunda maneira trabalha exatamente como foi descrita a FFD do *Bin-packing*.

Já a heurística *Gulosa* procura melhorar a primeira maneira de se implementar a heurística FFD, em que um padrão de corte é gerado em cada

iteração. Ao invés de gerar tal padrão de corte, colocando os itens em ordem decrescente como na heurística FFD, resolve-se um Problema de Mochila Restrito para encontrar um padrão de corte que preencha o máximo possível da capacidade da barra de estoque, sendo que neste problema os pesos e utilidades são iguais (como no problema de Soma de Subconjuntos), e a demanda é a demanda residual do item, pois a cada iteração, o padrão de corte gerado deve descontar na demanda os itens produzidos. Para a implementação da heurística Gulosa costuma-se utilizar o padrão de corte gerado quantas vezes for possível (enquanto não estiver produzindo itens em excesso) e só depois gerar um novo padrão de corte.

A solução por heurísticas residuais, como mencionado, resolve o problema com as variáveis relaxadas e encontra uma solução aproximada (é difícil ocorrer de a solução do problema relaxado ser inteira, mas caso ocorra, o algoritmo deve parar e há garantia de solução ótima para o PCE). Uma maneira de aproximar a solução é truncando (arredondando para baixo) as variáveis. Lembrando que cada variável representa um padrão de corte, então ao truncá-las menos barras serão cortadas com relação à solução exata do problema relaxado. Sendo assim, faltarão itens a serem produzidos, e então a demanda é atualizada e um novo problema com demanda menor, chamado *problema residual*, deve ser resolvido. Este novo problema pode ser resolvido da mesma maneira, gerando um novo resíduo, e sucessivamente, até que a demanda residual se anule ou a solução aproximada seja nula.

No caso da demanda residual se anular o problema foi resolvido e o algoritmo para. Mas caso a solução aproximada seja nula, o último problema residual obtido deve ser resolvido de outra forma, entretanto, como a solução relaxada para esse problema não possui variáveis maiores do que 1, resta uma baixa demanda a ser atendida. A forma como este último problema residual será resolvido designa diferentes heurísticas residuais.

Sendo assim, a *Heurística Residual FFD* consiste em resolver o problema residual final com a heurística construtiva FFD. Analogamente, a *Heurística Residual Gulosa* consiste em resolver o problema residual final com a heurística construtiva Gulosa.

Outra alternativa é tentar encontrar uma solução exata para o problema residual final com um algoritmo que combina geração de colunas (por causa das muitas variáveis, o que não foi necessário no exemplo ilustrado na Figura

7) e *branch-and-bound*, inserindo-se restrições e criando ramificações para o problema até encontrar-se uma solução inteira.

Pode-se modificar também, em heurísticas residuais, a forma como a solução do PCE relaxado é arredondada. Uma possível maneira é, de algum modo ordenados os padrões de corte escolhidos na solução do problema relaxado (decorrentes de variáveis básicas da base ótima), uma a uma cada variável é arredondada para cima e, caso viole a demanda (produza itens em excesso), seu valor é diminuído em uma unidade até que não haja violação. Assim, constrói-se uma aproximação que não ultrapassa a demanda, mas também pode não atendê-la, deixando um problema residual a ser resolvido, que pode ser resolvido da mesma maneira até que a demanda seja cumprida, não sendo necessária, portanto, a solução de um último problema residual por meio de uma heurística construtiva. Os modos como se ordena tais variáveis determinam diferentes heurísticas, mais detalhes podem ser vistos em Poldi e Arenales (2006).

CAPÍTULO 3

O PROBLEMA DA MOCHILA COMPARTIMENTADA

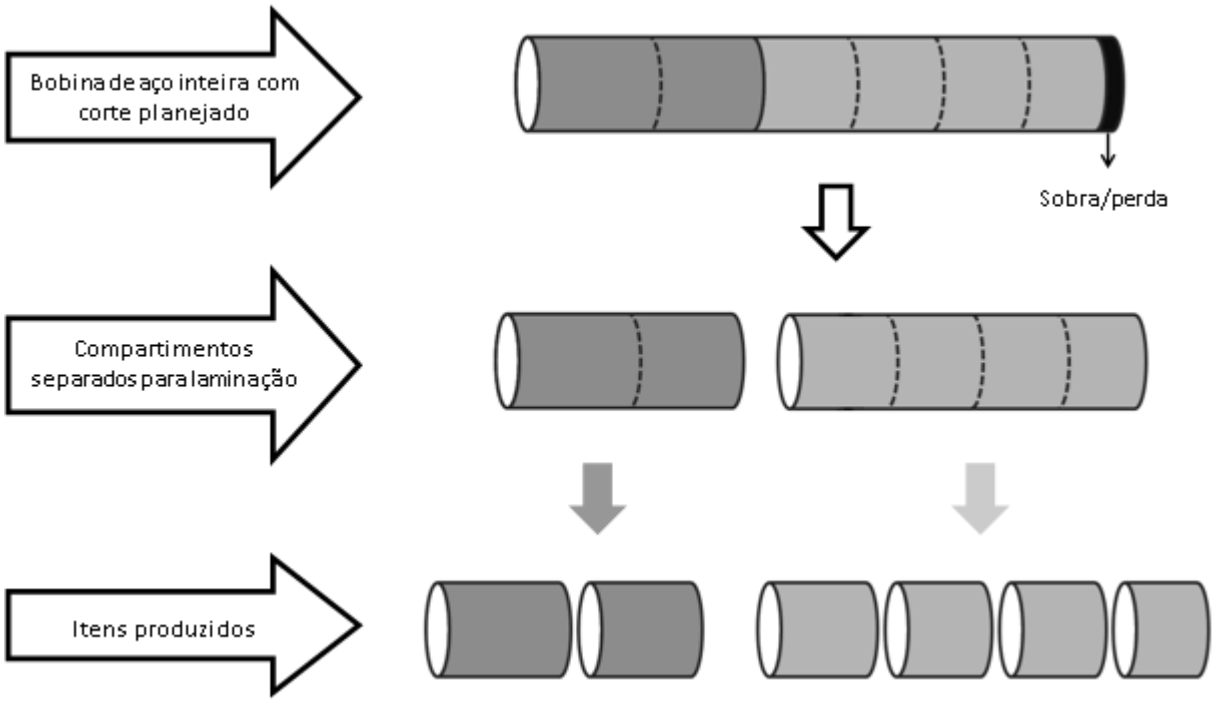
No capítulo anterior, foi vista a aplicabilidade dos problemas de mochila em Problemas de Corte e Empacotamento, em especial, no Problema de Corte de Estoque Unidimensional. A seção 2.1.1 apresentou uma alternativa para a resolução do PCE relaxado em que não é necessário computar todos os padrões de corte, mas resolver a cada iteração um problema de mochila a qual a restrição de mochila nada mais é do que a condição das variáveis comporem um padrão de corte. Logo, este subproblema, que visa gerar um padrão de corte, depende da forma como o corte deve ser executado. No caso exibido foi considerada apenas a restrição física da barra ter espaço para gerar os itens, mas outras condições podem surgir na prática. Num caso específico tem-se o Problema da Mochila Compartimentada.

No Problema da Mochila Compartimentada os itens deverão ser inseridos na mochila organizados em classes disjuntas. A compartimentação de uma mochila ocorre quando os itens devem ser agrupados no seu interior segundo afinidades. Por exemplo, quando não se deseja misturar remédios, ferramentas e comidas no interior de uma mochila, requerendo a construção de compartimentos distintos para que neles sejam alocados itens de mesma natureza. Esta particularidade induz no conjunto de itens uma partição matemática (classes de itens com mesma afinidade).

A motivação desta versão de mochila surgiu da necessidade de se gerar padrões de corte em duas fases. Hoto (2001) apresentou uma aplicação no qual bobinas de aço são cortadas numa primeira fase, e laminadas (para reduzir a espessura do aço, de acordo com a finalidade dos itens, como para confecção de bicicletas, indústria automobilística e construção civil) numa segunda fase, sendo que as bobinas cortadas sofrerão diferentes laminações, o que torna necessário criar no esquema de corte diferentes grupos de bobinas que serão laminadas juntas.

A Figura 8 ilustra o processo de corte e laminação. A primeira fase de corte produz as sub-bobinas a serem laminadas e a segunda fase produz fitas (itens).

Figura 8: Corte em duas fases.



Fonte: Hoto (2001, p. 33).

Seguindo essa ideia, suponha que o conjunto de índice dos itens, denotado por N, esteja particionado em q classes N_k $(k = 1,\ldots,q)$. Dessa forma $N = N_1 \cup \ldots \cup N_q$ e $N_r \cap N_s = \varnothing$ se $r \neq s$. No problema de corte de bobinas de aço pode-se entender uma classe como sendo o conjunto de índices dos itens que irão sofrer a mesma laminação entre a primeira e a segunda fase de corte.

Ainda no problema de corte de bobinas de aço, itens devem ser laminados juntos na segunda etapa, respeitando uma largura mínima e uma máxima aceita pela máquina para fazer a laminação. Para isso, deve-se construir no interior da mochila compartimentos de capacidades limitadas entre um valor mínimo $L^k_{\min}$ e um máximo $L^k_{\max}$, onde os índices dos itens de um compartimento devem pertencer a uma mesma classe N_k.

Observação 2. *A ideia original para formalizar o modelo matemático consiste em:* ***imaginar todos os compartimentos (sub-bobinas) viáveis construídos, para, em seguida, selecionar os que deverão fazer parte da compartimentação da mochila.*** *Esta metodologia é idêntica à usada na geração de colunas (GILMORE, GOMORY, 1961).*

Com este enfoque, denote por V_k o conjunto de índices de todos os compartimentos viáveis com itens indexados em N_k, sendo $V = V_1 \cup \ldots \cup V_q$ o conjunto de índices de todos os compartimentos, onde $V_r \cap V_s = \varnothing$ se $r \neq s$, ou seja, $V_1, \ldots, V_q$ é uma partição de V.

Seja L a capacidade da mochila (largura de cada bobina), u_i a utilidade (multiplicadores simplex vindos do problema mestre) e l_i o peso do item de índice i (largura da fita). Considere como variáveis de decisão: a_{ij} que controlará a quantidade de itens de índice i no compartimento de índice j, y_j que definirá a quantidade de compartimentos de índice j utilizados, $\delta_j = 1$ se o compartimento de índice j foi construído e, $\delta_j = 0$ no caso contrário. A formulação matemática apresentada por Hoto (2001) consiste em:

Maximizar:

$$\sum_{j \in V_1} \left(\sum_{i \in N_1} u_i a_{ij} \right) y_j + \cdots + \sum_{j \in Vq} \left(\sum_{i \in Nq} u_i a_{ij} \right) y_j \tag{3.1}$$

sujeito a:

$$\sum_{j \in V_1} \left(\sum_{i \in N_1} l_i a_{ij} \right) y_j + \cdots + \sum_{j \in Vq} \left(\sum_{i \in Nq} l_i a_{ij} \right) y_j \leq L \tag{3.2}$$

$$\delta_j L^k_{\min} \leq \sum_{i \in N_k} l_i a_{ij} \leq \delta_j L^k_{\max}, \; j \in V_k, \; k = 1, \ldots, q \tag{3.3}$$

$$\delta_j \in \{0,1\} \text{ e } a_{ij}, y_j \geq 0 \text{ e inteiros, } i \in N \text{ e } j \in V. \tag{3.4}$$

A restrição (3.2) assegura que a soma dos espaços ocupados por cada compartimento não ultrapassa a capacidade da mochila. As restrições em (3.3) garantem que cada compartimento tenha capacidade limitada entre o mínimo e o máximo aceito para compartimentos da classe N_k. Em (3.4) é determinado o domínio das variáveis.

Exemplo 1. *Considere apenas 2 classes e 5 itens, sendo 3 itens na primeira classe e 2 itens na segunda classe. Neste caso,* $q = 2$, $N = \{1,2,3,4,5\}$, $N_1 = \{1,2,3\}$ *e* $N_2 = \{4,5\}$. *Suponha que os compartimentos da primeira classe devem ter de 8 a 10 uc (unidades de comprimento) e os compartimentos da segunda classe devem ter de*

7 a 12 uc, então $L^1_{\min} = 8\,uc$, $L^2_{\min} = 7\,uc$, $L^1_{\max} = 10\,uc$, $L^2_{\max} = 12\,uc$. *Considere também as larguras dos itens, como* $l_1 = 2$ *uc,* $l_2 = 3\,uc$, $l_3 = 5\,uc$, $l_4 = 5$ *uc e* $l_5 = 6\,uc$. *Sendo assim, um compartimento de índice* j *representado por* $a_j = (a_{1j}, a_{2j}, a_{3j})$ *pertence a* V_1 *se* $8 \le 2a_{1j} + 3a_{2j} + 5a_{3j} \le 10$ *, onde* a_{ij} *é um inteiro positivo que representa a quantidade de itens de índice* i *no compartimento de índice* j. *Analogamente, um compartimento de índice* j *representado por* $a_j = (a_{4j}, a_{5j})$ *pertence à* V_2 *se* $7 \le 5a_{4j} + 6a_{5j} \le 12$.

Assim, os compartimentos possíveis de serem construídos são: $a_1 = (5,0,0)$, $a_2 = (4,0,0)$, $a_3 = (3,1,0)$, $a_4 = (2,2,0)$, $a_5 = (2,0,1)$, $a_6 = (1,2,0)$, $a_7 = (1,1,1)$, $a_8 = (0,3,0)$, $a_9 = (0,1,1)$, $a_{10} = (0,0,2)$, $a_{11} = (2,0)$, $a_{12} = (1,1)$ *e* $a_{13} = (0,2)$. *Logo, pode-se definir* $V_1 = \{1,2,3,4,5,6,7,8,9,10\}$ *e* $V_2 = \{11,12,13\}$. *(A subseção 5.2.1 exibe uma maneira de se enumerar todos os compartimentos viáveis).*

Note que, embora os itens de índice 3 e 4 possuam a mesma largura, são itens diferentes, uma vez que pertencem a classes diferentes, podendo sofrer diferentes processos de laminação ou qualquer processo entre a primeira e a segunda fase de corte.

Numa primeira observação, se o conjunto V de todos os compartimentos fosse previamente determinado, então os valores $L_j = \sum_{i \in N_k} l_i a_{ij}$ e $U_j = \sum_{i \in N_k} u_i a_{ij}$ que representam a largura e a utilidade do compartimento $j \in V_k$, respectivamente, estariam bem determinados, e a equação (3.3) seria satisfeita. Logo, o problema (3.1)-(3.4) se resume a maximizar $\sum_{j \in V_1} U_j y_j + \cdots + \sum_{j \in Vq} U_j y_j$ sujeito a $\sum_{j \in V_1} L_j y_j + \cdots + \sum_{j \in Vq} L_j y_j \le L$ e y_j inteiro positivo para todo $j \in V$, ou seja, um Problema da Mochila Irrestrito. Todavia, gerar todo o conjunto V é computacionalmente impossível para certos exemplares, uma vez que em cada classe o conjunto de compartimentos possíveis possui tantos elementos quanto são as possibilidades de solução factível para um problema de mochila, algo já discutido nos capítulos anteriores, especialmente na seção 1.3.1. Com esta observação, conforme discutido

na seção 1.2, pode-se dizer que o PMC “não é mais difícil do que” o Problema da Mochila Irrestrito. Portanto, o PMC é NP-difícil.

Em outra observação importante a respeito do conjunto V, nota-se que ele é “flexível”, no sentido de que não precisa estar bem definido com uma quantidade determinada de elementos diferentes entre si. Esta é a Observação 3, que será utilizada no Capítulo 5, e por este motivo é aqui colocada em destaque:

Observação 3. *Não há uma única maneira de definir o conjunto* $V = V_1 \cup \ldots \cup V_q$. *A única exigência é que contenha os índices de todos os compartimentos possíveis de serem construídos, mas a ordem na qual a indexação é feita pode ser diferente sem alterar a solução ótima do problema. Além disso, o conjunto* V *pode conter mais índices do que o necessário, relacionados a compartimentos idênticos (compostos pelas mesmas quantidades dos mesmos itens).*

Por exemplo, no Exemplo 1, o conjunto V_2 *poderia ser definido como* $V_2 = \{11,12,13,14\}$, *com* $a_{11} = (1,1)$, $a_{12} = (2,0)$, $a_{13} = (0,2)$ *e* $a_{14} = (2,0)$.

Observe que tanto a função-objetivo quanto a restrição de mochila, no modelo de Hoto (2001), são não-lineares, o que faz com que o Problema da Mochila Compartimentada seja tratado como um problema não-linear. Cruz (2010) desenvolveu simulações baseadas num modelo linear para o Problema da Mochila Compartimentada Restrito, que consiste no problema como descrito acima, adicionando algumas restrições que costumam aparecer em problemas práticos.

Por exemplo, no corte de estoque em duas fases, há uma demanda a ser atendida para cada item. Considere d_i a demanda do item de índice i. Assim, para garantir que não sejam produzidos itens em excesso, deve-se impor a restrição (3.5):

$$\sum_{k=1}^{q} \sum_{j \in V_k} a_{ij} y_j \leq d_i,\ i \in N = N_1 \cup \ldots \cup N_q, \tag{3.5}$$

onde a soma das quantidades de um determinado item em cada compartimento que o produz é limitada pela demanda desse item.

Outras restrições podem estar relacionadas à maneira com que os itens são alocados na mochila (ou cortados, no caso de um problema de corte).

Considere, por exemplo, no problema de corte de bobinas de aço, conforme ilustra a Figura 8, que a quantidade de facas que cortam o aço seja limitada, tanto no primeiro como no segundo processo de corte. O primeiro processo de corte dará origem a compartimentos (sub-bobinas a serem laminadas), o que implica que a quantidade de compartimentos deve ser limitada por uma constante F_1. Enquanto que, no segundo processo de corte, cada compartimento (sub-bobina laminada) será separado nos diversos itens que o compõe, portanto o número de itens em cada compartimento deve ser limitado por uma constante F_2. Neste caso devem valer as restrições:

$$\sum_{k=1}^{q} \sum_{j \in V_k} y_j \leq F_1 \tag{3.6}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2, \text{ para } j \in V = V_1 \cup \ldots \cup V_q,\ k = 1, \ldots, q \tag{3.7}$$

Com a adição das restrições (3.5)-(3.7) restrições ao modelo (3.1)-(3.4) obtém-se o Problema da Mochila Compartimentada Restrito:

Maximizar:

$$\sum_{j \in V_1} \left(\sum_{i \in N_1} u_i a_{ij} \right) y_j + \cdots + \sum_{j \in Vq} \left(\sum_{i \in Nq} u_i a_{ij} \right) y_j \tag{3.8}$$

sujeito a:

$$\sum_{j \in V_1} \left(\sum_{i \in N_1} l_i a_{ij} \right) y_j + \cdots + \sum_{j \in Vq} \left(\sum_{i \in Nq} l_i a_{ij} \right) y_j \leq L \tag{3.9}$$

$$\delta_j L^k_{\min} \leq \sum_{i \in N_k} l_i a_{ij} \leq \delta_j L^k_{\max},\ j \in V_k,\ k = 1, \ldots, q \tag{3.10}$$

$$\sum_{k=1}^{q} \sum_{j \in V_k} a_{ij} y_j \leq d_i,\ i \in N = N_1 \cup \ldots \cup N_q \tag{3.11}$$

$$\sum_{k=1}^{q} \sum_{j \in V_k} y_j \leq F_1 \tag{3.12}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2, \text{ para } j \in V = V_1 \cup \ldots \cup V_q,\ k = 1, \ldots, q \tag{3.13}$$

$$\delta_j \in \{0,1\} \text{ e } a_{ij}, y_j \geq 0 \text{ e inteiros, } i \in N \text{ e } j \in V. \tag{3.14}$$

O modelo (3.8)-(3.14) é mais fiel às condições do Problema de Corte das Bobinas de Aço Sujeitas à Laminação a Frio, sendo, portanto, empregado na geração dos padrões de corte. Trata-se da compartimentação de uma mochila ((3.1)-(3.4)) com restrições adicionais, portanto, ainda uma **Mochila Não-linear**.

3.1 ALGUMAS HEURÍSTICAS

O modelo (3.8)-(3.14) e a Observação 2 sugerem uma maneira de relaxar o problema: considerar apenas uma parcela dos compartimentos viáveis. Com isto, surgem heurísticas de decomposição, às quais as mais conhecidas são a Heurística da Decomposição, a Heurística dos “z” Melhores Compartimentos e a Heurística das “w” Capacidades.

No modelo (3.8)-(3.14), para que as variáveis do tipo a_{ij} sejam factíveis, independente das variáveis do tipo δ_j e y_j, devem satisfazer, para cada $j \in V_k,\ k = 1,\ldots,q$, as restrições:

$$\sum_{i \in N_k} l_i a_{ij} \leq L \tag{3.15}$$

$$L^k_{\min} \leq \sum_{i \in N_k} l_i a_{ij} \leq L^k_{\max} \quad \text{ou } a_{ij} = 0 \text{ para todo } i \in N \tag{3.16}$$

$$a_{ij} \leq d_i,\ i \in N \tag{3.17}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2 \tag{3.18}$$

$$a_{ij} \geq 0 \text{ e inteiro, } i \in N. \tag{3.19}$$

Como $L \geq L^k_{\max}$ para todo $k = 1,\ldots,q$, a restrição (3.15) é redundante ao assumir (3.16)-(3.19). Ademais, se $a_{ij} = 0$ para todo $i \in N$, nada será somado à função-objetivo, e por isso compartimentos assim não precisam ser gerados. Assim considere:

$$L^k_{\min} \leq \sum_{i \in N_k} l_i a_{ij} \leq L^k_{\max} \tag{3.20}$$

e as restrições (3.17)-(3.20) definem todos os compartimentos viáveis. Com todos os valores de a_{ij} possíveis, definidos, pode-se definir:

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

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

e o problema (3.8)-(3.14) pode ser escrito como:

Maximizar:

$$\sum_{j \in V_1} U_j y_j + \cdots + \sum_{j \in Vq} U_j y_j \tag{3.21}$$

sujeito a:

$$\sum_{j \in V_1} L_j y_j + \cdots + \sum_{j \in Vq} L_j y_j \leq L \tag{3.22}$$

$$\sum_{k=1}^{q} \sum_{j \in V_k} a_{ij} y_j \leq d_i,\ i \in N = N_1 \cup \ldots \cup N_q \tag{3.23}$$

$$\sum_{k=1}^{q} \sum_{j \in V_k} y_j \leq F_1 \tag{3.24}$$

$$y_j \geq 0 \text{ e inteiro, } j \in V, \tag{3.25}$$

que será denominado Problema Mestre. Note que as restrições (3.10) e (3.13) são satisfeitas em (3.17)-(3.20), por isso não possuem análogas em (3.21)-(3.25).

As heurísticas de decomposição visam gerar, ao invés de todos os compartimentos viáveis (que satisfazem (3.17)-(3.20)), alguns compartimentos e então fazer a compartimentação da mochila (resolver (3.21)-(3.25)). No Capítulo 5 é mencionado um algoritmo que gera todos os valores a_{ij} que satisfazem (3.17)-(3.20), utilizando um método de ramificação semelhante ao descrito na subseção 1.3.1, a fim de obter a solução ótima de alguns exemplares.

Visando selecionar compartimentos dentre todos os viáveis, o seguinte Problema de Mochila Restrito é usado para gerar o “melhor” compartimento $j \in V_k$ de capacidade máxima $L_{\text{cap}} \leq L_{\max}^k$:

Maximizar:

$$\sum_{i \in N_k} u_i a_{ij} \tag{3.26}$$

sujeito a:

$$L_{\min}^k \leq \sum_{i \in N_k} u_i a_{ij} \leq L_{\text{cap}} \tag{3.27}$$

$$a_{ij} \leq d_i,\ i \in N \tag{3.28}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2 \tag{3.29}$$

$$a_{ij} \geq 0 \text{ e inteiro, } i \in N. \quad (3.30)$$

A Heurística da Decomposição consiste em, para cada classe de itens, gerar o "melhor" compartimento, utilizando-se do Problema de Mochila Restrito (3.26)-(3.30), e em seguida resolver o problema (3.21)-(3.25), que define os compartimentos a serem inseridos na mochila. A Heurística dos "z" Melhores Compartimentos segue a mesma ideia, porém com "z" compartimentos gerados em cada classe e não somente um. O Quadro 6 descreve a Heurística dos "z" Melhores Compartimentos, e caso "z"=1, tem-se a Heurística da Decomposição.

Quadro 6: Heurística dos "z" Melhores Compartimentos

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 $k = 1, \ldots, q$ faça

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

Calcule as z melhores soluções do problema (3.26)-(3.30);

Para $contador = 1, \ldots, z$ faça

$j = comp + contador - 1$;

$j \in V_k$;

Grave a_{ij}, U_j, e L_j de acordo com a $contador$-ésima melhor solução de (3.26)-(3.30);

$comp = comp + z$

Resolva (3.21)-(3.25);

Fonte: Leão *et al* (2011)

A Heurística do Melhor Compartimento visa eliminar a resolução do problema (3.21)-(3.25), para isso são escolhidos sucessivamente compartimentos a compor a mochila e atualizada a demanda. Mais precisamente, define-se o "melhor" compartimento de cada classe, como na Heurística da Decomposição, e então escolhe-se o "melhor" entre todos de acordo com sua eficiência (U_j / L_j), e o insere na mochila, então a demanda é atualizada e o processo é repetido inserindo-se mais

um compartimento na mochila (o “melhor” dos “melhores” de cada classe), até que o padrão de corte esteja formado.

A Heurística das “w” Capacidades calcula, para cada classe, o melhor compartimento para “w” diferentes capacidades, ou seja, varia-se a largura máxima L_{cap} para obter compartimentos de larguras diferentes, obtendo-se assim $q\times$”w” compartimentos que irão compor a compartimentação da mochila. Observe, conforme o algoritmo descrito no Quadro 7, que se “w”=1, tem-se a Heurística da Decomposição.

Quadro 7: Heurística das “w” Capacidades

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 $k = 1,\ldots,q$ faça

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

Para $j = comp,\ldots,comp + w - 1$ faça

Resolva (3.26)-(3.30);

$j \in V_k$;

Grave a_{ij}, U_j, e L_j;

$L_{\text{cap}} = L_j - 1$;

$comp = comp + w$

Resolva (3.21)-(3.25);

Fonte: Leão *et al* (2011)

São encontradas na literatura heurísticas de retro-alimentação, que geram compartimentos conforme uma das heurísticas descritas acima, e na compartimentação da mochila contam com variáveis adicionais que geram um novo compartimento (HOTO *et al*, 2006; CRUZ, 2010). Também encontram-se heurísticas que exploram as ideias de mais de uma dessas heurísticas mencionadas.

3.2 UMA REVISÃO TEÓRICA

Como referido, o Problema da Mochila Compartimentada surge em problemas de corte em que os itens são cortados em duas fases. Pode-se dizer que os problemas de corte em várias etapas são estudados desde a década de setenta por Haessler.

3.2.1 Problemas de Corte de Estoque em Duas ou Mais Etapas

Haessler (1979) ressaltou que vários estudos com problemas de corte unidimensionais e bidimensionais foram realizados, porém não havia discussão do problema de corte em duas fases na literatura, ainda que problemas do tipo aparecessem comumente na prática, e nas mesmas ocasiões dos problemas de corte unidimensionais e bidimensionais comumente discutidos, tais como a indústria do papel e filmes plásticos.

No mesmo trabalho (HAESSLER, 1979), foi apresentada a aplicação do estudo de problemas de corte em duas fases na indústria de filmes plásticos. Neste exemplo, os itens (bobinas de filmes finais) são produzidos em diferentes diâmetros, e para isso a bobina mestre deve ser cortada em bobinas intermediárias. Haessler (1979) propõe, para resolver a modelagem deste problema, uma heurística que gera padrões de corte, seguindo a ideia de Gilmore e Gomory (1961), porém com algumas adaptações heurísticas.

Zak (2000) apresentou um estudo de problemas de corte em que a necessidade de múltiplos estágios está na limitação das facas. A modelagem apresentada para problemas com dois estágios de corte utiliza uma matriz formada por blocos, um definido por padrões de corte do primeiro estágio, outro definido por padrões de corte do segundo estágio, e um terceiro bloco definido por vetores que conectam os outros dois. Para resolver este problema, foi proposto um algoritmo baseado na geração de colunas, porém que gera tanto colunas quanto linhas, sendo as linhas os compartimentos gerados (bobinas intermediárias), e as colunas os padrões de corte para a bobina mestre. Para isso, dois problemas de mochila e um *bin-packing* com restrições adicionais foram utilizados como subproblemas. O *bin-packing*, por ser mais complicado, foi abordado com algumas hipóteses que o torna um problema de mochila com uma restrição adicional, a qual pode ser resolvido com

pequenas modificações no algoritmo *Backtracking* apresentado na subseção 1.3.1 (Capítulo 1).

Como subproblema de um problema de corte de estoque em duas etapas, é comum surgir o conhecido *Nasted Knapsack Problem*, que pode ser traduzido como Problema da Mochila Encapsulada, provavelmente o problema mais semelhante ao Problema da Mochila Compartimentada encontrado na literatura. Nele, as “cápsulas” possuem capacidade fixa, diferenciando o problema do PMC, em que as larguras dos compartimentos são variáveis.

Johnston e Khan (1995) abordou o *Nasted Knapsack Problem* e citou o famoso exemplo do ladrão usado para descrever o Problema da Mochila 0-1, similar ao do viajante descrito nesta dissertação, porém o ladrão quer separar em sua mochila os itens de acordo com determinadas categorias, por exemplo: comida, vestuário e outras. O problema não se limita a dois níveis de mochilas, em alguns casos constrói-se sub-mochilas dentro das sub-mochilas e sucessivamente. O problema foi aplicado na produção de cabos de fibra ótica, que precisam ser coloridos, a efeito de identificação, fazendo necessárias duas etapas de corte, a primeira corta os cabos que irão passar por diferentes processos de coloração, então os cabos coloridos são agrupados, e a segunda produz os itens demandados.

Correia *et al* (2004) apresentou um estudo em uma indústria de papel portuguesa em que buscou-se minimizar as perdas no corte de bobinas de papel. Os itens demandados são folhas e bobinas de papel. A necessidade de corte em duas fases se deu pela limitação no número de facas, e cortes perpendiculares são efetuados posteriormente para que parte das bobinas produzam folhas. O método de solução apresentado consiste em, primeiramente, gerar padrões de forma semelhante à utilizada em Ferreira *et al* (1990), eliminando os que não satisfazem condições como a largura máxima e mínima para inserir os rolos nas cortadoras, o número de facas nas cortadoras e a necessidade de rolos com a largura das folhas demandadas para produzi-las posteriormente. Depois esses padrões criados são usados como colunas no Método Simplex, e por fim faz-se uma heurística de arredondamento.

3.2.2 O Problema de Corte das Bobinas de Aço

Além de problemas de corte em várias etapas e do *Nasted Knapsack Problem*, estudos do Problema de Corte das Bobinas de Aço (PCBA) foram fundamentais para a formulação do PMC. Ferreira *et al* (1990) descreveu que vários processamentos podem ser feitos entre a primeira fase de corte e a segunda, tais como temperamento, tensionamento, laminação e endurecimento.

Ferreira *et al* (1990) propôs uma heurística para o Problema de Corte das Bobinas de Aço (algoritmo de Ferreira-Neves-Castro), a qual contraria a usada na prática, sendo que na indústria é comum planejar-se a segunda fase de corte (criar-se compartimentos) e depois verificar se os compartimentos podem ser incluídos num plano de corte da primeira fase, então o plano criado é usado (uma ou mais vezes), atualiza-se a demanda e cria-se, analogamente, um novo plano de corte, até que a demanda seja satisfeita. Contrariamente, o algoritmo de Ferreira-Neves-Castro consiste em, primeiramente, ordenar os itens de acordo com um fator de prioridade (sua largura multiplicada pela demanda), selecionar alguns itens utilizando-se dessa ordenação, fazer uso de uma heurística pra criar padrões de corte com esses itens, testar se os padrões criados são factíveis, e caso sejam usar o melhor padrão de corte factível, e repete-se o procedimento até que a demanda seja satisfeita.

No algoritmo de Ferreira-Neves-Castro os padrões de corte gerados inicialmente podem ser infactíveis, por causar superprodução quando combinados ou por não respeitar a capacidade das bobinas intermediárias. Se isso acontecer, o número de itens considerados na geração dos padrões e o número de compartimentos em um padrão são aumentados, e procura-se gerar novos padrões. Assim, pode ocorrer de muitos padrões terem de ser analisados, e possivelmente nenhum satisfaça as condições de compartimentação.

Valério de Carvalho e Rodrigues (1994, apud Leão *et al,* 2011, p. 456) propuseram a resolução do PCBA com a estratégia de geração de colunas. No entanto, para simplificar a geração de padrões compartimentados, consideraram apenas os padrões formados por compartimentos homogêneos, ou seja, cada compartimento sendo composto por apenas um tipo de item. Novamente, a dificuldade encontrada no PCBA foi gerar padrões compartimentados, o que motiva o estudo do Problema da Mochila Compartimentada.

3.2.3 O Problema da Mochila Compartimentada

Em Hoto (1996, apud HOTO, 2001, p. 119) são encontrados os primeiros elementos matemáticos que permitiram a definição do PMC, como um problema unidimensional sujeito a restrições de agrupamento. Além disso, em tal dissertação foi descrito um algoritmo heurístico para o PCBA mais elaborado que o de algoritmo de Ferreira-Neves-Castro, uma vez que procura gerar padrões compartimentados viáveis.

Marques (2000) apresentou três heurísticas para gerar compartimentos, a saber: a Heurística da Decomposição, a Heurística do Melhor Compartimento e a Heurística dos “z” Melhores Compartimentos. Neste trabalho é considerado o caso restrito (PMCR).

Hoto (2001) apresentou, além da formulação concisa do Problema da Mochila Compartimentada e de um estudo aprofundado do PCBA, um algoritmo de compartimentação exata para o caso irrestrito (COMPEX), uma heurística (COMPMT), juntamente com uma comparação de desempenho desses algoritmos com a Heurística da Decomposição. Também descreveu o Problema da Mochila Compartimentada 0-1, a qual calculou limitantes e determinou critérios de poda, propondo assim um algoritmo *branch-and-bound* para resolvê-lo.

No caso irrestrito do PMC, ou seja, quando as demandas não são consideradas (pode-se inserir quanto for desejável um item), Hoto (2001) considerou o conceito de compartimentos dominantes para excluir compartimentos que não farão parte da solução ótima. Como exemplo, considere um compartimento com utilidade 15 e largura 9, e outro compartimento com utilidade 13 e largura 10. Certamente existe uma solução ótima do problema em que o segundo compartimento mencionado não faz parte. Diz-se que tal compartimento é dominado pelo primeiro, e um compartimento é dito dominante se for viável e não for dominado por nenhum outro.

Assim o algoritmo COMPEX define as possíveis larguras de compartimentos (os compartimentos viáveis) e a utilidade de cada compartimento, construindo apenas compartimentos dominantes, e em seguida define a compartimentação por um problema de mochila. Note que para cada compartimento viável, ou seja, cada tamanho possível de compartimentos, um problema de mochila precisou ser resolvido, e para isso há um custo.

Visando diminuir esse custo, Hoto (2001) apresentou o algoritmo heurístico COMPMT, que usa os limitantes de Martello-Toth para estimar a utilidade dos compartimentos, e após escolher os que serão inseridos na mochila resolve-se o problema de mochila que determina a real utilidade desses compartimentos, assim a resolução desse subproblema se faz necessária apenas nos compartimentos escolhidos. Uma mudança no algoritmo consiste em, após atualizar as utilidades de alguns compartimentos, calcular uma nova compartimentação, e repetir esse processo até que não haja mais atualizações, ou seja, as utilidades dos compartimentos escolhidos são iguais às estimadas ou já atualizadas. Embora essa mudança pareça melhorar a solução (ou pelo menos não a piora), vale lembrar que, no pior caso, a utilidade estimada de todos os compartimentos é atualizada para a utilidade real, prejudicando o tempo computacional. Nas simulações descritas em Hoto (2001), embora o COMPMT não tenha alcançado solução exata em muitos casos, a função-objetivo não foi em muito alterada, o que mostra competitividade da heurística em relação ao algoritmo exato.

Na comparação com a Heurística da Decomposição, os algoritmos COMPEX e COMPMT apresentaram soluções melhores em alguns casos, lembrando que os exemplos gerados tenham sido para mochilas compartimentadas irrestritas. Para resolver o PMCR, Hoto (2001) propôs um algoritmo heurístico que trata-se de uma modificação do COMPEX.

Marques e Arenales (2007) apresentou a Heurística das “w” Capacidades. Os autores relatam terem obtido melhor performance que a Heurística dos “z” Melhores Compartimentos em suas simulações. No mesmo trabalho também encontra-se uma relaxação linear para o PMC, cuja solução traz um limitante superior. Em Leão *et al* (2011), o limitante superior dado por esta relaxação foi utilizado para verificar a otimalidade de solução por algumas heurísticas, e o tempo computacional de cálculo desse limitante para os exemplares rodados foi em torno de 0,01 segundos.

Outro limitante superior descrito em Leão *et al* (2011) utiliza geração de colunas para resolver o problema mestre relaxado, onde a cada iteração simplex q subproblemas (um para cada classe) são calculados. Esta heurística obteve tempo um pouco mais rápido que a relaxação do problema feita por Marques e Arenales (2007), porém o limitante superior foi um pouco maior (ou seja, pior) na maioria dos exemplares testados.

Leão *et al* (2011) também apresenta uma heurística híbrida que explora as ideias das heurísticas dos “z” melhores compartimentos e das “w” capacidades, ou seja, para cada uma das “w” capacidades de cada compartimento, guardam-se as “z” melhores soluções. Note que a Heurística dos “z” Melhores Compartimentos é o caso particular da heurística híbrida com “w”=1, e a Heurística das “w” capacidades é o caso particular da heurística híbrida com “z”=1. Logo a heurística híbrida é melhor que as duas em termos de poder gerar soluções melhores. Os autores relataram que, além da solução melhorar quando “z” e “w” aumentam, ocorre de a solução ter seu desempenho mais influenciado com o aumento de “w” do que com o aumento de “z”.

Leão (2013) apresentou um estudo de um caso bidimensional do PMC, onde uma placa é dividida horizontalmente em compartimentos paralelos (cada compartimento é visto como uma faixa da placa mestre, e a largura da faixa é limitada entre um valor mínimo e um máximo), e em cada compartimento itens retangulares são alocados. Foram apresentadas formulações para diferentes casos e heurísticas. A autora também apresentou uma heurística nova para o PMC unidimensional, que usa geração de colunas, e na geração de compartimentos usa a heurística híbrida.

A primeira tentativa de eliminar a não-linearidade do modelo original da compartimentação de uma mochila é expressa em Hoto *et* al (2006), que também desenvolveu heurísticas de retro-alimentação. Cruz (2010) apresentou testes baseados nessa ideia de eliminar a não-linearidade, e Leão *et al* (2011) usou estas ideias para comparar os resultados de suas heurísticas. Foram estas simulações a motivação da pesquisa apresentada no Capítulo 5, pois, nenhum desses trabalhos provou que uma Mochila Compartimentada é, na realidade, um problema linear.

CAPÍTULO 4

UMA MODELAGEM LINEAR PARA O PROBLEMA DA MOCHILA COMPARTIMENTADA

Neste capítulo é apresentado o modelo linear do PMCR trabalhado em Hoto *et al* (2006), e provado que este modelo não é equivalente ao original apresentado no Capítulo 3. Esta prova permite uma análise do modelo com o intuito de justificar mudanças necessárias para que seja equivalente ao modelo original. Algumas dessas mudanças já foram consideradas em Cruz (2010).

4.1 Formulação De Um Modelo Linear

A ideia para construção de um modelo linear surge do fato de que, a cada classe N_k, a quantidade de compartimentos que podem ser escolhidos não ultrapassa F_1 nem $\left\lfloor L \middle/ L^k_{\min} \right\rfloor$. Assim limitado o número de compartimentos utilizados, incluindo-se repetições implícitas, pode-se criar compartimentos repetidos de forma a eliminar as variáveis y_j do problema, sem ultrapassar um certo limitante para a quantidade de compartimentos, que será denotado por $p_k = \min\left\{F_1, \left\lfloor L \middle/ L^k_{\min} \right\rfloor\right\}$.

Observação 4. *Diferente da abordagem clássica, que considera todos os compartimentos viáveis (bem determinados) e determina quais devem ser construídos (incluídos na mochila) e em quais quantidades, na abordagem linear serão construídos p_k compartimentos para cada classe, sendo que estes compartimentos não possuem determinação prévia nas quantidades de cada item que os compõem (essas serão as variáveis), sendo que alguns desses compartimentos serão nulos, ou seja, a quantidade em que cada item será incluído no compartimento será nula.*

Sendo assim, embora teoricamente sejam construídos $\sum_{k=1}^{q} p_k$ compartimentos, alguns serão nulos e, na prática, não irão compor a compartimentação da mochila. Além disso, outros podem ser implicitamente repetidos, ou seja, com mesmas quantidades de cada item.

Nesta abordagem é feita uma modificação na indexação dos compartimentos. No Exemplo 1 do Capítulo 3, foi esclarecido que tanto a indexação dos itens quanto a dos compartimentos na modelagem original do PMCR são globais. A maneira de indexar os itens (globalmente ou localmente) é teoricamente irrelevante, pois apenas altera os elementos dos conjuntos N_k e cada item continua ligado com sua respectiva classe, por isso foi mantida uma indexação global, facilitando a descrição das demandas, uma vez que basta considerar o valor de demanda d_i para o item i, ao invés de haver necessidade de indexar a classe a qual pertence o item i em seu valor de demanda. Já a indexação dos compartimentos foi considerada local, pois nesta abordagem (HOTO *et al,* 2006; CRUZ, 2010), para cada classe k haverá p_k compartimentos a serem construídos, e assim atribui-se índices variando de 1 a p_k para tais compartimentos.

Devido a essa mudança na indexação dos compartimentos, faz-se necessário acrescentar o índice da classe nas variáveis. Dessa forma, a variável a_{ij}^k representa a quantidade de itens de índice i do compartimento j, onde o item i pertence à N_k e j é um compartimento referente também à classe k. Com a nova indexação também foi necessário modificar a variável δ_j para δ_j^k, a qual possui valor 1 se o compartimento de índice j referente à classe k é não-nulo, e valor 0 no caso contrário.

Exemplo 2. *Considere 2 classes e 5 itens, sendo 3 itens na primeira classe e 2 itens na segunda classe. Neste caso,* $q=2$, $N=\{1,2,3,4,5\}$, $N_1=\{1,2,3\}$ *e* $N_2=\{4,5\}$. *Suponha* $L_{\min}^1=8\,uc$, $L_{\min}^2=7\,uc$, $L_{\max}^1=10\,uc$, $L_{\max}^2=12\,uc$. *Considere também* $L=22$ *e* $F_1=3$. *Assim tem-se* $p_1=\min\{3,\lfloor 22/8\rfloor\}=2$ *e* $p_2=\min\{3,\lfloor 22/7\rfloor\}=3$. *Significa que no máximo 2 compartimentos da classe* N_1 *e no máximo 3 compartimentos da classe* N_2 *serão inseridos na mochila. Teoricamente, serão construídos (na nova abordagem) 5 compartimentos, mas alguns desses podem ser nulos (*$a_{ij}^k=0$ *para todo* $i\in N_k$ *onde* $j=1,\ldots,p_k$*).*

Agora, os índices dos compartimentos construídos com itens indexados em N_1 *pertencem à* $\{1,2\}$ *e os índices dos compartimentos com itens*

indexados em N_2 *estão em* $\{1,2,3\}$*, observe que a indexação dos compartimentos não é mais global como foi tratada no Exemplo 1 do Capítulo 3.*

Além disso, os compartimentos não podem mais ser todos previamente construídos e indexados, o que é irrelevante, pois, como já mencionado no Capítulo 3, na prática é inviável criar todos os possíveis compartimentos.

Seguindo estas ideias, o primeiro modelo linear (HOTO *et al,* 2006) formulado para o PMCR foi:

Maximizar:

$$\sum_{i\in N}\sum_{k=1}^{q}\sum_{j=1}^{p_k} u_i a_{ij}^k \tag{4.1}$$

sujeito a:

$$\sum_{i\in N}\sum_{k=1}^{q}\sum_{j=1}^{p_k} l_i a_{ij}^k \le L \tag{4.2}$$

$$\delta_j^k L_{\min}^k \le \sum_{i\in N_k} l_i a_{ij}^k \le \delta_j^k L_{\max}^k,\ j=1,\ldots,p_k,\ k=1,\ldots,q \tag{4.3}$$

$$\sum_{k=1}^{q}\sum_{j=1}^{p_k} a_{ij}^k \le d_i,\ i\in N = N_1\cup\cdots\cup N_q \tag{4.4}$$

$$\sum_{k=1}^{q}\sum_{j=1}^{p_k} \delta_j^k \le F_1 \tag{4.5}$$

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

$$\sum_{i\in N_k} l_i a_{ij}^k \ge \sum_{i\in N_k} l_i a_{i(j+1)}^k,\ j=1,\ldots,p_k\text{-}1,\ k=1,\ldots,q \tag{4.7}$$

$$\delta_j^k \in \{0,1\} \text{ e } a_{ij}^k \ge 0 \text{ e inteiros, } i\in N,\ j=1,\ldots,p_k \tag{4.8}$$

$$\text{e } k=1,\ldots,q\,.$$

A função-objetivo apresentada em (4.1) representa a soma das utilidades de todos os itens escolhidos em cada compartimento. A restrição (4.2) garante que a soma das larguras dos itens escolhidos é limitada pela capacidade da mochila. As restrições em (4.3) limitam a largura dos compartimentos não-nulos e anula os coeficientes dos compartimentos que não serão utilizados, sendo $\delta_j^k = 1$ se o compartimento de índice j da classe N_k for não-nulo e $\delta_j^k = 0$ no caso contrário.

As restrições em (4.4) restringem a quantidade de cada item por sua respectiva demanda. A restrição (4.5) limita a quantidade de compartimentos não-nulos e em (4.6) são restringidas as quantidades de itens em cada compartimento. As restrições em (4.7) eliminam soluções simétricas. As restrições em (4.8) estabelecem o domínio das variáveis.

4.2 O MODELO (4.1)-(4.8) NÃO É EQUIVALENTE AO ORIGINAL

Nesta seção é analisado o modelo (4.1)-(4.8) e discutidas suas falhas, justificando as mudanças apresentadas em Cruz (2010) e nesta dissertação (Capítulo 5).

Teorema 1. *Os modelos (3.8)-(3.14) e (4.1)-(4.8) não são equivalentes.*

Prova: Considere o exemplar que possui os seguintes coeficientes: $q=2$, $N_1=\{1\}$, $N_2=\{2,3\}$, $l_1=2$, $l_2=3$, $l_3=5$, $u_1=1$, $u_2=1$, $u_3=2$, $d_1=2$, $d_2=2$, $d_3=1$, $L^1_{\min}=1$, $L^1_{\max}=3$, $L^2_{\min}=2$, $L^2_{\max}=6$, $F_1=2$, $F_2=3$ e $L=10$.

Construindo todos os compartimentos viáveis, obtém-se (em uma das possibilidades) $V_1=\{1\}$ e $V_2=\{2,3,4\}$, com $a_{11}=1$, $a_{21}=0$, $a_{31}=0$, $a_{12}=0$, $a_{22}=1$, $a_{32}=0$, $a_{13}=0$, $a_{23}=0$, $a_{33}=1$, $a_{14}=0$, $a_{24}=2$ e $a_{34}=0$. Satisfazendo as restrições (3.9)-(3.14), há 10 possibilidades para as variáveis y_1, y_2, y_3 e y_4. De fato, (y_1,y_2,y_3,y_4) pode ser (1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1), (1,1,0,0), (1,0,1,0), (1,0,0,1), (0,1,1,0), (2,0,0,0) ou (0,2,0,0), e os valores da função-objetivo (3.8), dentro dessas possibilidades, são respectivamente 1, 1, 2, 2, 2, 3, 3, 3, 2 e 2. Portanto, $y_1=1$, $y_2=0$, $y_3=0$ e $y_4=1$ é uma solução ótima do modelo (3.8)-(3.14) para o exemplar escolhido, cujo valor ótimo da função-objetivo (3.8) é 3.

Por outro lado, $p_1=\min\{2,\lfloor 10/1\rfloor\}=2$ e $p_2=\min\{2,\lfloor 10/3\rfloor\}=2$. Assim, as variáveis do modelo (4.1)-(4.8) são a^1_{11}, a^1_{21}, a^1_{31}, δ^1_1, a^1_{12}, a^1_{22}, a^1_{32}, δ^1_2, a^2_{11}, a^2_{21}, a^2_{31}, δ^2_1, a^2_{12}, a^2_{22}, a^2_{32} e δ^2_2. Tome $a^1_{11}=a^1_{21}=a^1_{31}=\delta^1_1=1$ e $a^1_{12}=a^1_{22}=a^1_{32}=a^2_{11}=$ $=a^2_{21}=a^2_{31}=a^2_{12}=a^2_{22}=a^2_{32}=\delta^1_2=\delta^2_1=\delta^2_2=0$.

Será provado que esses valores definidos estão na região factível do modelo (4.1)-(4.8). Tem-se $\sum_{i\in N}\sum_{k=1}^{2}\sum_{j=1}^{2} l_i a_{ij}^k = 2\cdot 1+3\cdot 1+5\cdot 1=10$, e assim a restrição (4.2) é satisfeita. Para $j=1$ e $k=1$, $\sum_{i\in N_1} l_i a_{ij}^1 = 2\cdot 1=2$ e $L_{\min}^1 = 1\le 2\le 3 = L_{\max}^1$, caso contrário $a_{ij}^k = \delta_j^k = 0$ para todo $i\in N_k$, logo a restrição (4.3) é satisfeita.

Agora, $\sum_{k=1}^{2}\sum_{j=1}^{2} a_{1j}^k = a_{11}^1 = 1\le 2 = d_1$, $\sum_{k=1}^{2}\sum_{j=1}^{2} a_{2j}^k = a_{21}^1 = 1\le 2 = d_2$ e $\sum_{k=1}^{2}\sum_{j=1}^{2} a_{3j}^k = a_{31}^1 = 1 = d_3$, logo a restrição (4.4) é satisfeita. Também $\sum_{k=1}^{2}\sum_{j\in\{1,2\}} \delta_j^k = \delta_1^1 = 1\le 2 = F_1$, o que garante a restrição (4.5). Para $j=1$ e $k=1$, $\sum_{i\in N_k} a_{ij}^k = 3 = F_2$, caso contrário $\sum_{i\in N_k} a_{ij}^k = 0 \le F_2$, logo a restrição (4.6) também é satisfeita. Por fim, $\sum_{i\in N_1} l_i a_{i1}^1 = 3 \ge 0 = \sum_{i\in N_1} l_i a_{i2}^1$ e $\sum_{i\in N_2} l_i a_{i1}^2 = \sum_{i\in N_2} l_i a_{i2}^2 = 0$, satisfazendo (4.7). Portanto a solução construída é factível no modelo (4.1)-(4.8).

Nesta solução factível a função-objetivo assume o valor $\sum_{i\in N}\sum_{k=1}^{2}\sum_{j=1}^{2} u_i a_{ij}^k = 1\cdot 1+1\cdot 1+2\cdot 1=4$, logo a solução ótima do modelo (4.1)-(4.8) para o exemplar escolhido é maior do que ou igual a 4, e como a solução ótima do modelo (3.8)-(3.14) é 3, segue que estes modelos não são equivalentes.

□

Com esta demonstração fica clara a falha no modelo (4.1)-(4.8) ao definir a_{ij}^k para $i\notin N_k$. O modelo (3.8)-(3.14) também trabalha com a_{ij} para todo $i\in N$ e $j\in V$, porém o valor $u_i a_{ij}$ (assim como $l_i a_{ij}$) não é somado à função-objetivo (nem $l_i a_{ij}$ à restrição de mochila) se $i\in N_r$ e $j\in V_s$, com $r\neq s,\ r,s=1,\ldots,q$.

Por exemplo, caso fosse definido um quinto compartimento na abordagem clássica (similar ao compartimento construído com $j=1$ e $k=1$ na abordagem linear), de forma que $V_1=\{1,5\}$ e $a_{15}=a_{25}=a_{35}=1$, a restrição (4.3) seria

satisfeita da mesma forma, visto que $\sum_{i \in N_1} l_i a_{i5} = 2 \cdot 1 = 2$, mas os valores a_{25} e a_{35}, que não foram contabilizados na capacidade do compartimento de índice 5, também não serão na restrição de mochila (3.9) e na função-objetivo (3.8), pois $2,3 \in N_2$ e $5 \in V_1$. As incógnitas a_{25} e a_{35} estão inativas no modelo original.

Portanto, as funções-objetivo e as restrições de mochila dos modelos não se identificam. No próximo capítulo é formalizado um novo modelo linear para o PMCR, e provado ser equivalente ao modelo original.

CAPÍTULO 5

MOCHILAS COMPARTIMENTADAS SÃO LINEARES: UMA DEMONSTRAÇÃO

Neste capítulo é apresentado um novo modelo linear para o Problema da Mochila Compartimentada Restrito, descritas simulações numéricas e provado que este modelo é equivalente ao clássico, permitindo-se afirmar que o Problema da Mochila Compartimentada é linear.

5.1 UMA NOVA MODELAGEM LINEAR PARA O PROBLEMA DA MOCHILA COMPARTIMENTADA

A ideia de se trabalhar com p_k compartimentos para cada classe ainda é essencial para a construção de um modelo linear. Sendo assim, as observações do Capítulo 4 são válidas para o modelo a ser desenvolvido, incluindo a indexação local dos compartimentos.

Nesta nova abordagem, as variáveis a_{ij}^k serão definidas apenas se o item i e o compartimento j são referentes à mesma classe N_k. Isso gera pequenas modificações em relação ao modelo (4.1)-(4.8), com destaque para a não necessidade de se incluir o conjunto V e os subconjuntos V_k no modelo.

Na função-objetivo (e na restrição de mochila), $u_i a_{ij}^k$ (e $l_i a_{ij}^k$) só será somado se $i \in N_k$, então a função-objetivo altera-se para:

$$\sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} u_i a_{ij}^k ,$$

e a restrição de mochila torna-se:

$$\sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} l_i a_{ij}^k \leq L .$$

A restrição que delimita a capacidade de cada compartimento permanece como em (4.3).

Dado $i \in N$, existe $k = 1,\ldots,q$ tal que $i \in N_k$, então $\sum_{r=1}^{q} \sum_{j=1}^{p_k} a_{ij}^r = \sum_{j=1}^{p_k} a_{ij}^k$, pois $a_{ij}^r = 0$ para $i \notin N_r$, $r = 1,\ldots,q$, logo pode-se substituir a restrição (4.4) por:

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

A restrição que limita a quantidade de compartimentos não-nulos permanece como em (4.5), e na restrição que limita a quantidade de itens em cada compartimento não é necessário utilizar a notação V, então a restrição (4.6) torna-se:

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

Não é necessário incluir uma restrição análoga à (4.7), uma vez que permitir a existência de soluções simétricas não altera a solução ótima do problema.

Desta forma, o modelo linear para o Problema da Mochila Compartimentada Restrito proposto neste trabalho é:

Maximizar:

$$\sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} u_i a_{ij}^k \tag{5.1}$$

sujeito a:

$$\sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} l_i a_{ij}^k \leq L \tag{5.2}$$

$$\delta_j^k L_{\min}^k \leq \sum_{i \in N_k} l_i a_{ij}^k \leq \delta_j^k L_{\max}^k,\ j = 1, \ldots, p_k,\ k = 1, \ldots, q \tag{5.3}$$

$$\sum_{j=1}^{p_k} a_{ij}^k \leq d_i,\ i \in N_k, k = 1, \ldots, q \tag{5.4}$$

$$\sum_{k=1}^{q} \sum_{j=1}^{p_k} \delta_j^k \leq F_1 \tag{5.5}$$

$$\sum_{i \in N_k} a_{ij}^k \leq F_2,\ j = 1, \ldots, p_k,\ k = 1, \ldots, q \tag{5.6}$$

$$\delta_j^k \in \{0,1\} \text{ e } a_{ij}^k \geq 0 \text{ e inteiros, } i \in N_k,\ j = 1, \ldots, p_k \text{ e } k = 1, \ldots, q\,. \tag{5.7}$$

5.2 Ensaios Numéricos

A fim de avaliar a qualidade das soluções provenientes do modelo (5.1)-(5.7), foi implementado tal modelo e comparadas suas soluções com as da Heurística das “w” Capacidades (Seção 3.1) e com as soluções de um algoritmo exato (Algoritmo de Decomposição Exaustiva), que utiliza uma decomposição

exaustiva, proveniente da Observação 2 (Capítulo 3), e fornece, portanto, a solução ótima de cada exemplar.

Foi escolhida a Heurística das “w” Capacidades por ser reconhecida em meio às heurísticas encontradas na literatura, lembrando que o objetivo desta dissertação não é comparar heurísticas, mas observar a diferença na qualidade das soluções provindas da modelagem linear em relação às soluções heurísticas e exatas. Foi utilizado “w”=5, por ser um valor recomendado na literatura.

Quanto ao Algoritmo de Decomposição Exaustiva, ele gera, por meio de um método de ramificação, todos os compartimentos viáveis e, então, um problema de programação inteira é resolvido para definir a compartimentação da mochila. Este procedimento é, obviamente, inviável para uso prático (grandes exemplares), conforme Observação 1 (seção 1.3.1), mas, permite obter-se a solução ótima de exemplares de tamanho limitado, com a finalidade de testes teóricos. Com base em testes iniciais, foi observado que 40 itens por classe é uma quantia máxima adequada para a geração dos exemplares, uma vez que alguns exemplares de 45 itens por classe e todos os de 50 itens por classe testados obtiveram falha em sua resolução, pois geraram muitos compartimentos (milhões) e exauriram a memória da máquina durante a compartimentação. Enquanto isto, trabalhar com muitas classes de poucos itens não exigiu tanta memória. Mais detalhes do algoritmo são dados na subseção 5.2.1.

Foram organizadas sete categorias de exemplares: 5/10, 10/5, 10/10, 10/30, 30/10, 10/40 e 40/10, onde a categoria q/n é formada por exemplares de q classes e n itens em cada classe.

Foi criado um gerador aleatório com referência em Gau e Wäsher (1995), utilizando valores realísticos do problema de corte de bobinas de aço (onde as larguras são medidas em milímetros), foi definido $L = 2200$, $F_1 = 12$, $F_2 = 8$ e, para toda classe N_k, $L^k_{\min} = 375$ e $L^k_{\max} = 750$. Além disso, as larguras dos itens possuem valores aleatórios entre 55 e 350, as utilidades entre 0 e 1 com três casas de arredondamento e convertidas a valores inteiros e, por fim, a demanda total corresponde a três vezes o número total de itens, gerando as demandas dos itens distribuídas aleatoriamente conforme sugerido em Gau e Wäsher (1995). Desta maneira, foram gerados 200 exemplares para cada categoria, um total de 1400 exemplares.

As implementações foram realizadas com o uso do solver FICO® Xpress Optimizer, num Intel® Core™ i7 de 2,67 GHz com 12 GB de RAM. Os resultados podem ser observados na Tabela 2, onde constam, para cada algoritmo implementado, a **média** e o **desvio padrão** (representado por σ) aproximados dos valores das funções-objetivo (Obj) e dos tempos computacionais (T), em segundos, dos exemplares de cada categoria.

Tabela 2: Resultados dos ensaios numéricos.

Categorias:		**5/10**	**10/5**	**10/10**	**10/30**	**30/10**	**10/40**	**40/10**
Algoritmo de Decomposição Exaustiva	$\overline{Obj}$	16,175	16,104	19,651	25,489	23,974	26,903	25,210
	$\sigma(Obj)$	2,483	2,121	2,307	1,945	1,887	1,675	1,680
	$\overline{T}$	0,602	0,094	0,655	91,803	1,695	834,620	2,485
	$\sigma(T)$	1,471	1,138	0,289	34,768	0,362	564,060	0,469
Modelo Linear	$\overline{Obj}$	16,175	16,104	19,651	25,489	23,974	26,903	25,210
	$\sigma(Obj)$	2,483	2,121	2,307	1,945	1,887	1,675	1,680
	$\overline{T}$	0,321	0,213	0,339	1,153	0,541	0,875	0,680
	$\sigma(T)$	1,323	0,159	0,264	6,135	0,386	1,508	0,307
Heurística	$\overline{Obj}$	15,258	15,760	18,636	21,974	22,656	22,375	23,750
	$\sigma(Obj)$	2,293	2,050	2,285	1,852	1,850	1,680	1,667
	$\overline{T}$	0,066	0,086	0,141	0,239	0,499	0,266	0,718
	$\sigma(T)$	0,012	0,013	0,017	0,027	0,026	0,034	0,032

Fonte: próprio autor.

Em cada exemplar testado, os valores ótimos da função-objetivo obtidos com o Algoritmo de Decomposição Exaustiva e com a implementação do modelo linear são **idênticos**, o que levantou a conjectura de que isto sempre acontece, fato matematicamente comprovado na seção 5.3.

5.2.1 O Algoritmo de Decomposição Exaustiva

O método de ramificação usado para gerar todos os compartimentos viáveis é semelhante ao explorado pelo *Backtracking*. Para cada classe é gerada uma árvore que enumera os compartimentos viáveis. A busca é feita em

profundidade, conforme modifica-se os valores das variáveis a_{ij}, $i \in N_k$, onde $a_j = (a_{ij})_{i \in N_k}$ constituirá um compartimento viável da classe N_k se:

$$L_{\min}^k \leq \sum_{i \in N_k} l_i a_{ij} \leq L_{\max}^k ;$$

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

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

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

Deste modo, no desenvolvimento de um ramo, é necessário considerar o limitante superior para o tamanho de um compartimento, a demanda do item e a quantidade máxima de itens no compartimento. Então, para criar um compartimento de V_k, o máximo que pode-se atribuir a a_{ij} é:

$$\hat{a}_{ij} = \min \left\{ \left\lfloor \left(L_{\max}^k - \sum_{\substack{n \in N_k \\ n \neq i}} l_n \hat{a}_{nj} \right) \Big/ l_i \right\rfloor, d_i, \bar{F} \right\},$$

onde $\bar{F}$ é a quantidade de itens que ainda podem ser inseridos no compartimento, dada por $\bar{F} = F_2 - \sum_{\substack{n \in N_k \\ n \neq i}} \hat{a}_i$.

O ramo desenvolvido deve fornecer uma capacidade do compartimento construído maior ou igual do que $L_{\min}^k$ para ser viável. Neste caso, a solução deve ser salva (é criado o compartimento) e a construção da árvore continua. Caso contrário, pode-se fazer a volta no nó, uma vez que, se $\sum_{n \in N_k} l_n \hat{a}_{nj} < L_{\min}^k$, então diminuir o valor da última variável produzirá um compartimento cuja capacidade é, ainda, menor do que o mínimo aceito. Com isto, a cada ramo desenvolvido até a folha, a solução é testada e, se factível, diminui-se o valor da última variável e faz-se novamente o teste, e isto se repete até que a solução seja infactível ou as possibilidades para a última variável estejam esgotadas. Quando o ramo produz um compartimento infactível, faz-se a volta nos nós.

Visando diminuir a quantidade de ramos testados, com base em ensaios numéricos iniciais, considerou-se ordenar as variáveis por ordem decrescente das larguras dos itens. Uma justificativa para o funcionamento desta regra é dada ao final desta subseção, com o suporte de um exemplo. O Quadro 8 descreve o gerador de compartimentos.

Quadro 8: Primeira Etapa do Algoritmo da Decomposição Exaustiva

Gerador de Compartimentos

Entrada: $N_k, L^k_{\min}, L^k_{\max}, l_i, d_i, F_2$.

Saída: V, a_{ij}.

Inicialização: faça $j = 1$.

Para cada $k = 1,\ldots,q$ faça

Ordene os itens em ordem decrescente de largura. Se $|N_k| = m$, então considere os índices dos itens variando entre 1 e m, com $\hat{l}_i \geq \hat{l}_{i+1},\ i = 1,\ldots,m-1$.

Faça $r = 0$ e $\bar{F} = F_2$;

Passo 1 (desenvolvimento de um ramo)**:**

Para $i = r+1,\ldots,m$ faça

$$\hat{a}_i = \min\left\{\left\lfloor\left(L^k_{\max} - \sum_{n=1}^{i-1} l_n \hat{a}_n\right) \Big/ \hat{l}_i\right\rfloor, \hat{d}_i, \bar{F}\right\};$$

e $\bar{F} = \bar{F} - \hat{a}_{ij}$;

Faça $r = m$;

Passo 2 (salvar solução corrente)**:**

Se $\sum_{n \in N_k} \hat{l}_n \hat{a}_n \geq L^k_{\min}$, então

salve o compartimento ($j \in V_k$ e $a_{ij} = \hat{a}_{i'}$ onde o item de índice i corresponde ao índice i' na reordenação);

e faça $j = j+1$;

caso contrário, vá ao passo **4**;

Passo 3 (ramifica)**:**

Se $\hat{a}_r > 0$ então

faça $\hat{a}_r = \hat{a}_r - 1$;

faça $\bar{F} = \bar{F} + 1$;

se $r = m$ então

volte ao passo **2**;

Quadro 9: Primeira Etapa do Algoritmo da Decomposição Exaustiva (continuação)

caso contrário, volte ao passo **1**;

Passo 4 (volta):

Se $r = 1$ então

pare!

caso contrário, faça

$r = r - 1$;

$\bar{F} = \bar{F} - \hat{a}_r$;

e volte ao passo **3**;

Fonte: Próprio autor.

Considere, como exemplo, $N = N_1 = \{1,2,3\}$, $l_1 = 2$, $l_2 = 3$, $l_3 = 5$, $d_1 = 4$, $d_2 = 2$, $d_3 = 2$, $L_{\min}^k = 8$, $L_{\max}^k = 10$ e $F_2 = 3$. Será gerada uma árvore para $k = 1$, que construirá os compartimentos de $V = V_1$. Reordenando os itens obtém-se $\hat{l}_1 = 5$, $\hat{l}_2 = 3$, $\hat{l}_3 = 2$, $\hat{d}_1 = 2$, $\hat{d}_2 = 2$ e $\hat{d}_3 = 4$. De início $r = 0$ e $\bar{F} = 3$. Assim, o primeiro ramo construído no Passo 1 é dado por:

$$\hat{a}_1 = \min\{\lfloor 10/5 \rfloor, 2, 3\} = 2;$$

$$\hat{a}_2 = \min\{\lfloor (10-10)/3 \rfloor, 2, 1\} = 0;$$

$$\hat{a}_3 = \min\{\lfloor (10-10)/2 \rfloor, 4, 1\} = 0.$$

O Passo 1 também faz $\bar{F} = 1$ e $r = 3$. A capacidade do compartimento construído é 10, então ele deve ser salvo (Passo 2), com $a_{11} = 0$, $a_{21} = 0$ e $a_{31} = 2$. Como $\hat{a}_3 = 0$, o Passo 3 (ramificação) é ignorado e faz-se o Passo 4 (volta). Então $r = 2$ e como $\hat{a}_2 = 0$, retorna-se ao Passo 4 e faz-se $r = 1$. Então o Passo 3 faz:

$$\hat{a}_1 = 1;$$

e $\bar{F} = 2$. No Passo 1 são calculados:

$$\hat{a}_2 = \min\{\lfloor (10-5)/3 \rfloor, 2, 2\} = 1;$$

$$\hat{a}_3 = \min\{\lfloor (10-8)/2 \rfloor, 4, 1\} = 1;$$

e $r = 3$. O compartimento gerado possui capacidade $10 \geq 8$, logo é salvo com $a_{12} = 1$, $a_{22} = 1$ e $a_{32} = 1$. No Passo 3, faz-se:

$$\hat{a}_3 = 0;$$

e $\bar{F} = 1$. O compartimento gerado possui capacidade 8, logo é salvo com $a_{13} = 0$, $a_{23} = 1$ e $a_{33} = 1$. Como $\hat{a}_3 = 0$, o Passo 3 é ignorado, e o Passo 4 faz $r = 2$. Então o Passo 3 faz:

$$\hat{a}_2 = 0;$$

e $\bar{F} = 2$. Voltando-se ao Passo 1, tem-se:

$$\hat{a}_3 = \min\{\lfloor (10-5)/2 \rfloor, 4, 2\} = 2;$$

e $\bar{F} = 0$. Isto cria um compartimento (Passo 2) de capacidade 9, fazendo $a_{14} = 2$, $a_{24} = 0$ e $a_{34} = 1$. No Passo 3 é feito:

$$\hat{a}_3 = 1;$$

e $\bar{F} = 1$. Volta-se ao Passo 2, mas a capacidade do possível compartimento é 7, logo ele não é gravado e o Passo 4 é chamado, fazendo $r = 2$ e $\bar{F} = 2$. Como $\hat{a}_2 = 0$, o Passo 4 torna a fazer $r = 1$ e o Passo 3 faz:

$$\hat{a}_1 = 0;$$

e $\bar{F} = 3$. O Passo 1 então calcula:

$$\hat{a}_2 = \min\{\lfloor 10/3 \rfloor, 2, 3\} = 2;$$

$$\hat{a}_3 = \min\{\lfloor (10-6)/2 \rfloor, 4, 1\} = 1;$$

Assim, obtém-se (Passo 2) um compartimento de capacidade 8 com coeficientes $a_{15} = 1$, $a_{25} = 2$ e $a_{35} = 0$. Ramificando-se (Passo 3), faz-se:

$$\hat{a}_3 = 0;$$

e $\bar{F} = 1$. O mínimo permitido para a capacidade de um compartimento é violado. Então, faz-se a volta (Passo 4) com $r = 2$, e uma nova ramificação (Passo 3) gera:

$$\hat{a}_2 = 1;$$

além de $\bar{F} = 2$. Então, no Passo 1 é calculado:

$$\hat{a}_3 = \min\{\lfloor (10-3)/2 \rfloor, 4, 2\} = 2;$$

Novamente o compartimento é inviável. O Passo 4 torna $r = 2$ e o Passo 3 faz:

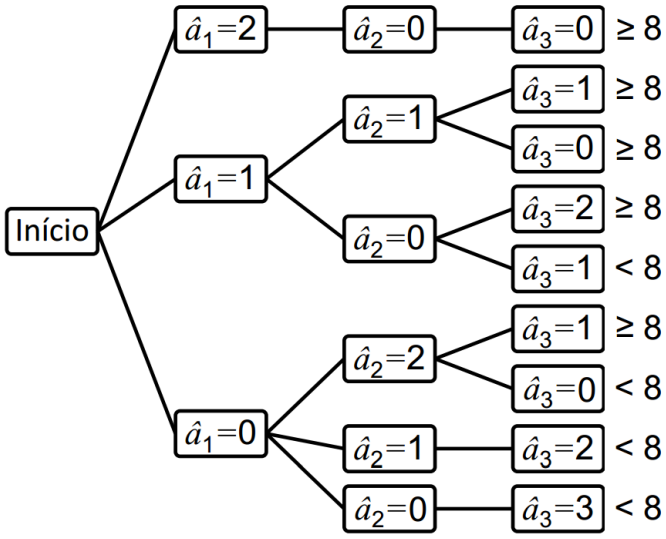
$$\hat{a}_2 = 0;$$

e $\bar{F} = 3$. Então, do Passo 1, obtém-se:

$$\hat{a}_3 = \min\{\lfloor 10/2 \rfloor, 4, 3\} = 3;$$

gerando um compartimento de capacidade 6, logo inviável. É então feita a volta, mas $\hat{a}_2 = 0$, então o Passo 4 é novamente chamado e faz $r = 1$. Como $\hat{a}_1 = 0$, o Passo 3 leva novamente ao Passo 4, e o algoritmo é encerrado. Assim obtém-se a árvore esboçada na Figura 9.

Figura 9: Construção de compartimentos.



Fonte: Próprio autor.

No canto direito da Figura 9, são marcados os compartimentos factíveis (capacidade ≥ 8) e infactíveis (capacidade < 8) ramificados. Assim $V = V_1 = \{1,2,3,4,5\}$ com $(a_{11}, a_{21}, a_{31}) = (0,0,2)$, $(a_{12}, a_{22}, a_{32}) = (1,1,1)$, $(a_{13}, a_{23}, a_{33}) = (0,1,1)$, $(a_{14}, a_{24}, a_{34}) = (2,0,1)$ e $(a_{15}, a_{25}, a_{35}) = (1,2,0)$.

Observe que 9 ramos foram construídos para a obtenção dos 5 compartimentos factíveis. Sem ordenar as variáveis, seriam desenvolvidos 13 ramos. A justificativa para que menos ramos precisem ser desenvolvidos com as variáveis ordenadas vem da regra de fazer-se a volta quando um compartimento inviável é encontrado, ao invés de ramificar diminuindo o valor da última variável. Com itens pequenos nas folhas, é possível um compartimento ser infactível mesmo com uma quantidade razoável desses itens incluídos, como pode ser visualizado no final da ramificação mostrada na Figura 9. Com itens grandes, é mais comum que a entrada ou não de uma unidade no compartimento o torne factível ou não. Logo itens grandes no final da árvore não darão muita vantagem nesta “poda” que o Passo 2 faz ao chamar diretamente o Passo 4 em caso de inviabilidade.

Após a geração de todos os compartimentos viáveis, o Algoritmo da Decomposição Exaustiva consiste na resolução do problema (3.21)-(3.25).

5.3 MOCHILAS COMPARTIMENTADAS SÃO LINEARES

Esta seção contém uma prova de que o modelo do PMCR formalizado neste capítulo possui a mesma solução do modelo original (HOTO, 2001) formalizado no Capítulo 3. A demonstração será dividida em três partes, começando por um lema e, então, dividindo a prova do resultado principal em duas partes.

5.3.1 Uma Relação Entre os Modelos e a Construção da Notação a Ser Utilizada

O Lema 1 estabelece uma relação entre os modelos (3.8)-(3.14) e (5.1)-(5.7).

Lema 1. *Dada uma solução*

$$y = \sum_{j \in V_1} \left(\sum_{i \in N_1} u_i a_{ij} \right) y_j + \cdots + \sum_{j \in Vq} \left(\sum_{i \in Nq} u_i a_{ij} \right) y_j$$

do modelo (3.8)-(3.14), existem valores β_{ij}^k, *com* $i \in N_k$, $j = 1, \ldots, p_k$ *e* $k = 1, \ldots, q$ *tais que:*

$$\sum_{j=1}^{p_k} \beta_{ij}^k = \sum_{j \in V_k} a_{ij} y_j, \text{ para todo } i \in N_k, \; k = 1, \ldots, q.$$

Analogamente, dada uma solução

$$z = \sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} u_i b_{ij}^k$$

e $\eta_j^k \in \{0,1\}$ $(j = 1, \ldots, p_k, \; k = 1, \ldots, q)$ *do modelo (5.1)-(5.7), existem valores* α_{ij} *e* z_j, *com* $i \in N$ *e* $j \in V$ *tais que:*

$$\sum_{j=1}^{p_k} b_{ij}^k = \sum_{j \in V_k} \alpha_{ij} z_j, \text{ para todo } i \in N_k, \; k = 1, \ldots, q.$$

Prova: Para cada $k = 1, \ldots, q$, será mostrado que $\sum_{j \in V_k} y_j \leq p_k$, onde $p_k = \min\{F_1, \lfloor L / L_{\min}^k \rfloor\}$, uma vez que $\sum_{k=1}^{q} \sum_{j \in V_k} y_j \leq F_1$, e como $\sum_{i \in N_k} l_i a_{ij} \geq L_{\min}^k$ se o compartimento de índice j é não-nulo, tem-se:

$$L \geq \sum_{j\in V_1}\left(\sum_{i\in N_1} l_i a_{ij}\right) y_j + \cdots + \sum_{j\in Vq}\left(\sum_{i\in Nq} l_i a_{ij}\right) y_j$$
$$\geq \sum_{j\in V_k}\left(\sum_{i\in N_k} l_i a_{ij}\right) y_j$$
$$\geq L^k_{min} \sum_{j\in V_k} y_j,$$

e por isso $\sum_{j\in V_k} y_j \leq \left\lfloor L/L^k_{\min} \right\rfloor$, pois $\sum_{j\in V_k} y_j$ é um número inteiro.

Agora, para cada item de índice $i \in N$ tem-se que:

$$\sum_{j\in V_k} a_{ij} y_j = a_{is_1} y_{s_1} + a_{is_2} y_{s_2} + \ldots + a_{is_r} y_{s_r}$$
$$= \underbrace{a_{is_1} + \ldots + a_{is_1}}_{y_{s_1} \text{ vezes}} + \underbrace{a_{is_2} + \ldots + a_{is_2}}_{y_{s_2} \text{ vezes}} + \ldots + \underbrace{a_{is_r} + \ldots + a_{is_r}}_{y_{s_r} \text{ vezes}},$$

onde assume-se que $V_k = \{s_1, \ldots, s_r\}$.

Uma vez que $y_{s_1} + y_{s_2} + \ldots + y_{s_r} \leq p_k$, pode-se definir

$$(\overline{\beta}^k_{i1}, \overline{\beta}^k_{i2}, \ldots, \overline{\beta}^k_{ip_k}) = (\underbrace{a_{is_1}, \ldots, a_{is_1}}_{y_{s_1} \text{ vezes}}, \underbrace{a_{is_2}, \ldots, a_{is_2}}_{y_{s_2} \text{ vezes}}, \ldots, \underbrace{a_{is_r}, \ldots, a_{is_r}}_{y_{s_r} \text{ vezes}}, \underbrace{0, \ldots, 0}_{p_k - \sum_{j\in V_k} y_j \text{ vezes}}).$$

Observação 5. *Observe que fixado $j = 1, \ldots, p_k$, existe $j' \in V_k$ tal que $\beta^k_{ij} = a_{ij'}$, ou seja, os elementos β^k_{ij} podem também ser entendidos como os coeficientes do compartimento de índice j da abordagem linear, que são os mesmos coeficientes do compartimento de índice j' da abordagem não-linear. Exceção a essa observação faz-se apenas no caso em que β^k_{ij} foi definido com valor nulo para todo $i \in N_k$.*

Além disso, como desejado, a construção feita garante que $\sum_{j=1}^{p_k} \beta^k_{ij} = \sum_{j\in V_k} a_{ij} y_j$ para cada item de índice $i \in N_k$, $k = 1, \ldots, q$.

Agora, considere os conjuntos:

$$W_1 = \{1, \ldots, p_1\}$$
$$W_2 = \{p_1 + 1, \ldots, p_1 + p_2\}$$
$$\vdots$$
$$W_k = \{p_1 + \cdots + p_{k-1} + 1, \ldots, p_1 + \cdots + p_k\}$$
$$\vdots$$
$$W_q = \{p_1 + \cdots + p_{q-1} + 1, \ldots, p_1 + \cdots + p_{q-1} + p_q\}$$
e
$$W = W_1 \cup \cdots \cup W_q.$$

Dado $j \in V$, se $j \notin W$, defina $z_j = 0$. Se $j \in W_k$, para algum $k = 1, \ldots, q$, então defina $z_j = \eta^k_{j-(p_1+\cdots+p_{k-1})}$.

Da mesma forma, defina

$$\alpha_{ij} = \begin{cases} 0, \text{ se } j \notin W_k \\ b^k_{i,j-(p_1+\cdots+p_{k-1})}, \text{ se } j \in W_k \end{cases}, \forall i \in N_k,$$

ou seja,

$$(\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ip_1}) = (b^1_{i1}, b^1_{i2}, \ldots, b^1_{ip_1}), \ \forall i \in N_1$$
$$(\alpha_{i(p_1+1)}, \ldots, \alpha_{i(p_1+p_2)}) = (b^2_{i1}, b^2_{i2}, \ldots, b^2_{ip_2}), \ \forall i \in N_2$$
$$\vdots$$
$$(\alpha_{i(p_1+\cdots+p_{q-1}+1)}, \alpha_{i2}, \ldots, \alpha_{i(p_1+\cdots+p_q)}) = (b^q_{i1}, b^q_{i2}, \ldots, b^q_{ip_q}), \ \forall i \in N_q.$$

Como todos os compartimentos com índices em W_k são formados por itens indexados em N_k, em vista do que está na Observação 3 (Capítulo 3), pode-se admitir que $W_k \subset V_k$ para todo $k = 1, \ldots, q$, uma vez que há uma maneira de definir os índices de V_k desta forma, sem alterar a solução ótima do problema.

Tem-se $\sum_{j \in W_k} \alpha_{ij} z_j = \sum_{j \in V_k} \alpha_{ij} z_j$, pois se $j \in V_k - W_k$, tanto α_{ij} quanto z_j foram definidos com valores nulos. Também tem-se que $\sum_{j=1}^{p_k} b^k_{ij} = \sum_{j \in W_k} \alpha_{ij} z_j$, pois se $z_j = 0$ é porque $j \notin W$ ou $\eta^k_{j-(p_1+\cdots+p_{k-1})} = 0$ que implica por (5.3) que $b^k_{i,j-(p_1+\cdots+p_{k-1})} = 0$. Assim, pode-se estabelecer uma relação “um a um” de $b^k_{ij'}$ ($j' = 1, \ldots, p_k$) com $\alpha_{ij} z_j$ ($j = p_1 + \cdots + p_{k-1} + 1, \ldots, p_1 + \cdots + p_{k-1} + p_k$) e concluir que $\sum_{j=1}^{p_k} b^k_{ij} = \sum_{j \in W_k} \alpha_{ij} z_j$.

Portanto, por transitividade, vale $\sum_{j=1}^{p_k} b_{ij}^k = \sum_{j \in V_k} \alpha_{ij} z_j$, para todo $i \in N_k$, $k = 1, \ldots, q$.

□

Teorema 2. *Os modelos (3.8)-(3.14) (não-linear) e (5.1)-(5.7) (linear) são equivalentes.*

Prova: Sejam y o valor ótimo da função-objetivo do modelo não-linear (3.8)-(3.14) e z o valor ótimo da função-objetivo do modelo linear (5.1)-(5.7). Sendo assim, para todo $j \in V$ e para todo $i \in N$, existem a_{ij}, y_j e δ_j tais que:

$$y = \sum_{j \in V_1} \left(\sum_{i \in N_1} u_i a_{ij} \right) y_j + \cdots + \sum_{j \in Vq} \left(\sum_{i \in Nq} u_i a_{ij} \right) y_j,$$

com os valores a_{ij}, y_j e δ_j satisfazendo as condições (3.9)-(3.14), e y sendo o maior valor com esta propriedade. Do mesmo modo, para todo $k = 1, \ldots, q$ e para todos $i \in N_k$, $j = 1, \ldots, p_k$ existem b_{ij}^k e η_j^k tais que:

$$z = \sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} u_i b_{ij}^k,$$

com os valores b_{ij}^k e η_j^k satisfazendo as condições (5.2)-(5.7), e sendo z o maior valor com esta propriedade.

Para provar que $y = z$ será provado, separadamente, que $y \leq z$ e $y \geq z$. Para mostrar que $y \leq z$ provar-se-á que existem, para cada $i \in N_k$, $j = 1, \ldots, p_k$ e $k = 1, \ldots, q$, valores β_{ij}^k e ζ_j^k na região viável do modelo linear tais que $\sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} u_i \beta_{ij}^k = y$, e como z é o máximo com essa propriedade poderá se concluir que $y \leq z$.

Analogamente, para provar que $z \leq y$, serão construídos valores α_{ij}, z_j e ξ_j na região viável do modelo não-linear tais que:

$$\sum_{j \in V_1} \left(\sum_{i \in N_1} u_i \alpha_{ij} \right) z_j + \cdots + \sum_{j \in Vq} \left(\sum_{i \in Nq} u_i \alpha_{ij} \right) z_j = z$$

e como y é o máximo com essa propriedade, poderá se concluir que $y \geq z$. Desta forma, a demonstração se dividirá em duas partes: *Parte 1* e *Parte 2*.

A fim de prosseguir com clareza, apresenta-se no quadro a seguir a notação a ser utilizada.

Modelo:	Não-linear	Linear
Valores definidos:	y, a_{ij}, y_j, e δ_j	z, b_{ij}^k, e η_j^k
Valores Construídos:	α_{ij}, z_j, e ξ_j	β_{ij}^k, e ζ_j^k

Na *Parte 1* serão definidos β_{ij}^k, e ζ_j^k com base em a_{ij}, y_j, e δ_j, e na *Parte 2* serão definidos α_{ij}, z_j, e ξ_j com base em b_{ij}^k, e η_j^k. É importante salientar que ao fazer referência a uma restrição de modelos anteriores que deva ser satisfeita por certas variáveis, fica subentendido que não se pretende satisfazer a restrição exatamente como está nos modelos, mas adaptada às variáveis em questão. Por exemplo, dizer que os valores β_{ij}^k satisfazem (5.4) significará dizer que $\sum_{j=1}^{p_k} \beta_{ij}^k \leq d_i$, para todo $i \in N_k$, $k = 1,\ldots,q$. Esta maneira de referenciar as equações do problema será utilizada para evitar que a demonstração torne-se muito carregada de informação.

5.3.2 Parte 1 da Demonstração

Pelo Lema 1, pode-se definir, para cada $k = 1,\ldots,q$, os valores β_{ij}^k de forma que valha a igualdade:

$$\sum_{j=1}^{p_k} \beta_{ij}^k = \sum_{j \in V_k} a_{ij} y_j, \ \forall i \in N_k, k = 1,\ldots,q. \tag{5.8}$$

e que satisfaçam também a Observação 5.

Imediatamente de (5.8) segue que:

$$
\begin{aligned}
y &= \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i a_{ij}\right) y_j + \cdots + \sum_{j\in Vq}\left(\sum_{i\in Nq} u_i a_{ij}\right) y_j \\
&= \sum_{i\in N_1}\left(\sum_{j\in V_1} y_j a_{ij}\right) u_i + \cdots + \sum_{i\in Nq}\left(\sum_{j\in V_q} y_j a_{ij}\right) u_i \\
&= \sum_{i\in N_1}\left(\sum_{j=1}^{p_1} \beta_{ij}^1\right) u_i + \cdots + \sum_{i\in Nq}\left(\sum_{j=1}^{p_q} \beta_{ij}^q\right) u_i \\
&= \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} u_i \beta_{ij}^k .
\end{aligned}
\tag{5.9}
$$

Logo $y = \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} u_i \beta_{ij}^k$, como desejado. Agora será provado que os valores definidos estão na região viável do modelo linear. A restrição (5.4) segue de (5.8), pois se $\sum_{j\in V_k} a_{ij} y_j \le \sum_{k=1}^{q}\sum_{j\in V_k} a_{ij} y_j \le d_i$ então $\sum_{j=1}^{p_k} \beta_{ij}^k \le d_i$, para todo $i \in N_k$, e todo $k = 1,\ldots,q$. E, com um cálculo análogo ao realizado em (5.9), mostra-se que $\sum_{j\in V_1}\left(\sum_{i\in N_1} l_i a_{ij}\right) y_j + \cdots + \sum_{j\in Vq}\left(\sum_{i\in Nq} l_i a_{ij}\right) y_j = \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} l_i \beta_{ij}^k$, donde (5.2) é satisfeito.

Pela Observação 5 (Lema 1), dado $j = 1,\ldots,p_k$, se $\beta_{ij}^k \neq 0$ para algum $i \in N_k$, então existe $j' \in V_k$ tal que $\beta_{ij}^k = a_{ij'}$ para todo $i \in N_k$. Caso $\beta_{ij}^k = 0$ para todo $i \in N_k$ tome $\zeta_j^k = 0$, caso contrário tome $\zeta_j^k = \delta_{j'}$. Em ambos os casos $\delta_{j'} L_{\min}^k \le \sum_{i\in N_k} l_i a_{ij'} \le \delta_{j'} L_{\max}^k$ implica que $\zeta_j^k L_{\min}^k \le \sum_{i\in N_k} l_i \beta_{ij}^k \le \zeta_j^k L_{\max}^k$, logo a restrição (5.3) também é satisfeita.

Ainda da Observação 5, tem-se $\sum_{i\in N_k} a_{ij'} = \sum_{i\in N_k} \beta_{ij}^k$, e como vale $\sum_{i\in N_k} a_{ij'} \le F_2$, tem-se $\sum_{i\in N_k} \beta_{ij}^k \le F_2$. Como j é geral $(j = 1,\ldots,p_k,\ k = 1,\ldots,q)$, pode-se concluir que a restrição (5.6) também é satisfeita.

Agora, para provar (5.5), observe que nomeando $\sum_{j\in V_k} y_j$ de R_k, tem-se:

$$\sum_{j=1}^{p_k}\zeta_j^k = \sum_{j=1}^{R_k}\zeta_j^k + \underbrace{\sum_{j=R_k+1}^{p_k}\zeta_j^k}_{=0}$$

$$= \sum_{j=1}^{R_k}\zeta_j^k$$

$$\leq R_k$$

$$= \sum_{j\in V_k} y_j, \text{ para cada } k = 1,\ldots,q.$$

Sendo assim, $\sum_{k=1}^{q}\sum_{j\in V_k} y_j \leq F_1$ implica $\sum_{k=1}^{q}\sum_{j=1}^{p_k}\zeta_j^k \leq F_1$, o que garante a restrição (5.5). Conclui-se que os elementos β_{ij}^k e ζ_j^k definidos estão na região viável do modelo linear, logo $y = \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} u_i\beta_{ij}^k \leq z$.

5.3.3 Parte 2 da Demonstração

Agora, serão construídos valores α_{ij}, z_j, e ξ_j e provado que satisfazem as restrições (3.9)-(3.14), além de que $z = \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i\alpha_{ij}\right)z_j + \cdots +$ $\sum_{j\in Vq}\left(\sum_{i\in Nq} u_i\alpha_{ij}\right)z_j$.

Sendo $z = \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} u_i b_{ij}^k$ uma solução ótima do modelo linear, com b_{ij}^k e η_j^k $(i \in N_k;\ j = 1,\ldots,p_k;\ k = 1,\ldots,q)$ satisfazendo as condições (5.2)-(5.7), segue do Lema 1 que existem α_{ij} e z_j tais que:

$$\sum_{j=1}^{p_k} b_{ij}^k = \sum_{j\in V_k}\alpha_{ij}z_j,\ \forall i \in N_k, k = 1,\ldots,q. \qquad (5.10)$$

Analogamente ao cálculo em (5.9), que contou com o uso da equação (5.8), usando (5.10) pode-se concluir que:

$$z = \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} u_i b_{ij}^k \overset{(5.10)}{\leq} \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i\alpha_{ij}\right)z_j + \cdots + \sum_{j\in Vq}\left(\sum_{i\in Nq} u_i\alpha_{ij}\right)z_j$$

e

$$\sum_{j\in V_1}\left(\sum_{i\in N_1} l_i\alpha_{ij}\right)z_j + \cdots + \sum_{j\in Vq}\left(\sum_{i\in Nq} l_i\alpha_{ij}\right)z_j \overset{(5.10)}{=} \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} l_i b_{ij}^k \overset{(5.2)}{\leq} L.$$

Logo, a restrição (3.9) é satisfeita para os elementos α_{ij} e z_j definidos.

Agora, defina $\xi_j = z_j$ para todo $j \in V$, com z_j ($j \in V$) tal como foi definido no Lema 1. Para provar a restrição (3.10), seja $j \in V_k$, para algum $k = 1,\ldots,q$. Se $j \notin W_k$ tem-se $\xi_j = \alpha_{ij} = 0$ para todo $i \in N_k$ e não há o que fazer. Caso contrário, $j \in W_k$, e assim $\xi_j = \eta^k_{j-(p_1+\cdots+p_{k-1})}$ e $\alpha_{ij} = b^k_{i,j-(p_1+\cdots+p_{k-1})}$. Como $j-(p_1+\cdots+p_{k-1}) \in \{1,\ldots,p_k\}$, tem-se por (5.3):

$$\eta^k_{j-(p_1+\cdots+p_{k-1})} L^k_{\min} \le \sum_{i\in N_k} l_i b^k_{i,j-(p_1+\cdots+p_{k-1})} \le \eta^k_{j-(p_1+\cdots+p_{k-1})} L^k_{\max},$$

logo $\xi_j L^k_{\min} \le \sum_{i\in N_k} l_i \alpha_{ij} \le \xi_j L^k_{\max}$, o que prova a condição (3.10).

A condição (3.11) é consequência da (5.4) e (5.10), pois, se $i \in N_m$ e $j \in V_n$, com $m \neq n$, $n,m = 1,\ldots,q$, foi definido $\alpha_{ij} = 0$. Logo, dado $i \in N_k$, tem-se:

$$\sum_{m=1}^{q}\sum_{j\in V_m} \alpha_{ij} z_j = \sum_{j\in V_k} \alpha_{ij} z_j \overset{(5.10)}{=} \sum_{j=1}^{p_k} b^k_{ij} \overset{(5.4)}{\le} d_i .$$

Agora, como $z_j = \begin{cases} 0, \text{ se } j \notin W \\ \eta^k_{j-(p_1+\cdots+p_{k-1})}, \text{ se } j \in W_k \end{cases}$, então

$$\sum_{j\in V_k} z_j = \sum_{j\in W_k} z_j = \sum_{j=1}^{p_k} \eta^k_j, \ \forall k = 1,\ldots,q,$$

logo $\sum_{k=1}^{q}\sum_{j=1}^{p_k} \eta^k_j \le F_1$ se, e somente se, $\sum_{k=1}^{q}\sum_{j\in V_k} z_j \le F_1$, o que prova a restrição (3.12).

Por fim, seja $j \in V$, se $j \notin W$ então $\alpha_{ij} = 0$ para todo $i \in N_k$ e segue trivialmente a restrição (3.13). Caso contrário, $j \in W_k$ para algum $k \in \{1,\ldots,q\}$ e assim $\alpha_{ij} = b^k_{i,j-(p_1+\cdots+p_{k-1})}$ para todo $i \in N_k$. Como $\sum_{i\in N_k} b^k_{i,j-(p_1+\cdots+p_{k-1})} \le F_2$, pois $j-(p_1+\cdots+p_{k-1}) \in \{1,\ldots,p_k\}$, conclui-se que $\sum_{i\in N_k} \alpha_{ij} \le F_2$, o que garante a restrição (3.13).

Assim, os elementos α_{ij}, z_j, e ξ_j definidos estão na região viável do modelo não-linear, o que implica:

$$z = \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i \alpha_{ij}\right) z_j + \cdots + \sum_{j\in Vq}\left(\sum_{i\in Nq} u_i \alpha_{ij}\right) z_j \le y .$$

Mas na *Parte 1* foi provado que $y \leq z$, logo $y = z$ e, portanto, os modelos são equivalentes, pois possuem o mesmo valor máximo para a função-objetivo.

□

CONCLUSÃO

A prova da linearidade do Problema da Mochila Compartimentada é um avanço teórico e justifica novos estudos do problema. Quanto à expansividade da demonstração, foram efetuadas simplificações, desde que não dificultasse seu entendimento, mas podem haver maneiras mais imediatas de exibir a comprovação desejada. Simplificações são sempre bem-vindas, embora o essencial seja o resultado validado.

Como o modelo matemático de um problema não é único, outros modelos podem surgir, além dos formulados nesta dissertação. Por exemplo, para este trabalho foi desenvolvido um modelo de fluxo de arcos para o Problema da Mochila Compartimentada (Apêndice B), no entanto, após a realização de alguns ensaios numéricos, verificou-se que a implementação deste modelo causa muito esforço computacional, inclusive, uma decomposição do modelo análoga à do modelo original implementada com as heurísticas conhecidas usou mais tempo computacional para encontrar a solução do que as heurísticas de decomposição do modelo original. Por isto não houve motivação para aprofundar estudos no modelo, como verificar se é equivalente ao original e desenvolver algoritmos com ele, mesmo se tratando de um modelo linear.

Quanto ao novo modelo apresentado nesta dissertação (Capítulo 5), foi considerada uma variação na abordagem trabalhada em Hoto *et al* (2006), entendendo que os elementos a_{ij}^{k} não fazem sentido prático para $i \notin N_k$, o que tornaria necessário igualar a zero esses elementos para obter-se um modelo linear equivalente ao modelo original. No novo modelo apresentado, essas variáveis não são definidas (diferente de Cruz (2010) que as define, mas as mantém inativas), além de que faz parte da descrição deste modelo apenas os p_k compartimentos construídos para cada classe, sem a necessidade de usar os conjuntos de índices de compartimentos V_k aparentes no modelo original do problema e na primeira proposta de modelagem linear.

Uma proposta de estudo posterior é verificar se as heurísticas de trabalhos anteriores podem ser adaptadas ao novo modelo. Também pode ser interessante verificar se outros modelos lineares podem melhorar o desempenho da resolução, assim como verificar métodos heurísticos que possam tirar proveito das

boas soluções provindas da programação linear sem, no entanto, causar demasiada oscilação no tempo computacional. Outra proposta é estudar a possibilidade do novo modelo induzir um algoritmo *branch-and-bound* para o problema, o que requer pesquisas no cálculo de limitantes superiores.

Com relação aos algoritmos heurísticos, trabalhos futuros podem explorar técnicas de programação que visam um melhor aproveitamento do tempo computacional, tal como usar teoria de congruência e otimização genética.

No momento, está sendo estudada a complexidade do problema, e em trabalhos a serem desenvolvidos em breve pretende-se exibir uma prova de que o Problema da Mochila Compartimentada Restrito é fortemente NP-difícil. Essas e outras ideias são discutidas no Apêndice C.

REFERÊNCIAS

ALVIN, A. C. de F. **A Hybrid Improvement Heuristics for the Bin Packing Problem and its Application to the Problem of Task Scheduling.** Tese de doutorado do programa Pós-Graduação em Informática da Pontifícia Universidade Católica do Rio de Janeiro, 2004. p. 15-18.

ANTON, H.; RORRES, C. **Álgebra linear com aplicações.** Tradução: Claus Ivo Doering. 8ª ed. Porto Alegre: Bookman , 2008.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. **Pesquisa Operacional.** Rio de Janeiro: Elsevier Editora Ltda, 2007. p. 172-176.

BALAS, E. ZEMEL, E. An algorithm for large zero-one knapsack problems. **Operations Research**, vol. 28, n. 5, p. 1130-1154, 1980.

CAMARGO, C. **Video on Demand.** Disponível em: <http://www.tecmundo.com.br/televisao/2602-video-on-demand.htm>. Acesso em: 03 jun. 2015.

CHAJAKIS, E. D.; GUIGNARD, M. Scheduling Deliveries in Vehicles with Multiple Compartments. **Journal of Global Optimization,** vol. 26, p. 43-78, 2003. p. 66-72.

CHVÁTAL, V. **Linear Programming.** New York: W.H.Freeman and Company, 1983. p. 195-211.

CHERRI, A. C.; ARENALES, N.; YANASSE, H. H. The one dimensional cutting stock problems with usable leftover: A heuristic approach. **European Journal of Operational Research**, vol. 196, p. 897-908, 2009.

CORREIA, M. H.; OLIVEIRA, J. F.; FERREIRA, J. S. Rell and sheet cutting at paper mill. **Computers & Operations Research**, vol. 31, p. 1223-1243, 2004, p. 1227-1230.

CRUZ, E. P. da. **Uma abordagem heurística linear para mochilas compartimentadas restritas.** Dissertação de Mestrado em Matemática Aplicada e Computacional. Universidade Estadual de Londrina, 2010.

DANTZIG, G. B. Discrete-variable extremum problems. **Operations Research**, vol. 5, p. 266-277, 1957.

FARIAS, P. C. **Problema de corte de estoque unidimensional com reaproveitamento de sobras: abordagem de resolução por meio de uma técnica de geração de colunas.** Dissertação de Mestrado em Matemática Aplicada e Computacional. Universidade Estadual de Londrina, 2011.

FERREIRA, J. S.; NEVES, M. A.; FONSECA E CASTRO, P. A two-phase roll cutting problem. **European Journal of Operational Research**, vol. 44, p. 185-196, 1990. p. 190-192.

FLORENTINO, H. de O.; SPADOTTO, A. F. O problema da mochila no carregamento do palhiço da cana-de-açúcar. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, XXIX, 2006, Campinas. **Anais...** Campinas: Sociedade Brasileira de Matemática Aplicada e Computacional, 2006.

GAREY, M. R.; JOHNSON, D. S. Complexity results for multiprocessor scheduling under resource constraints. **SIAM Journal on Computing**, vol. 4, n. 4, p. 397-411, 1975.

GAU, T.; WÄSCHER, G. CUTGEN1: A problem generator for the standard one-dimensional cutting stock problem. **European Journal of Operational Research**, vol. 84, n. 3, p. 572-579, 1995.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problem. **Operations Research**, vol. 9, n. 2, p. 849-859, 1961.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problem - Part II. **Operations Research**, vol. 11, p. 863-888, 1963.

GILMORE, P. C.; GOMORY, R. E. Multistage cutting stock problems of two and more dimensions. **Operations Research**, vol. 13, p.94-120, 1965.

GOODRICH, M. T.; TAMASSIA, R. **Algoritmos: Fundamentos, análise e exemplos da Internet.** São Paulo: Bookman , 2002. p. 603.

HAESSLER, R. W. Solving the Two-Stage Cutting Stock Problem. **Omega - The International Journal of Management Science**, vol. 7, n. 2, p. 145-151, 1979.

HOROWITZ, E.; SAHNI, S. Computing partitions with applications to the knapsack problem. **Journal of ACM**, vol. 21, p.277-292, 1974.

HOTO, R. S. V. **Otimização no corte de peças unidimensionais com restrições de agrupamento.** Dissertação de Mestrado em Ciências da Computação e Matemática Computacional . ICMSC-USP, São Carlos, 1996.

HOTO, R. S. V. **O Problema da Mochila Compartimentada aplicado no corte de Bobinas de aço.** Tese de doutorado em Engenharia de Sistemas e Computação. Universidade Federal do Rio de Janeiro, 2001.

HOTO, R.; FENATO, A.; YANASSE, H.; MACULAN, M.; SPOLADOR, F. Uma nova abordagem para o Problema da Mochila Compartimentada. **Simpósio Brasileiro de Pesquisa Operacional**, vol. 38, p. 1760-1771, 2006.

JOHNSON, D. S. **Near-optimal bin packing algorithms.** Technical Report, Project MAC. Massachusetts Institute of Technology, Cambridge, 1973. p. 273.

JOHNSTON, R. E.; KHAN, L. R. Bounds for nested knapsack problems. **European Journal of Operational Research**, vol. 81, p. 154-165, 1995.

JURKIEWICZ, S. **Grafos – Uma Introdução**. Programa de Iniciação Científica da OBMEP 2007, vol. 5. Ministério da Educação, 2007.

KANTOROVICH, L. Mathematical methods of organising and planning production (traduzido de um artigo da Rússia de 1939). **Management Science,** v. 6, n. 4, p. 366-422, 1960.

KOLESAR, P. J. A Branch And Bound Algorithm For The Knapsack Problem. **Management Science,** v. 13, n. 9, p. 723-735, 1967.

LAGOUDAKIS, M. G. **The 0-1 knapsack problem – an introductory survey.** Relatório técnico. 1996. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.47.4378&rep=rep1&type=pdf>. Acesso em: 06 jun. 2015.

LEÃO, A. A. S.; SANTOS, M. O.; HOTO, R. S. V.; ARENALES, M. N. The constrained compartmentalized knapsack problem: mathematical models and solution methods. **European Journal of Operational Research**, vol. 212, p. 455-463, 2011.

LEÃO, A. A. S. **Extensões em problemas de corte: padrões compartimentados e problemas acoplados.** Tese de Doutorado em Ciências da Computação e Matemática Computacional. ICMC-USP, São Carlos, 2013. p. 39-42.

LINS, M. P. E.; CALÔBA, M. G. **Programação Linear: com aplicação em teoria dos jogos e avaliação de desempenho.** Rio de Janeiro: Interciência, 2006.

LORIE, J. H.; SAVAGE, L. J. Three problems in rationing capital. **The Journal of Business,** vol. 28, p. 229-239, 1955.

MALKEVITCH, J. **Bin Packing and Machine Scheduling.** Disponível em: <http://www.ams.org/samplings/feature-column/fcarc-packings1>. Acesso em: 04 jun. 2015.

MARQUES F. P. **O problema da mochila compartimentada.** Dissertação de Mestrado em Ciências da Computação e Matemática Computacional. ICMC-USP, São Carlos, 2000. p. 45-51.

MARQUES F. P.; ARENALES, M. N. The constrained compartmentalised knapsack problem. **Computers & Operations Research**, vol. 34, p. 2109-2129, 2007.

MARTELLO, S.; TOTH, P. An upper bound for the zero-one knapsack problem and a branch and bound algorithm. **European Journal of Operational Resarch**, vol. 1, p. 169-175, 1977.

MARTELLO, S.; TOTH, P. **Knapsack Problems – Algorithms and Computer Implementations.** Chichester: John Wiley & Sons Ltd, reimpressão de 1997.

MORIHARA, I.; IBARAKI, T.; HASEGAWA, T. Bin Packing and multiprocessor scheduling problems with side constraint on job types. **Discrete Applied Mathematics,** vol. 6, p. 173-191, 1983.

NELIβEN, J. How to use structural constraints to comput an upper bound for the pallet loading problem. **European Journal of Operational Research,** vol. 84, p.662-680, 1995. p. 672.

NEMHAUSER, G L.; WOLSEY, L. A. **Integer and Combinatorial Optimization.** New York: John Wiley & Sons, 1988.

PINTO, T. de S. **Uma proposta para resolver o problema de corte de estoque unidimensional com reaproveitamento de sobras por meio de dois objetivos.** Dissertação de Mestrado em Matemática Aplicada e Computacional. Universidade Estadual de Londrina, 2008.

PISINGER, D. **Algorithms for knapsack problems.** Ph.D. thesis, Department of Computer Science. University of Copenhagen, 1995.

POLDI, K. C.; ARENALES, M. N. Heurísticas para o problema de corte de estoque unidimensional inteiro. **Pesquisa Operacional,** v. 26, n.3, Rio de Janeiro, p. 473-492, 2006.

ROSA NETO, E. A. da. **O problema de corte de estoque com aproveitamento de sobras: um estudo de comparação de diferentes modelos matemáticos e heurísticas de resolução.** Dissertação de Mestrado em Matemática Aplicada e Computacional. Universidade Estadual de Londrina, 2015.

SALKIN, H. M.; DE KLUYVER, C. A. The knapsack problem: a survey. **Naval Research Logistics Quarterly,** vol. 22, p. 127-144, 1975. p. 2-3.

STRASZAK, A.; LIBURA, M.; SIKORSKI, J.; WAGNER, D. Computer-assisted Constrained Approval Voting. **Group Decision and Negotiation,** v. 2, p. 275-285, 1993.

VALÉRIO DE CARVALHO, J. M. Exact solution of bin-packing problems using column generation and branch-and-bound. **Annals of Operations Research,** 86**.** Braga, p.629-659, 1999.

VALÉRIO DE CARVALHO, J. M. LP models for bin packing and cutting stock problems. **European Journal of Operation Research,** 141. Braga, p. 253-273, 2002.

VANCE, P. H. Branch-and-price algorithms for the one-dimensional cutting stock problem. **Computational Optimization and Applications,** vol. 9, p. 211-228, 1998.

VAN DE VEL, H.; SHIJIE, S. An application of the Bin-packing technique to job scheduling on uniform processors. **The Fournal of the Operational Research Society,** vol. 42, n. 2. p. 169-172, 1991. p. 169.

VASQUEZ, M.; HAO, J.-K. A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an Earth observation satellite. **Journal of Computational Optimization and Applications,** vol. 20, p. 137-157, 2001.

WOEGINGER, G. J. **The P-versus-np page.** Disponível em: <http://www.win.tue.nl/~gwoegi/P-versus-NP.htm>. Acesso em: 11 mai. 2015.

XAVIER, E. C.; MIYAZAWA, F. K. The class constrained bin packing problem with applications to video-on-demand. **Theoretical Computer Science,** vol. 393, p. 1-25, 2008. p. 1-5.

ZAK, E. J. Row and column generation technique for a multistage cutting stock problem. **Computers & Operations Research,** vol. 29, p. 1143-1156, 2002.

APÊNDICES

APÊNDICE A

O MÉTODO SIMPLEX E A GERAÇÃO DE COLUNAS

Este apêndice tem o objetivo de complementar a teoria em torno da Geração de Colunas abordada na seção 2.1.1 do Capítulo 2 desta dissertação. Entretanto, o Método Simplex é considerado um pré-requisito para a leitura deste trabalho e não será explicada por completa sua fundamentação.

A fim de definir a notação a ser utilizada, considere o seguinte problema com variáveis contínuas:

Minimizar:

$$c^{\mathrm{T}}x \tag{A.1}$$

sujeito a:

$$Ax = d$$

$$x \in \mathbb{R}^{n}_{+}$$

onde c é um vetor $n\times 1$, A uma matriz $m\times n$, d um vetor $m\times 1$ e x o vetor das variáveis de decisão de tamanho $n\times 1$. Suponha que o problema tenha mais variáveis do que restrições, ou seja, $n > m$, tal como no problema de corte de estoque, onde $n >> m$ para n grande.

Seja B a base corrente no Método Simplex e N a matriz formada pelas colunas de A que não estão na base. Seja c_B o vetor $m\times 1$ formado pelas constantes do vetor c relacionadas às variáveis básicas, e c_N o vetor $(n-m)\times 1$ formado pelas constantes do vetor c relacionadas às variáveis não-básicas. Analogamente, x_B será o vetor $m\times 1$ formado pelas variáveis básicas e x_N o vetor das variáveis não-básicas.

Nessas notações, tem-se que:

$$Ax = d \Leftrightarrow Bx_B + Nx_N = d \Leftrightarrow x_B = B^{-1}d - B^{-1}Nx_N\,,$$

lembrando-se que B é não-singular e procura-se uma base $\bar{B}$ em que cx seja minimizado com $x_B = \bar{B}^{-1}d$ e $x_{\bar{N}} = 0_{(n-m)\times 1}$.

No passo simplex pretende-se mudar a base de forma a diminuir o valor de cx, que pode ser calculado como:

$$\begin{aligned} cx &= c_B x_B + c_N x_N \\ &= c_B(B^{-1}d - B^{-1}Nx_N) + c_N x_N \\ &= c_B B^{-1}d - c_B B^{-1}Nx_N + c_N x_N \\ &= c_B B^{-1}d + (c_N - c_B B^{-1}N)x_N. \end{aligned}$$

Assim, tornar a j-ésima variável não-básica (atualmente igualada à zero) em variável básica (valor maior ou igual que zero) não irá diminuir a função-objetivo se o j-ésimo valor do vetor $c_N - c_B B^{-1}N$ por positivo. E se todas as entradas desse vetor forem positivas há garantia de solução ótima por não haver vértices adjacentes com melhoria para a função-objetivo.

Deste, modo, considerando que as entradas de x_N são positivas, para que alguma dessas variáveis torne-se não nula e diminua o valor da função-objetivo, é necessário que a entrada respectiva no vetor $c_N - c_B B^{-1}N$ seja negativa. É comum então escolher a variável cuja entrada neste vetor seja mínima possível, ou seja:

Minimizar:

$$c_j - ya_j \tag{A.2}$$

sujeito a:

a_j é coluna de N,

onde $y = c_B B^{-1}$. Observe que o j-ésimo valor do vetor $c_N - c_B B^{-1}N$ é o j-ésimo valor do vetor c_N subtraído da multiplicação do vetor $c_B B^{-1}$ pela j-ésima coluna de N. Assim, gerar uma coluna é resolver o problema (A.2), a fim de encontrar a coluna de N a entrar na base sem precisar calcular todo o vetor $c_N - c_B B^{-1}N$ e consecutivamente todas as colunas de N, que representam padrões de corte no PCE. Se o valor da função-objetivo em (A.2) for maior ou igual à zero, nenhuma coluna gerada e o problema (A.1) está resolvido.

A garantia de que a resolução do problema (A.2) para o PCE não precisa calcular todas as colunas de N é que ele pode ser relaxado ao seguinte problema de mochila:

Minimizar:

$$c_j - ya_j \tag{A.3}$$

sujeito a:

a_j é coluna de A,

pois as colunas de A que estão em B e não em N possuem custo reduzido $c_j - ya_j$ igual à zero, então o critério garante que não será gerada uma coluna que já está na base. Finalmente, no PCE com estoque homogêneo, o problema (A.3) é um problema de mochila, porque equivale a:

Maximizar:

$$y_1 a_{1j} + y_2 a_{2j} + \ldots + y_n a_{nj} - 1$$

sujeito a:

$$l_1 a_{1j} + l_2 a_{2j} + \ldots + l_n a_{nj} \leq L$$

$$a_{ij} \in \mathbb{Z}_+ \text{ para } i = 1, \ldots, n,$$

onde $y = [y_1, \ldots y_n]$ e $a_j = [a_{1j}, \ldots a_{nj}]$.

Pode-se então considerar o problema gerador de colunas como em (A.4)-(A.6):

Maximizar:

$$y_1 a_1 + y_2 a_2 + \ldots + y_n a_n \tag{A.4}$$

sujeito a:

$$l_1 a_1 + l_2 a_2 + \ldots + l_n a_n \leq L \tag{A.5}$$

$$a_i \in \mathbb{Z}_+ \text{ para } i = 1, \ldots, n, \tag{A.6}$$

e se a função-objetivo encontrar um valor maior do que 1, uma coluna foi encontrada para entrar na base, caso contrário, o método se encerra.

Sistematizando, tem-se o algoritmo apresentado no Quadro 10.

Quadro 10: Método Simplex com geração de colunas

Algoritmo
Entrada: n, l_i, d_i, L.
Saída: B, $c_B x_B$, x_B.

Quadro 11: Método Simplex com geração de colunas (continuação)

Inicialização: crie os padrões de corte homogêneos maximais $b_1,\ldots,b_n$

faça $B=[b_1|\cdots|b_n]$;

e faça $x_B = B^{-1}d$;

Passo 1 (cálculo de y)**:**

Faça $y = c_B B^{-1}$;

Passo 2 (gerar colunas)**:**

Resolva (A.4)-(A.6);

Se o valor da função-objetivo é maior que 1 então

tem-se nova coluna $[a_1,\ldots,a_n]^{\mathrm{T}}$;

caso contrário, **pare!**

Passo 3 (escolhe coluna para sair da base)**:**

Faça $[r_1,\ldots,r_n]^{\mathrm{T}} = B^{-1}[a_1,\ldots,a_n]^{\mathrm{T}}$;

Sendo $x_B=[x_1',\ldots,x_n']^{\mathrm{T}}$, compare os valores a_i/r_i e o índice com menor razão positiva deixa a base.

Passo 4 (atualiza a base)**:**

Substitua em B a coluna que foi escolhida para sair por $[a_1,\ldots,a_n]^{\mathrm{T}}$;

faça $x_B = B^{-1}d$;

e volte ao passo **1.**

Fonte: Chvátal (1983, p. 199).

Considere um simples exemplo apresentado em Chvátal (1983, p. 199): o PCE com barras de estoque de 91 metros e os seguintes itens:

1) 78 de 25,5 metros;
2) 40 de 22,5 metros;
3) 30 de 20 metros;
4) 30 de 15 metros;

Iniciando a resolução, os padrões de corte homogêneos maximais são $[3,0,0,0]^{\mathrm{T}}$, $[0,4,0,0]^{\mathrm{T}}$, $[0,0,4,0]^{\mathrm{T}}$ e $[0,0,0,6]^{\mathrm{T}}$. Logo:

$$B=\begin{bmatrix}3&0&0&0\\0&4&0&0\\0&0&4&0\\0&0&0&6\end{bmatrix} \text{ e } x_B = B^{-1}.\begin{bmatrix}78\\40\\30\\30\end{bmatrix}=\begin{bmatrix}26\\10\\7{,}5\\5\end{bmatrix}.$$

Passo 1)

$$y = c_B B^{-1} = \left[1/3, 1/4, 1/4, 1/6 \right].$$

Passo 2)

Resolvendo-se:

Maximizar:

$$1/3\, a_1 + 1/4\, a_2 + 1/4\, a_3 + 1/6\, a_4$$

sujeito a:

$$25{,}5a_1 + 22{,}5a_2 + 20a_3 + 15a_4 \leq 91$$

$$a_i \in \mathbb{R}_+ \text{ para } i = 1, \ldots, 4,$$

encontra-se um valor maior que 1 para a função-objetivo com $[a_1, a_2, a_3, a_4]^{\mathrm{T}} = [2, 0, 2, 0]^{\mathrm{T}}$.

Passo 3)

$$[r_1, \ldots, r_n]^{\mathrm{T}} = B^{-1}[2, 0, 2, 0]^{\mathrm{T}} = \left[2/3, 0, 1/2, 0 \right]^{\mathrm{T}}$$

Comparando–se $\frac{26}{2/3} = 39$ e $\frac{7{,}5}{1/2} = 15$ segue-se que a terceira coluna deve deixar a base.

Passo 4)

$$B = \begin{bmatrix} 3 & 0 & 2 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix} \text{ e } x_B = B^{-1} d = \begin{bmatrix} 16 \\ 10 \\ 15 \\ 5 \end{bmatrix}.$$

Passo 1)

$$y = c_B B^{-1} = \left[1/3, 1/4, 1/6, 1/6 \right].$$

Passo 2)

Resolvendo-se:

Maximizar:

$$1/3\, a_1 + 1/4\, a_2 + 1/6\, a_3 + 1/6\, a_4$$

sujeito a:

$$25,5a_1 + 22,5a_2 + 20a_3 + 15a_4 \leq 91$$
$$a_i \in \mathbb{Z}_+ \text{ para } i = 1,\ldots,4,$$

encontra-se um valor maior que 1 para a função-objetivo com $[a_1, a_2, a_3, a_4]^T = [2,1,0,1]^T$.

Passo 3)

$$[r_1,\ldots,r_n]^T = B^{-1}[2,1,0,1]^T = \left[\frac{2}{3}, \frac{1}{4}, 0, \frac{1}{6}\right]^T$$

Comparando–se $\frac{16}{\frac{2}{3}} = 24$, $\frac{10}{\frac{1}{4}} = 40$ e $\frac{5}{\frac{1}{6}} = 30$ segue-se que a primeira coluna deve deixar a base.

Passo 4)

$$B = \begin{bmatrix} 2 & 0 & 2 & 0 \\ 1 & 4 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 1 & 0 & 0 & 6 \end{bmatrix} \text{ e } x_B = B^{-1}d = \begin{bmatrix} 24 \\ 4 \\ 15 \\ 1 \end{bmatrix}.$$

Passo 1)

$$y = c_B B^{-1} = \left[\frac{7}{24}, \frac{1}{4}, \frac{5}{24}, \frac{1}{6}\right].$$

Passo 2)

Resolvendo-se:

Maximizar:

$$\frac{7}{24}a_1 + \frac{1}{4}a_2 + \frac{5}{24}a_3 + \frac{1}{6}a_4$$

sujeito a:

$$25,5a_1 + 22,5a_2 + 20a_3 + 15a_4 \leq 91$$
$$a_i \in \mathbb{Z}_+ \text{ para } i = 1,\ldots,4,$$

encontra-se um valor menor que 1 para a função-objetivo, logo já se tem solução ótima com o corte de $24+4+15+1=44$ barras, sendo $[2,1,0,1]^T$, $[0,4,0,0]^T$, $[2,0,2,0]^T$ e $[0,0,0,6]^T$ os padrões de corte que são feitos 24, 4, 15 e 1 vezes, respectivamente.

APÊNDICE B

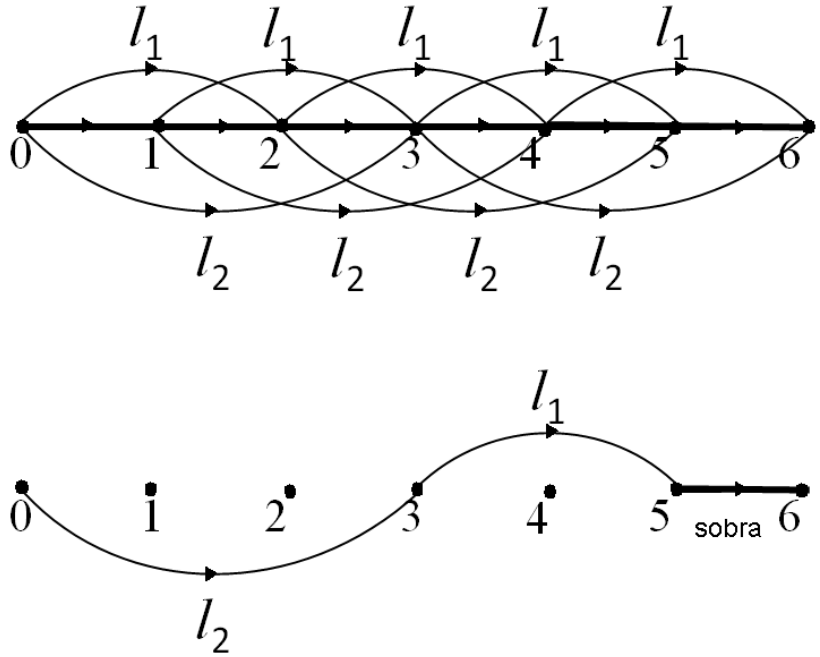
UMA MODELAGEM COM FLUXO DE ARCOS

Neste apêndice é descrito um modelo diferente para o Problema da Mochila Compartimentada Restrito, que utiliza dos conceitos de Teoria de Grafos para ser formulado. Ele foi criado para esta dissertação, e foi estudado e implementado em paralelo com as outras atividades descritas nos capítulos 4 e 5.

B.1 UMA MODELAGEM ARC-FLOW PARA O *BIN-PACKING*

A ideia para uma nova modelagem do PMCR por meio de grafos surgiu de uma modelagem feita por Valério de Carvalho (1999) para o *Bin-packing*. No trabalho de Valério de Carvalho, é considerado um grafo $G(V, A)$ com $V = \{0,1,2,\ldots L\}$ e $A = \{(i, j) : 0 \le i < j \le L$ e $j - i = l_r$ para todo $r = 1,\ldots,n\}$, nas notações desta dissertação. Também são considerados em A arcos adicionais de comprimento unitário, ou seja, $(k, k+1)$, para $k = 1,\ldots L-1$. Assim, um caminho entre o vértice 0 e o vértice L representa uma maneira de se preencher uma mochila, onde cada arco (i, j) de comprimento l_i pertencente ao caminho representa um item

Figura 10: Exemplo de um grafo e um caminho.



Fonte: Valério de Carvalho (1999).

de índice i utilizado, e os arcos unitários são usados para completar o caminho, representando um espaço não utilizado da mochila.

Considere, por exemplo, $L = 6$ e que os tamanhos dos itens são 2 e 3. A Figura 10 exibe o grafo gerado, e uma maneira de completar a mochila, com um item de cada largura, representada no grafo.

Seja x_{ij} a quantidade de vezes que o arco (i, j) é utilizado, e z a quantidade de caminhos de 0 a L utilizados, correspondente à quantidade de mochilas. O *Bin-packing* pode ser assim formulado como:

Minimizar:

$$z \tag{B.1}$$

sujeito a:

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,k)\in A} x_{jk} = \begin{cases} -z, & \text{se } j = 0; \\ 0, & \text{se } j = 1,\ldots,L-1; \\ z, & \text{se } j = L; \end{cases} \tag{B.2}$$

$$x_{ij} \in \{0,1\}, \text{ para } (i, j) \in A, \tag{B.3}$$

onde as restrições em (B.2) são ditas restrições de fluxo. O problema pode ser generalizado de maneira a considerar d_i itens de índice i. Para isto, basta considerar as variáveis inteiras positivas e as restrições:

$$\sum_{(k,k+l_i)\in A} x_{k,k+l_i} \geq d_i \text{ para } i = 1,\ldots,n,$$

de forma a obter-se uma formulação para o Problema de Corte de Estoque Unidimensional com Estoque Homogêneo. Neste sentido, cada caminho no grafo representa um padrão de corte.

B.2 Uma Formulação com Fluxo de Arcos Para Gerar Compartimentos.

Ao invés de se considerar z como uma quantidade de caminhos de 0 a L, pode-se considerar $z = x_{L0}$, ou seja, um arco de *feedback*, e assim o objetivo é minimizar a quantidade de ciclos no grafo. Desta maneira, para criar compartimentos de capacidade limitada no interior da mochila, pode ser considerado um grafo que contenha arcos de *feedback* que vão dos vértices $L^k_{\min}, L^k_{\min}+1,\ldots,L^k_{\max}$ ao vértice 0. Desta forma, fixado $k = 1,\ldots,q$, seja $V = \{0,1,\ldots,L^k_{\min},\ldots L^k_{\max}\}$ e

$$A = \{(j, j+l_i) : 0 \le j \le L^k_{\max} - l_i, \text{ para todo } i = 1,\ldots,n\}$$
$$\cup\ \{(j, j+1) : L^k_{\min} \le j \le L^k_{\max} - 1\}$$
$$\cup\ \{(j, 0) : L^k_{\min} \le j \le L^k_{\max}\}.$$

O Grafo $G(V, A)$ será usado para representar compartimentos viáveis por ciclos no grafo sujeitos a restrições de demanda e facas. Desta forma, o melhor compartimento de capacidade máxima $L^k_{\max}$ é dado pela solução do problema:

Maximizar:

$$\sum_{(i,j)\in A} x_{ij}\bar{u}_{ij} \tag{B.4}$$

sujeito a:

$$\sum_{i:(i,j)\in A} x_{ij} - \sum_{k:(j,k)\in A} x_{jk} = 0, \text{ para } j = 0,1,\ldots L^k_{\max} \tag{B.5}$$

$$\sum_{\substack{(i,j)\in A \\ j=i+l_i}} x_{ij} \le d_i, \text{ para } i \in N \tag{B.6}$$

$$\sum_{\substack{(i,j)\in A \\ j>i \\ x_{ij}>1}} x_{ij} \le F_2 \tag{B.7}$$

$$x_{ij} \in \mathbb{Z}_+, \text{ para } (i,j) \in A, \tag{B.8}$$

onde $\bar{u}_{ij}$ é a utilidade do arco (i, j), dada por $\bar{u}_{ij} = u_e$, sendo $e \in N_k$ o índice do item de largura $l_e = j - i$. A utilidade de um arco de *feedback* ou de comprimento unitário usado para completar um ciclo é nula. As restrições de fluxo em (B.5) garantem os ciclos formados no grafo. As restrições em (B.6) limitam a quantidade utilizada de arcos que representam um mesmo item. A restrição (B.7) limita a quantidade de arcos que representam itens a serem utilizados, onde assume-se que nenhum item possui largura unitária. Em (B.8) é estabelecido o domínio das variáveis.

Note que, para criar compartimentos de capacidade máxima L_{cap}, com $L_{cap} = L^k_{\min},\ldots,L^k_{\max}$, basta excluir do conjunto A os arcos de *feedback* $(j,0)$ tais que $L_{cap} < j \le L^k_{\max}$ e resolver o problema (B.4)-(B-8). Isto foi usado em implementações da Heurística das “w” Capacidades.

Nestes ensaios numéricos, comparou-se o desempenho computacional da heurística estruturada no modelo com fluxo de arcos e da mesma

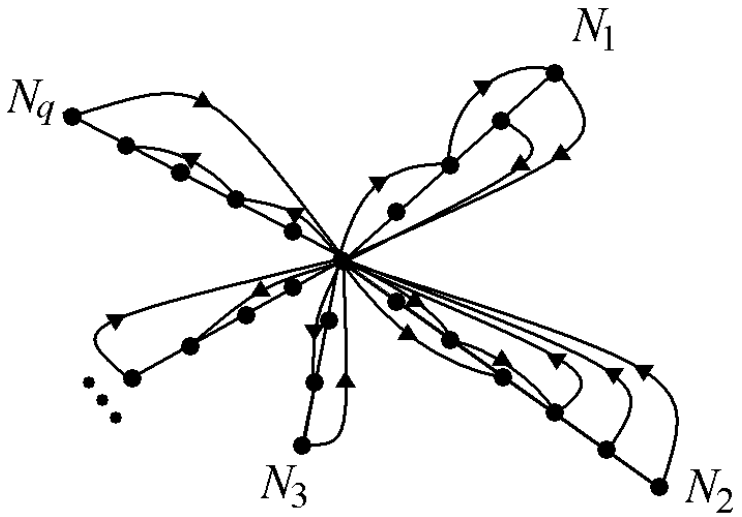
heurística no modelo original do problema, além de implementações mistas que geram compartimentos segundo uma formulação e fazem a compartimentação da mochila segundo outras. Como esperado, usar o modelo com fluxo de arcos gerou soluções equivalentes às obtidas do modelo original com a heurística, porém o tempo computacional foi um pouco maior.

Na próxima seção, será exposto um modelo para o problema que não utiliza a ideia de decomposição. Sua implementação gerou tempos comparáveis aos do Algoritmo da Decomposição Exaustiva, e muito distantes (para exemplares de tamanho grande) aos do modelo linear formulado no Capítulo 5.

B.3 Uma Nova Modelagem Para o Problema da Mochila Compartimentada

Objetivando-se em criar um modelo completo para o PMCR, sem decomposição, considerou-se um grafo que seja a união de q grafos similares ao exposto na seção anterior. A "conecção" entre estes grafos é dada pela origem: o vértice 0. Dele surgem arcos que vão aos vértices l_i $(i \in N_k, k = 1,\ldots,q)$ em cada subgrafo representando uma classe, e chegam arcos que vêm dos pontos $L^k_{\min}, L^k_{\min}+1,\ldots,L^k_{\max}$, para todo $k = 1,\ldots,q$, como ilustrado na Figura 11.

Figura 11: Grafo para o Problema da Mochila Compartimentada.



Fonte: próprio autor.

Sejam A_k os arcos gerados na classe de índice k e $A = \bigcup_{k=1}^{q} A_k$. Considere x^k_{ij} a quantidade de arcos $(i, j) \in A_k$ utilizados, e $\bar{u}^k_{ij}$ a utilidade do arco

$(i, j) \in A_k$, dada por $\bar{u}_{ij}^k = u_e$, onde $e \in N_k$ é o índice do item de largura $l_e = j - i$. Tem-se $\bar{u}_{ij}^k = 0$ se $j = 0$ (arco de *feedback*) ou $i + 1 = j$ (arco para completar o ciclo). A formulação proposta consiste em:

Maximizar:

$$\sum_{k=1}^{q} \sum_{(i,j) \in A_k} x_{ij}^k \bar{u}_{ij}^k \tag{B.9}$$

sujeito a:

$$\sum_{k=1}^{q} \sum_{j:(j,0) \in A_k} x_{j0}^k - \sum_{k=1}^{q} \sum_{k:(0,j) \in A_k} x_{0j}^k = 0 \tag{B.10}$$

$$\sum_{i:(i,j) \in A_k} x_{ij}^k - \sum_{e:(j,e) \in A_k} x_{je}^k = 0, \text{ para } j = 1, \ldots, L_{\max}^k, \; k = 1, \ldots, q \tag{B.11}$$

$$\sum_{k=1}^{q} \sum_{j:(j,0) \in A_k} x_{j0}^k \cdot j \leq L \tag{B.12}$$

$$\sum_{\substack{(i,j) \in A_k \\ j = i + l_e}} x_{ij}^k \leq d_e, \text{ para } e \in N_k, k = 1, \ldots, q \tag{B.13}$$

$$\sum_{k=1}^{q} \sum_{j:(0,j) \in A_k} x_{0j}^k \leq F_1 \tag{B.14}$$

$$x_{ij}^k \in \mathbb{Z}_+, \text{ para } (i, j) \in A_k, \; k = 1, \ldots, q, \tag{B.15}$$

No modelo (B.9)-(B.15), não há restrição de itens por compartimento, logo este modelo não é equivalente ao original do PMCR. No entanto, na prática, podem haver facas suficientes para o corte de sub-bobinas de aço, o que pode tornar o modelo (B.9)-(B.15) viável. A restrição (B.10) garante o fluxo no vértice 0. As restrições em (B.11) estão relacionadas ao fluxo nos outros vértices do grafo. A restrição (B.12) é uma restrição de mochila, que limita a soma das capacidades dos compartimentos construídos. Note que, em (B.12), embora j seja um índice, representa o comprimento do arco $(j, 0) \in A_k$, logo é uma constante (pois o grafo deve estar previamente gerado), garantindo que a restrição seja linear. As restrições em (B.13) limitam a quantidade utilizada de arcos que representam um mesmo item. A restrição (B.14) limita a quantidade de ciclos no grafo. E em (B.15) é estabelecido o domínio das variáveis.

O modelo (B.9)-(B.15) é linear. Logo, para exemplares em que $F_2 \geq \min_{k=1}^{q} \left\{ \left\lfloor L_{\min}^{k} \Big/ \min_{i \in N_k} \{l_i\} \right\rfloor \right\}$, trata-se de um novo modelo linear para o Problema da Mochila Compartimentada Restrito.

APÊNDICE C

PROJETO DE DOUTORADO

Este apêndice contém ideias e esboços de pesquisas em andamento, que constituem um projeto de doutorado em fase de construção.

Algumas ideias sendo consideradas visam o uso de técnicas de programação consagradas na literatura que servem para um melhor aproveitamento de tempo computacional na execução de algoritmos. Por exemplo, pode-se calcular sem muito esforço quais valores entre $L^k_{\min}$ e $L^k_{\max}$ podem gerar compartimentos viáveis. Além disso, há outras técnicas, tais como otimização genética, que podem ser estudadas com o objetivo de verificar sua aplicabilidade no problema.

Durante a escrita da dissertação, criou-se a conjectura de que o problema em foco é fortemente NP-difícil. Assim, buscou-se uma prova de que esta conjectura é verdadeira. As pesquisas a respeito da complexidade do Problema da Mochila Compartimentada estão se desenvolvendo; porém, uma prova de que o Problema da Mochila Compartimentada Restrito é fortemente NP-difícil foi formulada, e é exibida na seção C.1.

C.1 O PROBLEMA DA MOCHILA COMPARTIMENTADA RESTRITO É FORTEMENTE NP-DIFÍCIL

Para demonstrar que o PMCR é fortemente NP-difícil, considerou-se um caso particular do problema, a qual foi provado ser pseudo-polinomialmente redutível ao 3-PARTITION.

O 3-PARTITION é o primeiro problema descoberto ser fortemente NP-difícil (GAREY, JOHNSON, 1975). Sua formulação é dada por: dados $n = 3m$ inteiros positivos $\ell_1, \ldots, \ell_n$, satisfazendo $\sum_{i=1}^{n} \frac{\ell_i}{m} = B$ inteiro e $\frac{B}{4} < \ell_i < \frac{B}{2}$ para $i = 1, \ldots, n$, existe uma partição de $N = \{1, \ldots, n\}$ em m subconjuntos $S_1, \ldots, S_m$ tal que $\sum_{i \in S_j} \ell_i = B$ para todo $j = 1, \ldots, m$?

Note que, para que a resposta seja “sim”, cada conjunto S_j, com $j = 1, \ldots, m$, deve conter exatamente três elementos de N, justificando o nome do problema.

A ideia é mostrar que um caso particular do Problema da Mochila Compartimentada Restrito (PMCR), denominado Problema da Mochila Compartimentada Particular (PMCP), é fortemente NP-difícil, usando que cada exemplar do 3-PARTITION pode ser pseudo-polinomialmente reduzido a um exemplar do PMCP. O Problema da Mochila Compartimentada Particular assume $q=1$, $N=N_1=\{1,\ldots,n\}$, $d_i=1$ para todo $i\in N$, $L/L^1_{\max}\in\mathbb{Z}_+$, $L^1_{\min}=\min_{i\in N}\ell_i$, $F_1=L/L^1_{\max}$ e $F_2=\left\lfloor L^1_{\max}/\min_{i\in N}\ell_i\right\rfloor$. Assim, o PMCP consistem em:

Maximizar:

$$\sum_{i\in N}\sum_{j\in V}a_{ij}u_iy_j \tag{C.1}$$

sujeito a:

$$\sum_{i\in N}\sum_{j\in V}a_{ij}\ell_iy_j\le L \tag{C.2}$$

$$\delta_j\le\sum_{i\in N}a_{ij}\ell_i\le\delta_jL^1_{\max},\ \text{para } j\in V=V_1 \tag{C.3}$$

$$\sum_{j\in V}a_{ij}y_j\le d_i=1,\ \text{para } i\in N \tag{C.4}$$

$$\sum_{j\in V}y_j\le F_1=\frac{L}{L^1_{\max}} \tag{C.5}$$

$$\sum_{i\in N}a_{ij}\le F_2=\left\lfloor\frac{L^1_{\max}}{\min_{i\in N}\ell_i}\right\rfloor,\ \text{para } j\in V \tag{C.6}$$

$$\delta_j\in\{0,1\}\ \text{e}\ a_{ij},y_j\ge0\ \text{e inteiros, para}\ i\in N,\ j\in V. \tag{C.7}$$

Observe que a restrição (C.2) está inativa, pois:

$$\begin{aligned}\sum_{i\in N}\sum_{j\in V}a_{ij}\ell_iy_j &\overset{\text{(C.3)}}{\le} L^1_{\max}\sum_{j\in V}y_j\\ &\overset{\text{(C.5)}}{\le} L^1_{\max}F_1\\ &= L.\end{aligned}$$

A restrição (C.6) também está inativa, pois, se $\delta_j=0$, então $\sum_{i\in N}a_{ij}=0$, e caso contrário:

$$\begin{aligned}\sum_{i\in N} a_{ij} &= \frac{\min_{i\in N}\ell_i}{\min_{i\in N}\ell_i}\sum_{i\in N} a_{ij}\\ &= \left\lfloor \frac{\min_{i\in N}\ell_i}{\min_{i\in N}\ell_i}\sum_{i\in N} a_{ij} \right\rfloor\\ &\le \left\lfloor \frac{\sum_{i\in N} a_{ij}\ell_i}{\min_{i\in N}\ell_i} \right\rfloor\\ &\overset{(C.3)}{\le} \left\lfloor \frac{L^1_{\max}}{\min_{i\in N}\ell_i} \right\rfloor\\ &= F_2.\end{aligned}$$

Então, escrevendo o PMCP como um problema de decisão, obtém-se o R(PMCP), com a seguinte formulação: Dados $L^1_{\max}$, $u_1,\ldots,u_n$, $\ell_1,\ldots,\ell_n$, F_1, r e a inteiros, existem $a_{ij}\in\mathbb{Z}_+$, $y_j\in\mathbb{Z}_+$ e $\delta_j\in\{0,1\}$, para $i\in N=\{1,\ldots,n\}$ e $j\in V=\{1,\ldots,r\}$, tais que $\delta_j \le \sum_{i\in N} a_{ij}\ell_i \le \delta_j L^1_{\max}$, para todo $j\in V$, $\sum_{j\in V} a_{ij}y_j \le 1$ para todo $i\in N$, $\sum_{j\in V} y_j \le F_1$ e $\sum_{i\in N}\sum_{j\in V} a_{ij}u_i y_j \ge a$?

Pretende-se reduzir pseudo-polinomialmente o 3-PARTITION ao R(PMCP). Porém, há uma diferença na linguagem dos dois problemas. O 3-PARTITION trabalha com a criação de conjuntos, enquanto que o R(PMCP) trabalha com variáveis inteiras.

Para escrever o R(PMCP) numa linguagem compatível com o 3-PARTITION, considere o R(PMCP-MOD), formulado por: Dados $L^1_{\max}$, $u_1,\ldots,u_n$, $\ell_1,\ldots,\ell_n$, F_1 e a inteiros, existem $S_1,\ldots,S_{F_1}\subset N$ disjuntos tais que $\sum_{i\in S_j}\ell_i \le L^1_{\max}$, para todo $j=1,\ldots,F_1$, e $\sum_{j=1}^{F_1}\sum_{i\in S_j} u_i \ge a$?

Lema C1. *O R(PMCP-MOD) é pseudo-polinomialmente redutível ao R(PMCP).*

Prova: Tome $r=F_1$. Se a resposta do R(PMCP) é “sim”, então, para cada $j\in V=\{1,\ldots,F_1\}$, defina:

$$S_j=\varnothing \Leftrightarrow \delta_j=0 \text{ ou } y_j=0.$$

Também, para cada $i\in N$ e $j\in V$ tal que $\delta_j=1$, defina:

$$i \in S_j \Leftrightarrow a_{ij} = 1.$$

Tem-se que, para cada $i \in N$, vale $\sum_{j \in V} a_{ij} y_j \leq 1$. Logo, $\sum_{j \in V} a_{ij} y_j = 0$ ou $\sum_{j \in V} a_{ij} y_j = 1$. Então $a_{ij} y_j = 0$, para todo $j \in V$, ou existe $j_0 \in V$ tal que $a_{ij_0} y_{j_0} = 1$ e $a_{ij} y_j = 0$ para $j \in V - \{j_0\}$. No primeiro caso, $i \notin \bigcup_{j \in V} S_j$. No segundo caso, $i \in S_{j_0}$ e $i \notin \bigcup_{j \in V - \{j_0\}} S_j$. Logo:

$$S_j \cap S_k = \varnothing \text{ se } j \neq k, \text{ para } j,k \in V. \tag{C.8}$$

Agora, seja $j \in V$. Tem-se $\delta_j \leq \sum_{i \in N} a_{ij} \ell_i \leq \delta_j L^1_{\max}$. Caso $\delta_j = 0$, então $S_j = \varnothing$ e $\sum_{i \in S_j} \ell_i = 0 \leq L^1_{\max}$. E caso $\delta_j = 1$, tem-se:

$$\begin{aligned} \sum_{i \in S_j} \ell_i &= \sum_{\substack{i \in N \\ a_{ij}=1}} a_{ij} \ell_i \\ &\leq \sum_{i \in N} a_{ij} \ell_i \\ &\leq L^1_{\max}. \end{aligned}$$

Em ambos os casos, vale:

$$\sum_{i \in S_j} \ell_i \leq L^1_{\max}, \text{ para todo } j = 1,\ldots,F_1. \tag{C.9}$$

Por fim, note que:

$$\sum_{j=1}^{F_1} \sum_{i \in S_j} u_i = \sum_{j \in V} \sum_{i \in S_j} a_{ij} u_i = \sum_{i \in N} \sum_{j \in V} a_{ij} u_i y_j \geq a. \tag{C.10}$$

Então, por (C.8), (C.9) e (C.10), a resposta do R(PMCP-MOD) é "sim".

Reciprocamente, suponha que a resposta do R(PMCP-MOD) seja "sim". Tome, para cada $j \in V = \{1,\ldots,F_1\}$:

$$\delta_j = y_j = 0 \text{ se } S_j = \varnothing;$$

$$\delta_j = y_j = 1 \text{ se } S_j \neq \varnothing.$$

Também defina, para cada $i \in N$ e $j \in V$:

$$a_{ij} = 1 \text{ se } i \in S_j;$$

$$a_{ij} = 0 \text{ se } i \notin S_j.$$

Seja $j \in V$. Tem-se $\sum_{i \in S_j} \ell_i \leq L^1_{\max}$ para todo $j \in V$. Caso $S_j = \varnothing$, $a_{ij} = \delta_j = 0$ para todo $i \in N$. Então, $0 = \delta_j \leq \sum_{i \in N} a_{ij} \ell_i \leq \delta_j L^1_{\max} = 0$. Caso $S_j \neq \varnothing$, $\delta_j = 1$ e existe $j_0 \in V$ tal que $a_{ij_0} = 1$, logo $\delta_j = 1 \leq \sum_{i \in N} a_{ij} \ell_i$. Também $\sum_{i \in N} a_{ij} \ell_i = \sum_{i \in S_j} \ell_i \leq L^1_{\max}$. Portanto:

$$\delta_j \leq \sum_{i \in N} a_{ij} \ell_i \leq \delta_j L^1_{\max}, \text{ para todo } j \in V. \quad \text{(C.11)}$$

Seja $i \in N$. Se $i \notin \bigcup_{j \in V} S_j$, então $a_{ij} = 0$ para todo $j \in V$, logo $\sum_{j \in V} a_{ij} y_j = 0$. Se $i \in S_{j_0}$ e $i \notin \bigcup_{j \in V - \{j_0\}} S_j$, para algum $j_0 \in V$, então $a_{ij_0} y_{j_0} = 1$ e $a_{ij} y_j = 0$ para $j \in V - \{j_0\}$. Assim:

$$\sum_{j \in V} a_{ij} y_j \leq 1, \text{ para todo } i \in N. \quad \text{(C.12)}$$

Como $V = \{1, \ldots, F_1\}$, obviamente:

$$\sum_{j \in V} y_j \leq F_1. \quad \text{(C.13)}$$

Por fim, para cada $j \in V$, tem-se $a_{ij} = y_j = 1$ se $i \in S_j$, e $a_{ij} = 0$ se $i \in V - S_j$, logo:

$$\sum_{i \in N} \sum_{j \in V} a_{ij} u_i y_j = \sum_{j \in V} \sum_{i \in S_j} u_i \geq a. \quad \text{(C.14)}$$

Portanto, por (C.11), (C.12), (C.13) e (C.14), a resposta do R(PMCP) é “sim”.

□

Lema C2. *O 3-PARTITION é pseudo-polinomialmente redutível ao R(PMCP-MOD).*

Prova: De fato, dado um exemplar do 3-PARTITION, composto por $n = 3m$ inteiros positivos $\ell_1, \ldots, \ell_n$, satisfazendo $\sum_{i=1}^{n} \frac{\ell_i}{m} = B$ inteiro e $\frac{B}{4} < \ell_i < \frac{B}{2}$ para $i \in N = \{1, \ldots, n\}$, tome $u_i = 1$ para todo $i \in N$, $L^1_{\max} = B$, $F_1 = m$ e $a = n$.

Suponha que a resposta do R(PMCP-MOD) seja “sim”. Então, existem $S_1,\ldots,S_m \subset N$ disjuntos tais que $\sum_{i\in S_j} \ell_i \le B$, para todo $j=1,\ldots,m$, e $\sum_{j\in V}\sum_{i\in S_j} 1 \ge n$, ou seja, $\bigcup_{j=1}^{m} S_j = N$, logo $S_1,\ldots,S_m$ particionam N.

Como $\sum_{i=1}^{n} \frac{\ell_i}{m} = B$, tem-se:

$$mB = \sum_{i\in N} \ell_i = \sum_{j=1}^{m}\sum_{i\in S_j} \ell_i \le \sum_{j=1}^{m} B = mB.$$

Assim, $\sum_{j=1}^{m}\sum_{i\in S_j} \ell_i = \sum_{j=1}^{m} B$, e como $\sum_{i\in S_j} \ell_i \le B$ para todo $j=1,\ldots,m$, segue que:

$$\sum_{i\in S_j} \ell_i = B, \text{ para todo } j=1,\ldots,m.$$

Logo, a resposta do 3-PARTITION é “sim”.

Reciprocamente, se a resposta do 3-PARTITION é “sim”, então $S_1,\ldots,S_m$ particionam N e:

$$\sum_{i\in S_j} \ell_i = B = L^1_{\max}, \text{ para todo } j=1,\ldots,m.$$

Além disso,

$$\sum_{j=1}^{F_1}\sum_{i\in S_j} u_i = \sum_{j=1}^{m}\sum_{i\in S_j} 1 = \sum_{i\in N} 1 = n = a.$$

Portanto, a resposta do R(PMCP-MOD) é “sim”.

□

Teorema C1. O Problema da Mochila Compartimentada Restrito é fortemente NP-difícil.

Prova: Pelo Lema C2, o 3-PARTITION é pseudo-polinomialmente redutível ao R(PMCP-MOD), logo o R(PMCP-MOD) é fortemente NP-difícil. E, pelo Lema C1, o R(PMCP-MOD) é pseudo-polinomialmente redutível ao R(PMCP), logo o R(PMCP) é também fortemente NP-difícil.

É óbvio que o R(PMCP) é pseudo-polinomialmente redutível ao PMCP, uma vez que a solução do problema de otimização responderá ao problema de decisão, com a mesma entrada. Sendo assim, o PMCP é fortemente NP-difícil.

Mas o PMCP é um caso particular do PMCR, e, portanto, o Problema da Mochila Compartimentada Restrito é fortemente NP-difícil.

□

ANEXOS

ANEXO A

ARTIGO “MOCHILAS COMPARTIMENTADAS SÃO LINEARES: UMA DEMONSTRAÇÃO”

Neste anexo será exibida uma versão em português do artigo “Mochilas Compartimentadas São Lineares: Uma Demonstração”, cuja tradução para a língua inglesa pretende-se submeter a uma revista internacional. Nesta versão de anexo, o artigo está formatado com as normas da revista Mathematics of Operations Research, do instituto Informs®.

MATHEMATICS OF OPERATIONS RESEARCH
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0364-765X | EISSN 1526-5471 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000

Mochilas Compartimentadas São Lineares: Uma Demonstração

Inarejos Filho, O.
Universidade Estadual de Londrina, osvaldoinarejos@hotmail.com

Hoto, R. S. V.
Universidade Estadual de Londrina, hoto@uel.br

O Problema da Mochila Compartimentada surgiu de problemas de corte em duas fases, especialmente no corte de bobinas de aço. Em sua formulação original, trata-se de um problema de otimização inteira não-linear, e até então este problema tem sido resolvido por meio de heurísticas de decomposição. Neste artigo temos por objetivo mostrar que o Problema da Mochila Compartimentada Restrita é um problema de Otimização Linear. Para isto, nós consideramos o modelo não-linear original e propomos um modelo linear para o problema, exibindo uma demonstração de que os dois são equivalentes.

Key words: Otimização Linear; Otimização Discreta; Problema da Mochila Compartimentada.
MSC2000 subject classification:
OR/MS subject classification: Primary: ; secondary:
History: Received...

1. Introdução. Problemas de mochila são geralmente simples de serem entendidos e de grande aplicabilidade, mas nem sempre de fácil resolução. De maneira geral, estes problemas envolvem a escolha de itens a serem colocados em uma ou mais mochilas, buscando-se maximizar uma função de utilidade [16, 12].

Possivelmente, a primeira menção ao Problema da Mochila foi feita por Dantzig [4], que apresentou o seguinte exemplo: uma pessoa planeja acampar e decide não carregar mais que 70 lb de diferentes itens, como roupas de cama, latas de comida, etc. Cada item possui um peso e uma utilidade. O objetivo da pessoa é levar consigo itens numa mochila que deverá ser a "mais útil possível" (soma linear das utilidades dos itens selecionados). Obviamente, se o conjunto de itens não excede o peso de 70 lb, então a escolha é trivial.

REMARK 1. Podemos interpretar o problema como o de decidir, no conjunto das partes dos itens, aquele que respeita a capacidade da mochila e fornece a maior soma linear de utilidades, assim, sendo n a cardinalidade do conjunto de itens, o conjunto das partes terá cardinalidade 2^n, deixando claro que qualquer tentativa de resolução exaustiva do problema é inviável.

Um dos exemplos mais conhecidos da aplicação de um problema de mochila é o Método Simplex com geração de colunas, para resolver o Problema de Corte de Estoque Unidimensional [7].

Existem variações do clássico Problema da Mochila, algumas podem ser vistas em [16] e [12]. Neste artigo, abordaremos uma variação em que é necessário construir compartimentos no iterior de uma mochila. A compartimentação de uma mochila [10] ocorre quando os itens devem ser

agrupados no seu interior segundo afinidades. Por exemplo, quando não se deseja misturar remédios, ferramentas e comidas no interior de uma mochila, requerendo a construção de compartimentos distintos, para que neles sejam alocados itens de mesma natureza. Esta particularidade induz no conjunto de itens uma partição matemática (classes de itens com mesma afinidade). A motivação desta versão de mochila, cujo modelo matemático original [9] é não-linear, surgiu da necessidade de se modelar "padrões de corte" em duas fases [8, 5, 1, 17, 2].

Possivelmente, o problema mais semelhante ao Problema da Mochila Compartimentada encontrado na literatura é o Nasted Knapsack Problem [11], aplicado na produção de cabos de fibra ótica.

Hoto [9] apresentou, além da formulação concisa do Problema da Mochila Compartimentada, um algoritmo de compartimentação exata (para exemplares pequenos) e alguns procedimentos heurísticos. Também descreveu o Problema da Mochila Compartimentada 0-1, para o qual calculou limitantes e determinou critérios de poda, propondo assim, um algoritmo branch-and-bound para resolvê-lo.

Marques e Arenales [14] apresentaram três novas heurísticas, com destaque para a Heurística dos "z" Melhores Compartimentos.

Os mesmos autores [15] também apresentaram a Heurística das "w" Capacidades, e relataram ter obtido melhor performace que a Heurística dos "z" Melhores Compartimentos. No mesmo trabalho, encontra-se uma relaxação linear para o Problema da Mochila Compartimentada, cuja solução traz um limitante superior. Em [13] é usado o limitante superior dado por esta relaxação para verificar a otimalidade de solução por algumas heurísticas.

Outro limitante superior abordado em [13] utiliza geração de colunas para resolver o problema de compartimentar mochilas, onde a cada iteração simplex alguns subproblemas (um para cada classe) são calculados. Em termos de tempo computacional, esta heurística foi um pouco mais rápida que a relaxação do problema feita por Marques e Arenales em [15], porém o limitante superior foi um pouco maior na maioria dos exemplares testados.

Também é apresentada em [13] uma heurística híbrida que explora as ideias das heurísticas dos "z" melhores compartimentos e das "w" capacidades.

Em [3] são apresentados testes baseados na ideia de eliminar a não-linearidade do modelo original da compartimentação de uma mochila, e em [13] estas ideias são usadas para comparar resultados de heurísticas. Foram estas simulações a motivação da nossa pesquisa apresentada neste artigo, pois, nenhum dos dois trabalhos foram capazes de provar que uma Mochila Compartimentada é, na realidade, um problema linear.

Na seção 2 apresentaremos o Problema da Mochila Compartimentada tal como tem sido abordado. Na seção 3 apresentaremos a formalização do modelo linear. Na seção 4 iremos expor ensaios numéricos que ilustram a motivação em investigar, teoricamente, a equivalência dos modelos não-linear e linear. Na seção 5 exibiremos uma demonstração de que o Problema da Mochila Compartimentada é linear.

A fim de facilitar a abstração da formulação matemática do problema, usaremos a aplicação no corte de bobinas de aço sujeitas a laminação a frio como ilustração.

2. Aplicação e formulação matemática. O Problema da Mochila Compartimenta surge como subproblema gerador de colunas do problema de corte de bobinas de aço (gerador de padrões de corte), no qual bobinas de aço são cortadas e laminadas para reduzir a espessura do aço, de acordo com a finalidade dos itens, como para confecção de tubos para bicicletas, automóveis, construção civil, etc, de modo que a primeira fase de corte produz as sub-bobinas a serem laminadas e a segunda fase produz fitas (itens), conforme Figura 1. As fitas serão soldadas em forma de tubos (Figura 2), assim, fitas de largura e laminação específica darão origem a tubos previamente demandados.

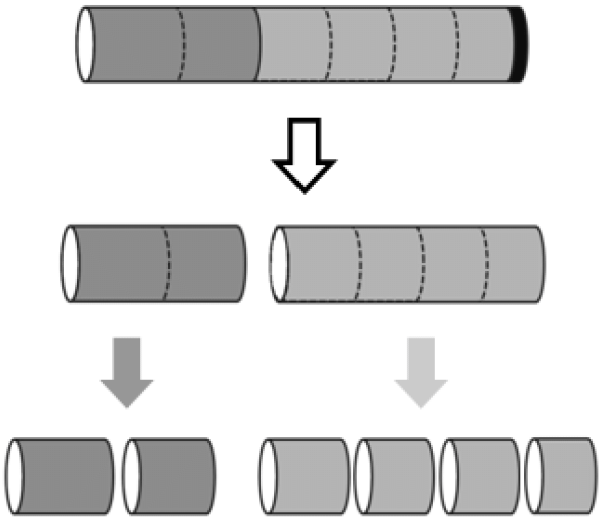


FIGURE 1. Corte em duas fases: bobina, sub-bobinas e fitas.

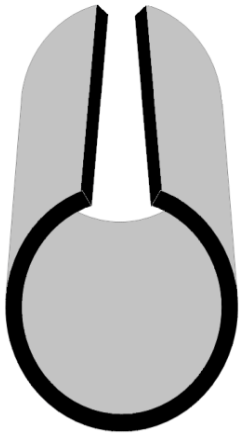


FIGURE 2. Tubo a ser produzido com fitas de aço.

Matematicamente, admita que o conjunto de índices dos itens, denotado por N, esteja particionado em q classes N_k, $k = 1, \ldots, q$. Desta forma, $N = N_1 \cup \ldots \cup N_q$ e $N_r \cap N_s = \emptyset$ se $r \neq s$, $r,s = 1, \ldots, q$. No problema de corte de bobinas de aço pode-se entender uma classe como o conjunto de índices dos itens que irão sofrer a mesma laminação entre a primeira e a segunda fase de corte.

Ainda, uma sub-bobina deve ter sua largura limitada entre um mínimo e um máximo, visto que na segunda etapa de corte a máquina possui limitações técnicas, bem como a máquina responsável pela laminação. Em razão disto, deve-se construir no interior da mochila compartimentos de capacidades limitadas entre um valor mínimo L^k_{min} e um máximo L^k_{max}, onde os índices dos itens de um compartimento devem pertencer a uma mesma classe N_k.

REMARK 2. A ideia original para formalizar o modelo matemático consiste em: ***imaginar todos os compartimentos (sub-bobinas) viáveis construídos para, em seguida, selecionar os que deverão fazer parte da compartimentação da mochila.*** Esta metodologia é idêntica à usada na geração de colunas [7].

Com este enfoque, denote por V_k o conjunto de índices de todos os compartimentos viáveis com itens indexados em N_k, sendo $V = V_1 \cup \ldots \cup V_q$ o conjunto de índices de todos os compartimentos, onde $V_r \cap V_s = \emptyset$ se $r \neq s$, $r,s = 1, \ldots, q$.

Seja L a capacidade da mochila (largura de cada bobina), u_i a utilidade (multiplicadores simplex vindos do problema mestre) e ℓ_i o peso do item de índice i (largura da fita). Considere como variáveis de decisão: a_{ij} que controlará a quantidade de itens de índice i no compartimento de índice j, y_j que definirá a quantidade de compartimentos de índice j utilizados, $\delta_j = 1$ se o compartimento de índice j foi construído, e $\delta_j = 0$ no caso contrário. A formulação matemática apresentada em [9] consiste em:

$$\texttt{maximizar:} \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i a_{ij}\right) y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} u_i a_{ij}\right) y_j \tag{1}$$

$$\texttt{sujeito a:} \sum_{j\in V_1}\left(\sum_{i\in N_1} \ell_i a_{ij}\right) y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} \ell_i a_{ij}\right) y_j \leq L \tag{2}$$

$$\delta_j L^k_{min} \leq \sum_{i\in N_k} \ell_i a_{ij} \leq \delta_j L^k_{max},\ j \in V_k, k = 1,\ldots,q \tag{3}$$

$$a_{ij}, y_j \geq 0 \text{ e inteiros}, \delta_j \in \{0,1\}, i \in N, j \in V. \tag{4}$$

A restrição (2) assegura que a soma das larguras de cada compartimento construído não ultrapassa a capacidade da mochila. As restrições em (3) garantem que cada compartimento construído tenha capacidade limitada entre o mínimo e o máximo aceito para compartimentos com itens indexados na classe N_k. As restrições em (4) determinam o domínio.

EXAMPLE 1. Considere apenas 2 classes e 5 itens, sendo 3 itens na primeira classe e 2 itens na segunda classe. Neste caso, $q = 2$, $N = \{1,2,3,4,5\}$, $N_1 = \{1,2,3\}$ e $N_2 = \{4,5\}$. Digamos que os compartimentos da primeira classe devem ter de 8 a 10 uc (unidades de comprimento) e os compartimentos da segunda classe devem ter de 7 a 12 uc, então $L^1_{min} = 8$ uc, $L^2_{min} = 7$ uc, $L^1_{max} = 10$ uc, $L^2_{max} = 12$ uc. Considere também as larguras dos itens, como $\ell_1 = 2$ uc, $\ell_2 = 3$ uc, $\ell_3 = 5$ uc, $\ell_4 = 5$ uc e $\ell_5 = 6$ uc. Sendo assim, um compartimento de índice j representado por $a_j = (a_{1j}, a_{2j}, a_{3j})$ pertence à V_1 se $8 \leq 2a_{1j} + 3a_{2j} + 5a_{3j} \leq 10$, onde a_{ij} é um inteiro positivo que representa a quantidade de itens de índice i no compartimento de índice j. Analogamente, um compartimento de índice j representado por $a_j = (a_{4j}, a_{5j})$ pertence à V_2 se $7 \leq 5a_{4j} + 6a_{5j} \leq 12$.

Assim, os compartimentos possíveis de serem construídos são: $a_1 = (5,0,0)$, $a_2 = (4,0,0)$, $a_3 = (3,1,0)$, $a_4 = (2,2,0)$, $a_5 = (2,0,1)$, $a_6 = (1,2,0)$, $a_7 = (1,1,1)$, $a_8 = (0,3,0)$, $a_9 = (0,1,1)$, $a_{10} = (0,0,2)$, $a_{11} = (2,0)$, $a_{12} = (1,1)$ e $a_{13} = (0,2)$. Logo, podemos definir $V_1 = \{1,2,3,4,5,6,7,8,9,10\}$ e $V_2 = \{11,12,13\}$.

Note que, embora os itens de índice 3 e 4 possuam a mesma largura, são itens diferentes, uma vez que pertencem a classes distintas, podendo sofrer diferentes processos de laminação ou qualquer processo entre a primeira e a segunda fase de corte.

REMARK 3. Não há uma única maneira de definir o conjunto $V = V_1 \cup \ldots \cup V_q$. A única exigência é que contenha os índices de todos os compartimentos viáveis, mas a ordem na qual a indexação é feita pode ser diferente sem alterar a solução ótima do problema. Além disso, o conjunto V pode conter mais índices do que o necessário, relacionados a compartimentos idênticos (compostos pelas mesmas quantidades dos mesmos itens).

Vejamos, no Exemplo 1, poderíamos definir o conjunto V_2 como $V_2 = \{11,12,13,14\}$, com $a_{11} = (1,1)$, $a_{12} = (2,0)$, $a_{13} = (0,2)$, e $a_{14} = (2,0)$.

Observe que se todos os compartimentos estão previamente gerados, o modelo (1)-(4) se reduz ao clássico Problema da Mochila, porém, em problemas práticos é inviável gerar todos os compartimentos tal como fizemos no Exemplo 1 (veja a Observação 1 na Introdução).

2.1. Restrições adicionais. No modelo (1)-(4), podem surgir novas restrições devido ao contexto prático, por exemplo, no corte de estoque em duas fases, há uma demanda a ser atendida para cada item, de acordo com pedidos de clientes. Considere d_i a demanda do item de índice i. Assim, para garantir que não sejam produzidos itens em excesso, deve-se impor a restrição (5):

$$\sum_{k=1}^{q} \sum_{j \in V_k} a_{ij} y_j \leq d_i, \ i \in N = N_1 \cup \ldots \cup N_q, \tag{5}$$

onde a soma das quantidades de um determinado item em cada compartimento que o produz é limitada pela demanda desse item.

No problema de corte de bobinas de aço, conforme ilustra a Figura 1, a quantidade de facas que podem ser ajustadas na máquina é limitada, tanto no primeiro como no segundo processo de corte. No primeiro processo de corte, a quantidade de compartimentos (sub-bobinas a serem laminadas) deve ser limitada por uma constante F_1, já no segundo processo de corte, cada compartimento (sub-bobina laminada) dará origem a itens (fitas de aço), portanto, o número de itens em cada compartimento deve ser limitado por uma constante F_2. Neste caso, devem valer as restrições (6) e (7):

$$\sum_{k=1}^{q} \sum_{j \in V_k} y_j \leq F_1 \tag{6}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2, \ j \in V = V_1 \cup \ldots \cup V_q, k = 1, \ldots, q. \tag{7}$$

Com a adição das restrições (5), (6) e (7) ao problema (1)-(4), obtém-se o Problema da Mochila Compartimentada Restrito:

$$\texttt{maximizar:} \ \sum_{j \in V_1} \left(\sum_{i \in N_1} u_i a_{ij} \right) y_j + \cdots + \sum_{j \in V_q} \left(\sum_{i \in N_q} u_i a_{ij} \right) y_j \tag{8}$$

$$\texttt{sujeito a:} \ \sum_{j \in V_1} \left(\sum_{i \in N_1} \ell_i a_{ij} \right) y_j + \cdots + \sum_{j \in V_q} \left(\sum_{i \in N_q} \ell_i a_{ij} \right) y_j \leq L \tag{9}$$

$$\delta_j L^k_{min} \leq \sum_{i \in N_k} \ell_i a_{ij} \leq \delta_j L^k_{max}, \ j \in V_k, k = 1, \ldots, q \tag{10}$$

$$\sum_{k=1}^{q} \sum_{j \in V_k} a_{ij} y_j \leq d_i, \ i \in N = N_1 \cup \ldots \cup N_q \tag{11}$$

$$\sum_{k=1}^{q} \sum_{j \in V_k} y_j \leq F_1 \tag{12}$$

$$\sum_{i \in N_k} a_{ij} \leq F_2, \ j \in V = V_1 \cup \ldots \cup V_q, k = 1, \ldots, q \tag{13}$$

$$\delta_j \in \{0,1\} \ \text{ e } a_{ij}, y_j \geq 0 \text{ e inteiros}, i \in N, j \in V. \tag{14}$$

O modelo (8)-(14) é mais fiel às condições do problema de corte de bobinas de aço sujeitas à laminação a frio, sendo portanto, empregado na geração dos padrões de corte. Trata-se da compartimentação de uma mochila com restrições adicionais, portanto, uma **Mochila Não-linear**.

Como já indicamos, existem heurísticas para este problema relatadas na literatura, assim, na próxima seção apresentaremos um modelo matemático no qual eliminamos a variável y_j.

3. Uma nova modelagem da compartimentação de mochilas. Agora, ao invés de abordarmos o problema pela Observação 2 (seção 2), observamos que, para cada classe N_k, $k = 1, \ldots, q$, há uma limitação na quantidade de compartimentos a serem construídos, que dependerá do limitante F_1 e do fator $\lfloor L/L^k_{min} \rfloor$. Assim, seja $p_k = \min\left\{F_1, \lfloor L/L^k_{min} \rfloor\right\}$ o número máximo de compartimentos viáveis construídos com itens indexados na classe N_k.

Dentro deste novo contexto, seja a_{ijk} a quantidade de itens de índice $i \in N_k$ no compartimento de índice $j = 1, \ldots, p_k$.

Muito bem, o modelo que compartimentará uma mochila, com as restrições adicionais já mencionadas na subseção 2.1 é:

$$\texttt{maximizar:} \sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} u_i a_{ijk} \tag{15}$$

$$\texttt{sujeito a:} \sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} \ell_i a_{ijk} \leq L \tag{16}$$

$$\delta_{jk} L^k_{min} \leq \sum_{i \in N_k} \ell_i a_{ijk} \leq \delta_{jk} L^k_{max}, \ j = 1, \ldots, p_k \tag{17}$$
$$\text{e } k = 1, \ldots, q$$

$$\sum_{j=1}^{p_k} a_{ijk} \leq d_i, \ i \in N_k, k = 1, \ldots, q \tag{18}$$

$$\sum_{k=1}^{q} \sum_{j=1}^{p_k} \delta_{jk} \leq F_1 \tag{19}$$

$$\sum_{i \in N_k} a_{ijk} \leq F_2, \ j = 1, \ldots, p_k, k = 1, \ldots, q \tag{20}$$

$$\delta_{jk} \in \{0, 1\} \text{ e } a_{ijk} \geq 0 \text{ e inteiros}, i \in N_k, \tag{21}$$
$$j = 1, \ldots, p_k \text{ e } k = 1, \ldots, q.$$

A função objetivo apresentada em (15) representa a soma das utilidades dos itens escolhidos em cada compartimento. A restrição (16) garante que a soma das larguras dos itens escolhidos é limitada pela capacidade da mochila. As restrições em (17) delimitam a largura dos compartimentos não-nulos e anulam os coeficientes dos compartimentos que não serão utilizados, sendo $\delta_{jk} = 1$ se o compartimento de índice j da classe N_k for não-nulo, e $\delta_{jk} = 0$ no caso contrário. As restrições em (18) limitam a quantidade de cada item por sua respectiva demanda. Diferente de (11), a soma das quantidades dos itens em (18) ocorre apenas nos compartimentos da classe correspondente, pois neste novo modelo os elementos a_{ijk} só são definidos para $i \in N_k$. A restrição (19) limita a quantidade de compartimentos não-nulos, e em (20) são restringidas as quantidades de itens em cada compartimento. As restrições em (21) estabelecem o domínio.

Example 2. Considere os dados do Exemplo 1, além de $L = 22$ e $F_1 = 3$. Temos $p_1 = \min\{3, \lfloor 22/8 \rfloor\} = 2$ e $p_2 = \min\{3, \lfloor 22/7 \rfloor\} = 3$. Significa que no máximo 2 compartimentos da classe N_1 e no máximo 3 compartimentos da classe N_2 serão inseridos na mochila. Teoricamente, serão construídos (na nova abordagem) 5 compartimentos, mas alguns desses podem ser nulos ($a_{ijk} = 0$ para todo $i \in N_k$, onde $j = 1, \ldots, p_k$).

Agora, os índices dos compartimentos construídos com itens indexados em N_1 pertencem à $\{1, 2\}$ e os índices dos compartimentos com itens indexados em N_2 estão em $\{1, 2, 3\}$, a indexação dos compartimentos não é mais global como tratamos no Exemplo 1.

Além disso, os compartimentos não podem mais ser todos previamente construídos e indexados, o que é irrelevante, pois, como já mencionamos, na prática é inviável criar todos os possíveis compartimentos.

4. Ensaios numéricos: a motivação. A fim de avaliar a qualidade das soluções provenientes do modelo (15)-(21), implementamos tal modelo e comparamos as soluções com um algoritmo exato, que utiliza uma decomposição exaustiva, proveniente da Observação 2 (seção 2), e fornece, portanto, a solução ótima de cada exemplar. Assim, o Algoritmo de Decomposição Exaustiva gera, por meio de um método de ramificação, todos os compartimentos viáveis e, então, um problema de programação inteira é resolvido para definir a compartimentação da mochila.

Este procedimento é, obviamente, inviável para uso prático, conforme Observação 1 (seção 1), mas, permite obtermos a solução ótima de exemplares com a finalidade de testes teóricos. Com base em testes iniciais, percebemos que 40 itens por classe é uma quantia máxima adequada para a geração dos exemplares, uma vez que alguns exemplares de 45 itens por classe e todos os de 50 itens por classe testados obtiveram falha em sua resolução, por gerar muitos compartimentos (milhões) e exaurir a memória da máquina durante a compartimentação. Enquanto isto, trabalhar com muitas classes de poucos itens não exigiu tanta memória. Sendo assim, organizamos sete categorias de exemplares: 5/10, 10/5, 10/10, 10/30, 30/10, 10/40 e 40/10, onde a categoria q/n é formada por exemplares de q classes e n itens em cada classe.

Nós criamos um gerador aleatório com referência em [6], utilizando valores realísticos do problema de corte de bobinas de aço (onde as larguras são medidas em milímetros), definimos $L = 2200$, $F_1 = 12$, $F_2 = 8$ e, para toda classe N_k, $L^k_{min} = 375$ e $L^k_{max} = 750$. Além disso, as larguras dos itens possuem valores aleatórios entre 55 e 350, as utilidades entre 0 e 1 com três casas de arredondamento e convertidas a valores inteiros e, por fim, a demanda total corresponde a três vezes o número total de itens, gerando as demandas dos itens distribuídas aleatoriamente conforme sugerido em [6]. Desta maneira, geramos 200 exemplares para cada categoria, um total de 1400 exemplares.

Nós realizamos as implementações, utilizando o solver FICO® Xpress Optimizer, num Intel® Core™ i7 de 2,67 GHz com 12 GB de RAM. Os resultados podem ser observados na Tabela 1, onde constam, para cada algoritmo, a **média** e o **desvio padrão** (representado por σ) aproximados dos valores das funções-objetivo (Obj) e dos tempos computacionais (T), em segundos, dos exemplares de cada categoria.

TABLE 1. Resultados dos ensaios numéricos.

	Algoritmo de Decomposição Exaustiva				Modelo Linear			
Categorias	$\overline{Obj}$	$\sigma(Obj)$	$\overline{T}$	$\sigma(T)$	$\overline{Obj}$	$\sigma(Obj)$	$\overline{T}$	$\sigma(T)$
5/10	16,175	2,483	0,602	1,471	16,175	2,483	0,321	1,323
10/5	16,104	2,121	0,094	1,138	16,104	2,121	0,213	0,159
10/10	19,651	2,307	0,655	0,289	19,651	2,307	0,339	0,264
10/30	25,489	1,945	91,803	34,768	25,489	1,945	1,153	6,135
30/10	23,974	1,887	1,695	0,362	23,974	1,887	0,541	0,386
10/40	26,903	1,675	834,623	564,056	26,903	1,675	0,875	1,508
40/10	25,210	1,680	2,485	0,469	25,210	1,680	0,680	0,307

Em cada exemplar testado, os valores ótimos da função-objetivo obtidos com o Algoritmo de Decomposição Exaustiva e com a implementação do modelo linear são **idênticos**, o que nos levou a acreditar que isto sempre acontece, fato matematicamente comprovado na próxima seção.

5. Mochilas Compartimentadas são lineares. Nesta seção exibiremos uma prova de que os dois modelos de Mochila Compartimentada Restrita formalizados nas seções anteriores possuem a mesma solução ótima, ou seja, mesmo valor máximo para a função objetivo. Dividimos a demonstração em três partes, por questão de organização, começando por um lema e, então, dividindo a prova em duas partes, que serão melhor explicadas ao final da subseção 5.1.

5.1. Uma relação entre os modelos e a construção da notação a ser utilizada. Antes de enunciar o principal resultado deste artigo, comecemos pelo Lema 1, que estabelece uma relação entre os modelos (8)-(14) e (15)-(21).

LEMMA 1. *Dada uma solução*

$$y = \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i a_{ij}\right) y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} u_i a_{ij}\right) y_j$$

do modelo (8)-(14), existem valores β_{ijk}, com $i \in N_k$, $j = 1,\ldots,p_k$ e $k = 1,\ldots,q$ tais que:

$$\sum_{j=1}^{p_k} \beta_{ijk} = \sum_{j\in V_k} a_{ij} y_j, \text{ para todo } i \in N_k, k = 1,\ldots,q.$$

Analogamente, dada uma solução

$$z = \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} u_i b_{ijk}$$

e $\eta_{jk} \in \{0,1\}$ $(j = 1,\ldots,p_k,\ k = 1,\ldots,q)$ do modelo (15)-(21), existem valores α_{ij} e z_j, com $i \in N$ e $j \in V$ tais que:

$$\sum_{j=1}^{p_k} b_{ijk} = \sum_{j\in V_k} \alpha_{ij} z_j, \text{ para todo } i \in N_k, k = 1,\ldots,q.$$

Proof. Com efeito, para cada $k = 1,\ldots,q$, vamos mostrar que $\sum_{j\in V_k} y_j \le p_k$, onde $p_k = \min\left\{F_1, \left\lfloor L/L^k_{min}\right\rfloor\right\}$, uma vez que $\sum_{k=1}^{q}\sum_{j\in V_k} y_j \le F_1$, e como $\sum_{i\in N_k} \ell_i a_{ij} \ge L^k_{min}$ se o compartimento de índice j é não-nulo, temos:

$$\begin{aligned} L &\ge \sum_{j\in V_1}\left(\sum_{i\in N_1} \ell_i a_{ij}\right) y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} \ell_i a_{ij}\right) y_j \\ &\ge \sum_{j\in V_k}\left(\sum_{i\in N_k} \ell_i a_{ij}\right) y_j \\ &\ge L^k_{min} \sum_{j\in V_k} y_j, \end{aligned}$$

e por isso $\sum_{j\in V_k} y_j \le \left\lfloor L/L^k_{min}\right\rfloor$, pois $\sum_{j\in V_k} y_j$ é um número inteiro.

Agora, para cada item de índice $i \in N$ temos que:

$$\begin{aligned} \sum_{j\in V_k} a_{ij} y_j &= a_{is_1} y_{s_1} + a_{is_2} y_{s_2} + \cdots + a_{is_r} y_{s_r} \\ &= \underbrace{a_{is_1} + \cdots + a_{is_1}}_{y_{s_1} \text{ vezes}} + \underbrace{a_{is_2} + \cdots + a_{is_2}}_{y_{s_2} \text{ vezes}} + \cdots + \underbrace{a_{is_r} + \cdots + a_{is_r}}_{y_{s_r} \text{ vezes}}, \end{aligned}$$

onde assumimos que $V_k = \{s_1,\ldots,s_r\}$.

Uma vez que $y_{s_1} + y_{s_2} + \cdots + y_{s_r} \le p_k$, podemos definir

$$\left(\beta_{i1k}, \beta_{i2k}, \ldots, \beta_{ip_kk}\right)$$

como

$$\left(\underbrace{a_{is_1},\ldots,a_{is_1}}_{y_{s_1} \text{ vezes}}, \underbrace{a_{is_2},\ldots,a_{is_2}}_{y_{s_2} \text{ vezes}}, \ldots, \underbrace{a_{is_r},\ldots,a_{is_r}}_{y_{s_r} \text{ vezes}}, \underbrace{0,\ldots,0}_{p_k - \sum_{j\in V_k} y_j \text{ vezes}}\right).$$

REMARK 4. Observe que fixado $j = 1, \ldots, p_k$, existe $j' \in V_k$ tal que $\beta_{ijk} = a_{ij'}$, ou seja, os elementos β_{ijk} podem também ser entendidos como os coeficientes do compartimento de índice j da abordagem linear, que são os mesmos coeficientes do compartimento de índice j' da abordagem não-linear. Excessão à essa observação faz-se apenas no caso em que β_{ijk} foi definido com valor nulo para todo $i \in N_k$.

Além disso, como desejado, a construção feita garante que $\sum_{j=1}^{p_k} \beta_{ijk} = \sum_{j \in V_k} a_{ij} y_j$ para cada item de índice $i \in N_k$, $k = 1, \ldots, q$.

Agora, considere os conjuntos:

$$\begin{aligned} W_1 &= \{\, 1, 2, \ldots, p_1\} \\ W_2 &= \{\, p_1 + 1, \ldots, p_1 + p_2\} \\ &\vdots \\ W_k &= \{\, p_1 + \cdots + p_{k-1} + 1, \ldots, p_1 + \cdots + p_k\} \\ &\vdots \\ W_q &= \{\, p_1 + \cdots + p_{q-1} + 1, \ldots, p_1 + \cdots + p_q\} \\ &\text{e} \\ W &= W_1 \cup \ldots \cup W_q. \end{aligned}$$

Dado $j \in V$, se $j \notin W$, defina $z_j = 0$. Se $j \in W_k$, para algum $k = 1, \ldots, q$, então defina $z_j = \eta_{j-(p_1+\cdots+p_{k-1}),k}$. Da mesma forma, defina

$$\alpha_{ij} = \begin{cases} 0, \text{ se } j \notin W_k \\ b_{i,j-(p_1+\cdots+p_{k-1}),k}, \text{ se } j \in W_k \end{cases}, \forall i \in N_k.$$

Como todos os compartimentos com índice em W_k são formados por itens indexados em N_k, em vista do que está na Observação 3 (seção 2), podemos admitir que $W_k \subset V_k$ para todo $k = 1, \ldots, q$, uma vez que há uma maneira de definir os índices de V_k desta forma, sem alterar a solução ótima do problema.

Temos que $\sum_{j \in W_k} \alpha_{ij} z_j = \sum_{j \in V_k} \alpha_{ij} z_j$, pois se $j \in V_k - W_k$, tanto α_{ij} quanto z_j foram definidos com valores nulos. Também temos que $\sum_{j=1}^{p_k} b_{ijk} = \sum_{j \in W_k} \alpha_{ij} z_j$, pois se $z_j = 0$ é porque $j \notin W$ ou $\eta_{j-(p_1+\cdots+p_{k-1}),k} = 0$ o que implica por (17) que $b_{i,j-(p_1+\cdots+p_{k-1}),k} = 0$. Assim, podemos estabelecer uma relação "um a um" de $b_{ij'k}$ $(j' = 1, \ldots, p_k)$ com $\alpha_{ij} z_j$ $(j = p_1 + \cdots + p_{k-1} + 1, \ldots, p_1 + \cdots + p_{k-1} + p_k)$ e concluir que $\sum_{j=1}^{p_k} b_{ijk} = \sum_{j \in W_k} \alpha_{ij} z_j$.

Portanto, por transitividade, vale $\sum_{j=1}^{p_k} b_{ijk} = \sum_{j \in V_k} \alpha_{ij} z_j$, para todo $i \in N_k$, $k = 1, \ldots, q$. □

THEOREM 1. *Os modelos (8)-(14) e (15)-(21) são equivalentes.*

Proof. Sejam y o valor ótimo da função objetivo do modelo não-linear (8)-(14) e z o valor ótimo da função objetivo do modelo linear (15)-(21). Sendo assim, para todo $j \in V$ e para todo $i \in N$, existem a_{ij}, y_j e δ_j tais que:

$$y = \sum_{j \in V_1} \left(\sum_{i \in N_1} u_i a_{ij} \right) y_j + \cdots + \sum_{j \in V_q} \left(\sum_{i \in N_q} u_i a_{ij} \right) y_j,$$

com os valores a_{ij}, y_j e δ_j satisfazendo as condições (9)-(14), e y sendo o maior valor com esta propriedade. Do mesmo modo, para todo $k = 1, \ldots, q$ e para todos $i \in N_k, j = 1, \ldots, p_k$ existem b_{ijk} e η_{jk} tais que:

$$z = \sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} u_i b_{ijk},$$

com os valores b_{ijk} e η_{jk} satisfazendo as condições (16)-(21), e sendo z o maior valor com esta propriedade.

Para provar que $y = z$ provaremos, separadamente, que $y \leq z$ e que $y \geq z$. Para mostrar que $y \leq z$ provaremos que existem, para cada $i \in N_k$, $j = 1, \ldots, p_k$ e $k = 1, \ldots, q$, valores β_{ijk} e ζ_{jk} na região viável do modelo linear tais que $\sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} u_i \beta_{ijk} = y$, e como z é o máximo com essa propriedade concluiremos que $y \leq z$.

Prosseguiremos de forma análoga para provar que $y \geq z$, construindo valores α_{ij}, z_j e ξ_j na região viável do modelo não-linear tais que:

$$\sum_{j \in V_1} \left(\sum_{i \in N_1} u_i \alpha_{ij} \right) z_j + \cdots + \sum_{j \in V_q} \left(\sum_{i \in N_q} u_i \alpha_{ij} \right) z_j = z,$$

e como y é o máximo com essa propriedade concluiremos que $y \geq z$. Desta forma, a demonstração se dividirá em duas partes: *Parte 1* e *Parte 2*.

Antes de começá-la, apresentamos no quadro a seguir a notação que será usada, a fim de prosseguir com clareza.

Modelo:	Não-linear	Linear
Valores definidos:	y, a_{ij}, y_j e δ_j	z, b_{ijk} e η_{jk}
Valores construídos:	α_{ij}, z_j e ξ_j	β_{ijk} e ζ_{jk}

Na *Parte 1* definiremos β_{ijk} e ζ_{jk} com base em a_{ij}, y_j e δ_j, e na *Parte 2* definiremos α_{ij}, z_j e ξ_j com base em b_{ijk} e η_{jk}. Uma observação importante é que nos referiremos às equações dos modelos (8)-(21) às quais as variáveis construídas devam satisfazer e fica subentendido que não queremos verificar estritamente como está a equação referida nas seções anteriores, mas adaptada às variáveis que estamos trabalhando. Por exemplo, dizer que os valores β_{ijk} satisfazem (18) significará dizer que $\sum_{j=1}^{p_k} \beta_{ijk} \leq d_i$, para todos $i \in N_k$ e $k = 1, \ldots, q$.

5.2. Parte 1 da demonstração. Pelo Lema 1, podemos definir, para cada $k = 1, \ldots, q$, os valores β_{ijk} de forma que valha a igualdade:

$$\sum_{j=1}^{p_k} \beta_{ijk} = \sum_{j \in V_k} a_{ij} y_j, \forall i \in N_k, k = 1, \ldots, q, \tag{22}$$

e que satisfaçam também a Observação 4.

Imediatamente de (22) segue que:

$$\begin{aligned}
y &= \sum_{j \in V_1} \left(\sum_{i \in N_1} u_i a_{ij} \right) y_j + \cdots + \sum_{j \in V_q} \left(\sum_{i \in N_q} u_i a_{ij} \right) y_j \\
&= \sum_{i \in N_1} \left(\sum_{j \in V_1} y_j a_{ij} \right) u_i + \cdots + \sum_{i \in N_q} \left(\sum_{j \in V_q} y_j a_{ij} \right) u_i \\
&= \sum_{i \in N_1} \left(\sum_{j=1}^{p_1} \beta_{ij1} \right) u_i + \cdots + \sum_{i \in N_q} \left(\sum_{j=1}^{p_q} \beta_{ijq} \right) u_i \\
&= \sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} \beta_{ijk} u_i.
\end{aligned} \tag{$\bigstar$}$$

Logo $y = \sum_{k=1}^{q} \sum_{i \in N_k} \sum_{j=1}^{p_k} u_i \beta_{ijk}$, como desejado. Agora provaremos que os valores definidos estão na região viável do modelo linear. A restrição (18) segue de (22), pois se $\sum_{j \in V_k} a_{ij} y_j \leq$

$\sum_{k=1}^{q}\sum_{j\in V_k} a_{ij}y_j \le d_i$, então $\sum_{j=1}^{p_k}\beta_{ijk} \le d_i$ para todo $i \in N_k$, e todo $k = 1,\dots,q$. E, com um cálculo análogo ao realizado em (★), podemos mostrar que $\sum_{j\in V_1}\left(\sum_{i\in N_1}\ell_i a_{ij}\right)y_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q}\ell_i a_{ij}\right)y_j = \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k}\ell_i\beta_{ijk}$, donde (16) é satisfeito.

Pela Observação 4 (Lema 1), dado $j = 1,\dots,p_k$, se $\beta_{ijk} \neq 0$ para algum $i \in N_k$, então existe $j' \in V_k$ tal que $\beta_{ijk} = a_{ij'}$ para todo $i \in N_k$. Caso $\beta_{ijk} = 0$ para todo $i \in N_k$, tome $\zeta_{jk} = 0$, caso contrário tome $\zeta_{jk} = \delta_{j'}$. Em ambos os casos $\delta_{j'}L^k_{min} \le \sum_{i\in N_k}\ell_i a_{ij'} \le \delta_{j'}L^k_{max}$ implica que $\zeta_{jk}L^k_{min} \le \sum_{i\in N_k}\ell_i\beta_{ijk} \le \zeta_{jk}L^k_{max}$, logo a restrição (17) também é satisfeita.

Ainda da Observação 4, temos que $\sum_{i\in N_k} a_{ij'} = \sum_{i\in N_k}\beta_{ijk}$, e como vale $\sum_{i\in N_k} a_{ij'} \le F_2$, temos $\sum_{i\in N_k}\beta_{ijk} \le F_2$. Como j é geral ($j = 1,\dots,p_k;\ k = 1,\dots,q$), podemos concluir que a restrição (20) também é satisfeita.

Agora, para provar (19), observe que nomeando $\sum_{j\in V_k} y_j$ de R_k, temos:

$$\begin{aligned}\sum_{j=1}^{p_k}\zeta_{jk} &= \sum_{j=1}^{R_k}\zeta_{jk} + \underbrace{\sum_{j=R+1}^{p_k}\zeta_{jk}}_{=0}\\ &= \sum_{j=1}^{R_k}\zeta_{jk}\\ &\le R_k\\ &= \sum_{j\in V_k} y_j, \text{ para cada } k = 1,\dots,q.\end{aligned}$$

Sendo assim, $\sum_{k=1}^{q}\sum_{j\in V_k} y_j \le F_1$ implica $\sum_{k=1}^{q}\sum_{j=1}^{p_k}\zeta_{jk} \le F_1$, o que garante a restrição (19). Assim, concluímos que os elementos β_{ijk} e ζ_{jk} definidos estão na região viável do modelo linear, logo $y = \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} u_i\beta_{ijk} \le z$.

5.3. Parte 2 da demonstração. Agora, vamos construir valores α_{ij}, z_j e ξ_j, e provar que tais valores satisfazem as restrições (9)-(14), além de que $z = \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i\alpha_{ij}\right)z_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} u_i\alpha_{ij}\right)z_j$.

Sendo $z = \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} u_i b_{ijk}$ uma solução ótima do modelo linear, com b_{ijk} e η_{jk} ($i \in N_k;\ j = 1,\dots,p_k;\ k = 1,\dots,q$) satisfazendo as condições (16)-(21), segue do Lema 1 que existem α_{ij} e z_j tais que:

$$\sum_{j=1}^{p_k} b_{ijk} = \sum_{j\in V_k}\alpha_{ij}z_j, \forall i \in N_k, k = 1,\dots,q. \tag{23}$$

Analogamente ao cálculo em (★), que contou com o uso da equação (22), podemos usar (23) para concluir que:

$$z = \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k} u_i b_{ijk} \overset{(23)}{=} \sum_{j\in V_1}\left(\sum_{i\in N_1} u_i\alpha_{ij}\right)z_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q} u_i\alpha_{ij}\right)z_j$$

e

$$\sum_{j\in V_1}\left(\sum_{i\in N_1}\ell_i\alpha_{ij}\right)z_j + \cdots + \sum_{j\in V_q}\left(\sum_{i\in N_q}\ell_i\alpha_{ij}\right)z_j \overset{(23)}{=} \sum_{k=1}^{q}\sum_{i\in N_k}\sum_{j=1}^{p_k}\ell_i b_{ijk} \overset{(16)}{\le} L.$$

Logo, a restrição (9) do modelo não-linear é satisfeita para os elementos α_{ij} e z_j definidos.

Defina $\xi_j = z_j$ para todo $j \in V$, com z_j tal como foi definido no Lema 1.

Para provar a restrição (10), seja $j \in V_k$, para algum $k = 1, \ldots, q$. Se $j \notin W_k$, temos $\xi_j = \alpha_{ij} = 0$ para todo $i \in N_k$ e não há o que fazer. Caso contrário, $j \in W_k$, e assim $\xi_j = \eta_{j-(p_1+\cdots+p_{k-1}),k}$ e $\alpha_{ij} = b_{i,j-(p_1+\cdots+p_{k-1}),k}$. Como $j - (p_1 + \cdots + p_{k-1}) \in \{1, \ldots, p_k\}$, temos por (17):

$$\eta_{j-(p_1+\cdots+p_{k-1}),k} L^k_{min} \leq \sum_{i \in N_k} \ell_i b_{i,j-(p_1+\cdots+p_{k-1}),k} \leq \eta_{j-(p_1+\cdots+p_{k-1}),k} L^k_{max},$$

logo $\xi_j L^k_{min} \leq \sum_{i \in N_k} \ell_i \alpha_{ij} \leq \xi_j L^k_{max}$, o que prova a condição (10).

A condição (11) é consequência de (18) e de (23), pois, se $i \in N_m$ e $j \in V_n$, com $m \neq n$, $m, n = 1, \ldots, q$, foi definido $\alpha_{ij} = 0$. Logo, dado $i \in N_k$, temos:

$$\sum_{m=1}^{q} \sum_{j \in V_m} \alpha_{ij} z_j = \sum_{j \in V_k} \alpha_{ij} z_j \overset{(23)}{=} \sum_{j=1}^{p_k} b_{ijk} \overset{(18)}{\leq} d_i.$$

Agora, como $z_j = \begin{cases} 0, \text{ se } j \notin W \\ \eta_{j-(p_1+\cdots+p_{k-1}),k}, \text{ se } j \in W_k \end{cases}$, então

$$\sum_{j \in V_k} z_j = \sum_{j \in W_k} z_j = \sum_{j=1}^{p_k} \eta_{jk}, \forall k = 1, \ldots, q,$$

logo $\sum_{k=1}^{q} \sum_{j=1}^{p_k} \eta_{jk} \leq F_1$ se, e somente se, $\sum_{k=1}^{q} \sum_{j \in V_k} z_j \leq F_1$, o que prova a restrição (12).

Por fim, seja $j \in V$, se $j \notin W$ então $\alpha_{ij} = 0$ para todo $i \in N$ e segue trivialmente a restrição (13). Caso contrário, $j \in W_k$ para algum $k = 1, \ldots, q$, e assim $\alpha_{ij} = b_{i,j-(p_1+\cdots+p_{k-1}),k}$ para todo $i \in N_k$. Como $\sum_{i \in N_k} b_{i,j-(p_1+\cdots+p_{k-1}),k} \leq F_2$, pois $j - (p_1 + \cdots + p_{k-1}) \in \{1, \ldots, p_k\}$, concluímos que $\sum_{i \in N_k} \alpha_{ij} \leq F_2$, o que garante a restrição (13).

Assim, os elementos α_{ij}, z_j e ξ_j definidos estão na região viável do modelo não-linear, o que implica:

$$z = \sum_{j \in V_1} \left(\sum_{i \in N_1} u_i \alpha_{ij} \right) z_j + \cdots + \sum_{j \in V_q} \left(\sum_{i \in N_q} u_i \alpha_{ij} \right) z_j \leq y.$$

Mas na *Parte 1* demonstramos que $y \leq z$, logo $y = z$, e portanto os modelos são equivalentes, pois possuem o mesmo valor máximo para a função objetivo. □

Considerações finais. A prova da linearidade do Problema da Mochila Compartimentada é um avanço teórico e justifica novos estudos do problema. Quanto à expansividade da demonstração, tentamos efetuar simplificações, desde que não dificultasse seu entendimento, mas podem haver maneiras mais imediatas de exibir a comprovação desejada. Simplificações são sempre bem-vindas, embora o essencial seja o resultado validado.

Nós entendemos que o modelo matemático de um problema não é único, de fato, outros modelos podem surgir.

Optamos neste trabalho por considerar uma variação na abordagem trabalhada em [3], por entendermos que os elementos a_{ijk} não fazem sentido prático para $i \notin N_k$, o que tornaria necessário igualar a zero esses elementos. No modelo apresentado, essas variáveis não são definidas e não é necessário igualá-las a zero, além de que faz parte da descrição do nosso modelo apenas os p_k compartimentos construídos, sem a necessidade de usar os conjuntos de índices de compartimentos V_k aparentes no modelo original do problema e no modelo linear apresentado em [3].

Uma possibilidade de estudo posterior é verificar se as heurísticas de trabalhos anteriores podem ser adaptadas ao novo modelo. Também pode ser interessante verificar se outros modelos lineares podem melhorar o desempenho da resolução, assim como verificar métodos heurísticos que possam tirar proveito das boas soluções provindas da programação linear sem, no entanto, causar demasiada oscilação no tempo computacional. No momento, nossas pesquisas estão concentradas no cálculo de limitantes superiores para auxiliarem na construção de um algortimo branch-and-bound eficiente.

Acknowledgments. Agradecemos à CAPES e ao CNPq pelo apoio financeiro, também à FICO® por nos ceder o Xpress Optimizer.

References

[1] Carvalho, J. M. V. and Rodrigues, A. J. G. (1994). A computer based interactive approach to a two-stage cutting-stock problem. *INFOR*, 32(4):243–252.

[2] Correia, M. H., Oliveira, J. F., and Ferreira, J. (2004). Reel and sheet cutting at a paper mill. *Computers & Operations Research*, 31(8):1223–1243.

[3] Cruz, E. P. d. (2010). Uma abordagem heurística linear para mochilas compartimentadas restritas. Master's thesis, Universidade Estadual de Londrina.

[4] Dantzig, G. B. (1957). Discrete-variable extremum problems. *Operations Research*, 5(2):266–288.

[5] Ferreira, J. S., Neves, M. A., and Fonseca e Castro, P. (1990). Cutting and packing a two-phase roll cutting problem. *European Journal of Operational Research*, 44(2):185–196.

[6] Gau, T. and Wäscher, G. (1995). Cutgen1: A problem generator for the standard one-dimensional cutting stock problem. *European Journal of Operational Research*, 84(3):572–579.

[7] Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859.

[8] Haessler, R. W. (1979). Solving the two-stage cutting stock problem. *Omega - The International Jornal of Management Science*, 7(2):145–151.

[9] Hoto, R. S. V. (2001). *O Problema da Mochila Compartimentada aplicado no corte de bobinas de aço.* PhD thesis, Universidade Federal do Rio de Janeiro.

[10] Hoto, R. S. V. and Arenales, M. N. (1996). Um problema de corte unidimensional com restrições de agrupamento e aplicações industriais. In *Anais do I ON PCE.*

[11] Johnston, R. E. and Khan, L. R. (1995). Bounds for nested knapsack problems. *European Journal of Operational Research*, 81(1):154–165.

[12] Kellerer, H., Pferschy, U., and Pisinger, D. (2013). *Knapsack Problems.* Springer Berlin Heidelberg.

[13] Leão, A. A. S., Santos, M. O., Hoto, R. S. V., and Arenales, M. N. (2011). The constrained compartmentalized knapsack problem: mathematical models and solution methods. *European Journal of Operational Research*, 212(3):455–463.

[14] Marques, F. P. and Arenales, M. N. (2002). O problema da mochila compartimentada e aplicaçẽs. *Pesquisa Operacional*, 22(3):285–304.

[15] Marques, F. P. and Arenales, M. N. (2007). The constrained compartmentalised knapsack problem. *Computers & Operations Research*, 34(7):2109–2129.

[16] Martello, S. and Toth, P. (1990). *Knapsack problems: algorithms and computer implementations.* J. Wiley & Sons.

[17] Zak, E. J. (2002). Row and column generation technique for a multistage cutting stock problem. *Computers & Operations Research*, 29(9):1143–1156.