Universidade Estadual Paulista - UNESP
Faculdade de Engenharia de Bauru - FEB
Departamento de Engenharia de Produção - DEP

Arthur Medeiros Figueiredo Barreto

O uso de K soluções para o problema de corte de estoque com sobras aproveitáveis

Bauru - São Paulo

2019

Universidade Estadual Paulista - UNESP
Faculdade de Engenharia de Bauru - FEB
Departamento de Engenharia de Produção - DEP

Arthur Medeiros Figueiredo Barreto

O uso de K soluções para o problema de corte de estoque com sobras aproveitáveis

Dissertação apresentada como requisito à obtenção do grau de Mestre em Engenharia de Produção, Programa de Pós-Graduação em Engenharia de Produção da Faculdade de Engenharia de Bauru - FEB - da Universidade Estadual Paulista - UNESP - Campus Bauru

Orientadora: Profa. Dra. Adriana Cristina Cherri
Co-Orientador: Dr. Luiz Henrique Cherri

Bauru - São Paulo
2019

B273u

Barreto, Arthur Medeiros Figueiredo
O uso de K soluções para o problema de corte de estoque com sobras aproveitáveis / Arthur Medeiros Figueiredo Barreto. -- Bauru, 2019
64 p. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Faculdade de Engenharia, Bauru
Orientadora: Adriana Cristina Cherri
Coorientador: Luiz Henrique Cherri

1. Pesquisa operacional. 2. Otimização matemática. 3. Algoritmos. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Engenharia, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Bauru

ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE ARTHUR MEDEIROS FIGUEIREDO BARRETO, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO, DA FACULDADE DE ENGENHARIA - CÂMPUS DE BAURU.

Aos 28 dias do mês de junho do ano de 2019, às 14:00 horas, no(a) Anfiteatro da Seção Técnica de Pós-graduação da FEB, reuniu-se a Comissão Examinadora da Defesa Pública, composta pelos seguintes membros: Profª. Drª. ADRIANA CRISTINA CHERRI NICOLA - Orientador(a) do(a) Departamento de Matemática / Faculdade de Ciências de Bauru - UNESP, Prof. Dr. PEDRO AUGUSTO MUNARI JUNIOR do(a) Departamento de Engenharia da Produção / Universidade Federal de São Carlos, Profª. Drª. EDILAINE MARTINS SOLER do(a) Departamento de Matemática / Faculdade de Ciências de Bauru - UNESP, sob a presidência do primeiro, a fim de proceder a arguição pública da DISSERTAÇÃO DE MESTRADO de ARTHUR MEDEIROS FIGUEIREDO BARRETO, intitulada **O USO DE K SOLUÇÕES PARA O PROBLEMA DE CORTE DE ESTOQUE COM SOBRAS APROVEITÁVEIS**. Após a exposição, o discente foi arguido oralmente pelos membros da Comissão Examinadora, tendo recebido o conceito final: aprovado. Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelos membros da Comissão Examinadora.

Profª. Drª. ADRIANA CRISTINA CHERRI NICOLA Adriana Cherri

Prof. Dr. PEDRO AUGUSTO MUNARI JUNIOR P/ Adriana Cherri

Profª. Drª. EDILAINE MARTINS SOLER Edilaine M. Soler

Arthur Medeiros Figueiredo Barreto

O uso de K **Soluções para o problema de corte de estoque com sobras aproveitáveis**

Dissertação apresentada como requisito à obtenção do grau de Mestre em Engenharia de Produção, Programa de Pós-Graduação em Engenharia de Produção da Faculdade de Engenharia de Bauru - FEB - da Universidade Estadual Paulista - UNESP - Campus Bauru

Banca examinadora

Profa. Dra. Adriana Cristina Cherri
UNESP - Bauru
Orientadora

Prof. Dr. Pedro Augusto Munari Júnior
UFSCAR - São Carlos

Profa. Dra. Edilaine Martins Soler
UNESP - Bauru

Bauru
2019

AGRADECIMENTOS

Agradeço aos meus orientadores Adriana Cherri e Luiz Cherri pela paciência nas inúmeras correções, e peço para que mantenham a mesma paciência no doutorado.

Agradeço à minha família e aos colegas de trabalho que me auxiliaram durante o decorrer do trabalho.

Agradeço à CAPES e ao Departamento de Pós graduação de Engenharia de Produção pelo auxílio fornecido.

Agradeço ao Felipe "yoda"Noronha pelo incentivo para continuar sempre estudando. Como ele diria, "isso aqui é trabalho". Meu muito obrigado meu amigo.

RESUMO

Este trabalho propõe diferentes estratégias de solução para a técnica de geração de colunas utilizada para resolver o problema de corte de estoque com sobras aproveitáveis (PCESA). Este problema consiste em atender a demanda de produção de itens solicitados por clientes a partir do corte de objetos comprados de fornecedores ou de sobras resultantes de cortes anteriores. O objetivo é de cunho econômico e visa, por exemplo, a minimização da perda gerada com o corte dos objetos. Neste problema, durante o processo de corte, sobras podem ser geradas para o estoque e não são contabilizadas como perdas. Uma técnica bastante utilizada para resolver este problema, assim como o clássico problema de corte de estoque, é o método de geração de colunas, que é um método iterativo no qual a cada iteração novas colunas (padrões de corte) são geradas a fim de encontrar a melhor solução para a relaxação linear do problema. A geração de colunas utiliza o problema da mochila para gerar as colunas que serão inseridas a cada iteração no problema de corte. Desta forma, o objetivo deste trabalho é estudar o impacto da inserção de K soluções, obtidas pelo problema da mochila em instâncias do PCESA unidimensional, a cada iteração do método de geração de colunas. Para as estratégias propostas, testes computacionais foram realizados utilizando classes de instâncias que variaram o valor de K. Os resultados mostraram que as estratégias reduzem o número de iterações e podem reduzir o tempo computacional em relação a geração de colunas padrão, quando apenas uma coluna é inserida a cada iteração .

Palavras-chave: Problema de corte de estoque com sobras aproveitáveis, K soluções, geração de colunas, problema da mochila.

ABSTRACT

This work proposes different solution strategies to the column generation method used to solve the cutting stock problem with usable leftovers (CSPUL). This problem consists in meeting client demands cutting objects bought from the market or leftovers resulted from previous cuts. The objective is economic and aims, for example, to minimize the total waste generated by cutting objects. In this problem, during the cutting process, leftovers can be generated for the stock and are not counted as wastes. A very useful technique to solve this problem, as well as the classic cutting stock problems, is the column generation method that is an iterative method. In each iteration of the method, new columns (cutting patterns) are generated in order to find the best solution for the linear relaxation of the problem. The column generation uses the knapsack problem to generate the columns that will be inserted at each iteration in the cutting problem. Thus, the objective of this work is to study the impact of the insertion of K solutions obtained in the knapsack problem in instances of the one-dimensional CSPUL in each iteration of the column generation method. For the proposed strategies, computational tests were performed using classes of instances that varied the k value. The results showed that the strategies reduce the number of iterations and can reduce computational time in relation to the standard column generation, when only one column is inserted at each iteration.

Key-words: Cutting stock problem with usable leftovers, K solutions, column generation, knapsack problem.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

INTRODUÇÃO

Com o avanço da tecnologia, as indústrias estão cada vez mais competitivas e, para se manterem no mercado, estas precisam determinar a melhor forma para realizar seus processos produtivos. Os problemas de corte de estoque (PCE) aparecem no setor industrial e são caracterizados pelo corte de objetos disponíveis em estoque, para atender uma demanda de itens previamente estabelecida. Nestes problemas, geralmente busca-se minimizar a perda, reduzir o número de objetos utilizados para atender uma demanda, maximizar o lucro, entre outros objetivos.

Os trabalhos pioneiros na literatura de corte foram apresentados na década de 60 por Kantorovich (1960), Gilmore e Gomory (1961) e Gilmore e Gomory (1963). Desde então, as pesquisas nessa área evoluíram na busca de métodos de solução eficientes para a resolução desses problemas e na elaboração de novos modelos matemáticos que contemplem as particularidades encontradas devido aos processos industriais em que esses problemas estão inseridos.

Uma das variações estudadas na literatura do PCE refere-se à geração de sobras durante o processo de corte. Estas sobras, desde que apresentem comprimentos suficientemente grandes, não são computadas como perdas e podem ser utilizadas como objetos no atendimento de futuras demandas. Na literatura, este problema é conhecido como problema de corte de estoque com sobras aproveitáveis (PCESA).

De acordo com Arenales et al. (2015), uma das características do PCESA é a redução da perda de material, devido a maior diversificação dos objetos em estoque, obtida pela inserção de sobras. Suas aplicações podem ser vistas em diversas situações, como na indústria têxtil em Gradišar, Jesenko e Resinovič (1997), no corte de tubos em Chu e Antonio (1999) e

Abuabara e Morabito (2009), entre outros.

Uma técnica bastante utilizada na resolução do PCE, assim como na resolução do PCESA, é o método de geração de colunas. Neste procedimento, as colunas geradas são padrões de corte que representam a maneira como um objeto é cortado em itens. Este método pode ser definido como um processo iterativo composto pelo modelo matemático que representa o problema mestre, e um problema secundário chamado de subproblema. O problema mestre tem a função de escolher entre as colunas disponíveis, a melhor combinação que otimize a função objetivo. O subproblema tem por finalidade fornecer novas colunas a cada iteração ao problema mestre. A execução do método termina quando não existirem mais colunas que sejam capazes de melhorar a função objetivo do problema mestre. Para um PCE, o subproblema é conhecido na literatura como problema da mochila e as colunas são parâmetros utilizados no problema mestre.

Embora uma ou várias soluções do subproblema possam ser inseridas no problema mestre, não se conhece nenhum trabalho da literatura que analisa o impacto de se inserir K soluções com critério de seleção do subproblema a cada iteração do método de geração de colunas no PCE, especialmente no PCESA. Com esta estratégia, é possível reduzir o número de iterações durante a resolução do problema e, consequentemente, o tempo computacional. De acordo com Yanasse, Soma e Maculan (2000) e Leão, Cherri e Arenales (2014), a possibilidade de aplicação das K-melhores soluções do problema da mochila em PCE é um estudo promissor.

Na resolução do PCESA, como sobras retornam ao estoque para atender futuras demandas, há grande variação de objetos para ser analisada durante o processo de corte, tornando sua resolução demorada para muitas instâncias. Desta forma, neste trabalho, K-soluções com critério de seleção do problema da mochila serão utilizadas na resolução do PCESA. Neste estudo, o modelo matemático proposto por Arenales et al. (2015) será utilizado. Quatro estratégias para a geração das K-soluções com critério de seleção foram propostas. Em duas das estratégias houve redução do tempo computacional para a resolução de problemas de grande porte quando comparado com o procedimento de geração de colunas padrão, em que uma única coluna é inserida no problema mestre a cada iteração, ou várias colunas são inseridas sem qualquer análise. Além disso,todas as estratégias convergiram com um menor número de iterações para a solução ótima, o que demonstra o potencial da criterização na seleção dos padrões de corte. As vantagens e desvantagens das estratégias propostas

também são apresentadas. Para simplificar a escrita, no restante do texto será utilizado K-soluções para referir as K-soluções com critério de seleção.

1.1 Estrutura do trabalho

Além desta introdução, esta dissertação está dividida da seguinte forma. O Capítulo 2 apresenta as primeiras formulações propostas para o PCE e uma revisão da literatura referente ao PCESA. O método de geração de colunas e a base teórica para seu funcionamento, bem como formas de seleção de K-soluções com critério de seleção também são apresentados nesta seção. No Capítulo 3, são apresentadas as quatro estratégias de seleção das K-soluções utilizadas nesta dissertação. Para fins didáticos, a resolução de uma pequena instância descrevendo a comparação entre as quatro estratégias propostas é apresentada passo a passo. O Capítulo 4 contém os resultados obtidos a partir de instâncias derivadas da literatura com a aplicação das estratégias propostas, em que são apresentadas análises acerca das soluções obtidas. Por fim, no Capítulo 5 apresentamos os objetivos atingidos e as expectativas para trabalhos futuros.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta uma breve revisão teórica sobre o PCE e os principais artigos que abordam o PCESA. O problema da mochila, o algoritmo de geração de colunas, assim como a base teórica para as 4 estratégias utilizadas para a seleção das K-soluções para o problema da mochila também são apresentadas neste capítulo.

2.1 Problemas de corte de estoque

Problemas de corte de estoque estão presentes no cotidiano de muitas empresas e podem ser vistos em diversas situações, como por exemplo no corte de bobinas, no corte de espuma de colchões e no corte de barras de ferro, entre outros. Para resolver estes problemas é necessário estabelecer as maneiras de cortar (padrões de corte) a matéria prima disponível em estoque (objetos) e definir a frequência na qual estes padrões serão utilizados a fim de atender a demanda.

Um padrão de corte representa a quantidade que cada item demandado será cortado do objeto em estoque. De modo geral, um padrão de corte é dado por:

$$a = (a_1, a_2, a_3, ..., a_m) \tag{2.1}$$

em que a_i, $i = 1, \ldots, m$ representa a quantidade de itens do tipo i no padrão de corte. Múlti-

plos padrões de corte podem ser representados na forma matricial por:

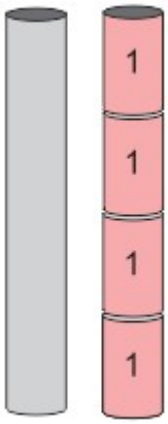
$$\mathrm{A} = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix} \tag{2.2}$$

em que a_{ij} representa a quantidade de itens do tipo i no padrão de corte $j, i = 1 \ldots m, j = 1, \ldots n$, sendo m a quantidade de itens e n o número de padrões de corte. Após determinar o conjunto de padrões de corte, deve-se estabelecer a frequência com que eles serão utilizados para atender a demanda.

Wäscher, Haußner e Schumann (2007) apresentam uma tipologia na qual caracterizam os problemas de corte e empacotamento em oito classes distintas. As principais características analisadas são: objetivo, número de tipos de itens e graus de liberdade nas dimensões dos objetos. Quanto ao número de dimensões, o PCE é classificado como unidimensional se houver apenas uma dimensão relevante para o processo de corte, bidimensional se houver duas dimensões relevantes ou tridimensional quando três dimensões são relevantes. Neste trabalho, o PCE unidimensional será abordado.

Na Figura 2.1, é ilustrado um padrão de corte unidimensional homogêneo. Neste tipo de padrão, apenas um tipo de item é cortado.

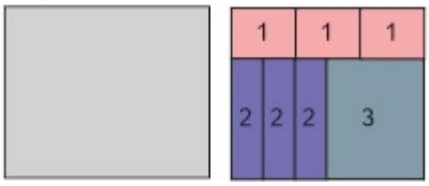
Figura 2.1: Corte unidimensional de um padrão homogêneo.



Quando duas dimensões são relevantes para o processo de corte (comprimento e largura), este é caracterizado como um problema de corte bidimensional. A Figura 2.2 apresenta um padrão de corte bidimensional.

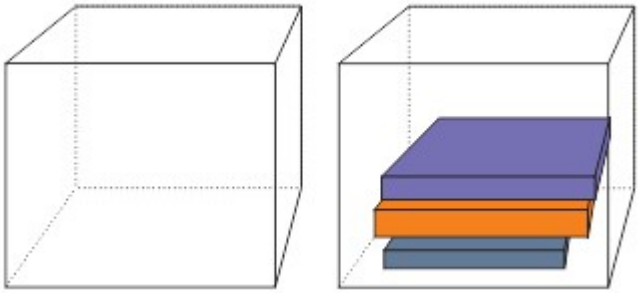
Outro tipo de corte presente na literatura é o corte em três dimensões, chamado de tri-

Figura 2.2: Corte bidimensional



dimensional. Suas principais aplicações são em problemas de empacotamento, como por exemplo o carregamento de contêineres. A Figura 2.3 apresenta um exemplo de padrão de corte em três dimensões.

Figura 2.3: Corte tridimensional.



O PCE t-dimensional, $t > 1$, é mais difícil de se resolver em relação ao PCE unidimensional devido a natureza dos padrões de corte. Os padrões de corte t-dimensionais devem incluir a quantidade e a posição dos m itens no objeto, o que os tornam mais complexos em relação a um padrão unidimensional, em que se considera apenas as quantidades.

Por ser abordado neste estudo, na Seção 2.2, é apresentada uma revisão bibliográfica referente ao PCE unidimensional.

2.2 Problemas de Corte de Estoque unidimensional

Os PCE consistem em atender uma determinada demanda de itens a partir do corte de objetos disponíveis em estoque. Entre os possíveis objetivos a serem otimizados, a minimização das perdas e a diminuição do número de objetos utilizados aparece constantemente nos trabalhos da literatura.

A primeira abordagem para o PCE foi proposta em 1939 por Kantorovich e posteriormente traduzida para o inglês em Kantorovich (1960). Conforme Lübbecke e Desrosiers (2005), Longhi (2013) e Longhi, Melega e Araujo (2014), o modelo matemático (2.3) - (2.7) foi atribuído à Kantorovich (1960), embora não apareça explicitamente no trabalho. O mo-

delo (2.3) - (2.7) e seus parâmetros estão representados a seguir.

Parâmetros:

- m: tipos de itens demandados;
- n: número de objetos em estoque;
- L: comprimento do objeto;
- b_j: demanda do item do tipo j;
- l_j: comprimento do item j.

Variáveis:

- y_i: 1 se o objeto i é usado durante o processo de corte, 0 caso contrário;
- x_{ij}: número de vezes que o item j é cortado no objeto i.

Modelo matemático:

$$\text{Min} \sum_{i=1}^{n} y_i \tag{2.3}$$

Sujeito a:

$$\sum_{i=1}^{n} x_{ij} \geq b_j \qquad j = 1, ..., m \tag{2.4}$$

$$\sum_{j=1}^{m} l_j x_{ij} \leq \text{L} y_i \qquad i = 1, ..., n \tag{2.5}$$

$$x_{ij} \in \mathbb{Z}_+, \qquad i = 1, ..., n,\ j = 1, ..., m \tag{2.6}$$

$$y_i \in \{0, 1\}, \qquad i = 1, ..., n \tag{2.7}$$

No modelo matemático (2.3) - (2.7) são cortados até n objetos disponíveis em estoque de tamanho L, para a produção de m tipos de itens, de tamanhos l_1, l_2, ..., l_m. A função objetivo (2.3) minimiza o número de objetos cortados. As restrições (2.4) garantem o atendimento da demanda, enquanto que, (2.5), asseguram que a quantidade de itens cortados em cada objeto não exceda o seu comprimento. As restrições (2.6) e (2.7) referem-se ao domínio das variáveis.

Gilmore e Gomory (1961) propuseram o método simplex com geração de colunas para a resolução de um modelo de otimização linear que, pela primeira vez foi capaz de resolver o PCE. Posteriormente, Gilmore e Gomory (1963) apresentaram uma nova metodologia para o problema da mochila, com as soluções para o problema enumeradas através de uma busca em profundidade. Nos PCE, o problema da mochila surge como um subproblema a ser resolvido.

Haessler (1975) apresentou um procedimento heurístico visando resolver o PCE através da associação de um custo fixo associado ao padrão de corte. Em Haessler (1980), foi modificado o procedimento de geração de colunas proposto por Gilmore e Gomory (1961) restringindo a quantidade de vezes que cada item poderia aparecer em um padrão de corte. O benefício desta estratégia é a redução das trocas de padrões geradas pelo aumento da quantidade de uso de cada padrão. Estes trabalhos apresentam grande relevância industrial, pois a mudança dos padrões de corte durante o processo produtivo pode implicar diretamente em custos com maquinário, aumento do tempo de preparação dos equipamentos entre outros.

Em Hinxman (1980) é apresentada uma revisão sobre a literatura do PCE unidimensional. O autor aponta que o método de geração de colunas apresentado em Gilmore e Gomory (1961), Gilmore e Gomory (1963) é a técnica mais eficiente para a resolução do PCE unidimensional e bidimensional, dado que bons métodos de solução também sejam aplicados na resolução dos subproblemas pertencentes à geração de colunas.

Stadtler (1990) propôs uma estratégia para minimizar a quantidade de objetos necessários para atender a demanda de itens. Para melhorar os resultados não satisfatórios da heurística FFD (*First-Fit-Decreasing*) utilizada até o momento, o autor apresentou um novo método, baseado no processo de geração de colunas (Gilmore e Gomory (1961), Gilmore e Gomory (1963)), acrescido de um procedimento de arredondamento para obtenção de soluções inteiras (heurística residual).

Wäscher e Gau (1996) reuniram em um artigo vários métodos heurísticos para a resolução do PCE. Os autores destacam o *trade-off* entre qualidade da solução e tempo computacional gasto na solução do problema inteiro. Um destes métodos heurísticos está descrito no Capítulo 3 e foi utilizado neste trabalho.

Em Poldi e Arenales (2009), foram apresentadas heurísticas construtivas e residuais da literatura para resolver o PCE, assim como foram propostas heurísticas residuais para resolver

o problema.

Em Jahromi et al. (2012) os autores resolvem o PCE unidimensional através de duas meta-heurísticas: busca tabu e *simulated annealing*. Os autores comparam as meta heurísticas em termos de tempo de resolução e qualidade da solução oferecida. De acordo com os autores, embora a busca-tabu forneceu as piores soluções, ela foi responsável pelos menores tempos de processamento.

Cui, Zhong e Yao (2015) abordam o PCE unidimensional com custos de *setup* para redução do número de padrões de corte. Após gerar um conjunto de padrões de corte, os mesmos são utilizados para resolver o PCE unidimensional tentando reduzir o número de padrões de corte da solução final. Os autores utilizam instâncias da literatura e demonstram que o método proposto supera a eficiência de modelos da literatura.

2.3 Problema de corte de estoque com sobras aproveitáveis

O PCESA é caracterizado pelo corte de objetos ou sobras disponíveis em estoque para a produção de itens demandados. Durante o processo de corte sobras podem ser geradas e retornam ao estoque para atenderem demandas futuras. Essas sobras possuem comprimento previamente definido e podem ser geradas em quantidades limitadas, desde que as perdas sejam reduzidas.

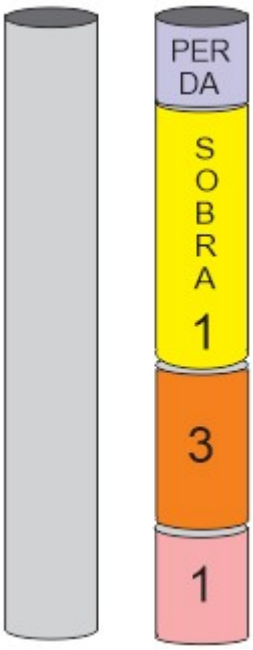
A Figura 2.4 ilustra um padrão de corte com uma possível sobra.

A possibilidade de gerar sobras para uso futuro aumenta a diversidade de objetos em estoque. Consequentemente, há maior diversidade de possíveis padrões de corte.

O PCESA foi abordado no trabalho de Roodman (1986). Este trabalho teve como objetivo principal minimizar a perda e como objetivo secundário concentrar as sobras em um número reduzido de padrões de corte para que pudessem retornar ao estoque. Para a resolução do problema o autor utilizou o modelo de Gilmore e Gomory (1963) e propôs um procedimento heurístico em três fases para factibilizar a solução e concentrar as sobras em poucos padrões de corte.

Scheithauer (1991) incluiu itens fictícios aos demandados (possíveis sobras), sem demanda para serem atendidas. Estes itens, quando cortados, retornavam ao estoque para serem utilizados no futuro. O problema abordado considerou um único tipo de objeto em estoque e foi resolvido pelo método de geração de colunas, seguido de um procedimento

Figura 2.4: Padrão de corte PCESA unidimensional



heurístico. Em Sinuany-Stern e Weiner (1994) os autores desenvolveram um estudo de caso para uma fábrica de pequeno porte que possui em sua linha de produção o corte de barras e canos de metais. Um modelo matemático com dois objetivos foi proposto para representar o problema. Os objetivos consistem em minimizar a perda e acumular as sobras no último objeto a ser cortado. Para resolver o problema, um procedimento heurístico foi proposto.

Gradišar, Jesenko e Resinovič (1997) resolveram o PCESA com o objetivo de minimizar a perda e a quantidade de itens não atendidos. Um modelo matemático foi proposto para representar o problema e, devido a sua complexidade, um procedimento heurístico, denominado COLA, foi desenvolvido para sua resolução.

Chu e Antonio (1999) abordaram o PCESA unidimensional em uma fábrica especializada no corte de metais. Os autores permitiram a geração de sobras para atender demandas futuras e minimizaram a perda resultante do corte de objetos, das sobras e o tempo gasto durante o processo do corte de metal. A quantidade de sobras a ser gerada foi limitada pela capacidade de armazenamento da empresa e pelos custos de locomoção das próprias sobras.

Trkman e Gradisar (2007) destacaram a importância de adequar métodos que consideram o aproveitamento de sobras para que não haja acúmulo de sobras no estoque e propuseram um método de solução que considera esta condição. O estudo abordou um problema em períodos de tempo e o método se mostrou eficiente especialmente para os últimos períodos de um horizonte de planejamento, não apresentando infactibilidade e sendo capaz

de obter perdas menores a longo prazo. Abuabara e Morabito (2009) reescreveram o modelo proposto por Gradišar, Jesenko e Resinovič (1997) como um problema inteiro misto para resolver o PCESA em uma empresa aeronáutica. O problema considerou diferentes tipos de objetos em estoque em quantidades suficientes para atender toda demanda de itens e possíveis sobras geradas em cortes anteriores.

Em Cherri, Arenales e Yanasse (2009), foram propostas modificações em heurísticas clássicas da literatura para resolver o PCESA. Com as alterações propostas, as sobras de materiais eram acumuladas em poucos objetos com comprimentos suficientes para retornar ao estoque. Cui e Yang (2010) propuseram uma extensão do trabalho de Scheithauer (1991) diversificando os tipos de objetos disponíveis em estoque e limitando a quantidade de sobras que poderiam ser geradas durante o processo de corte.

Cherri, Arenales e Yanasse (2013) modificaram as heurísticas desenvolvidas em Cherri, Arenales e Yanasse (2009) priorizando o corte de retalhos disponíveis em estoque. Cherri et al. (2014) escreveram uma revisão envolvendo artigos da literatura que resolvem o PCESA. Os autores apresentaram a modelagem matemática, quando proposta, estratégias de solução adotada pelos autores, comentários sobre os resultados obtidos e propostas para a continuidade de estudos envolvendo os PCESA. Neste trabalho, os autores observaram que, de modo geral, as sobras são tratadas de forma implícita nos problemas.

Arenales et al. (2015) propuseram um modelo matemático para o PCESA. Diferentemente dos trabalhos anteriores, os retalhos foram tratados de forma explícita, sendo possível prédefinir seus comprimentos e limitar suas quantidades quando gerados. Além disso, os retalhos em estoque podem ser utilizados prioritariamente, sendo esta uma necessidade em algumas situações práticas como, por exemplo, no corte de bobinas de aço em que, após desembaladas, as mesmas oxidam em um determinado período de tempo. Para resolver este problema, os autores relaxaram a condição de integralidade do modelo proposto e utilizaram o método simplex com geração de colunas proposto em Gilmore e Gomory (1963) . Nenhum procedimento heurístico foi proposto para obtenção de soluções inteiras.

Tomat e Gradišar (2017) propuseram uma estratégia para determinar o comprimento ideal de novas sobras para que as mesmas não permanecessem muito tempo em estoque. Em Coelho et al. (2017) é proposto um modelo matemático que considera a tomada de decisão entre vender e usar as sobras disponíveis em estoque. Dois procedimentos heurísticos são propostos e a análise das soluções obtidas são realizadas em termos de implicações sus-

tentáveis.

Devido às particularidades do modelo matemático proposto por Arenales et al. (2015) o mesmo será utilizado neste trabalho para analisar o comportamento da geração de colunas quando diferentes estratégias para a geração das K-soluções na resolução do subproblema serão utilizadas. Desta forma, o modelo matemático é apresentado em detalhes a seguir.

2.3.1 O modelo matemático de Arenales et al. (2015)

O modelo matemático proposto em Arenales et al. (2015) consiste em determinar o conjunto de itens demandados, a partir do corte de objetos padronizados (objetos comprados no mercado) ou de sobras de processos de corte anteriores. Como objetivo, busca-se minimizar a perda (mensurada em unidades de comprimento) permitindo que sobras com dimensões e quantidades pré definidas sejam geradas para estoque. Para modelar esse problema, os seguintes parâmetros e variáveis foram definidos:

- S: número de tipos de objetos padronizados. São denotados objetos do tipo s, $s \in \{1,...,S\}$.
- R: número de tipos de sobras em estoque. São denotadas sobras do tipo s, $s \in \{S+1,...,R\}$.
- e_s: número de objetos/sobras tipo s disponíveis em estoque s, $s = 1,...,S+R$;
- m: número de tipos de itens demandados;
- d_i: demanda do item do tipo i, $i = 1,...,m$;
- J_s: conjunto de padrões de corte para o objeto do tipo s, $s = 1,...,S+R$;
- $J_s(k)$: conjunto de padrões de corte para objetos padronizados do tipo s gerando sobra do tipo $S+k$, k=1,..., R, $s = 1,...,S$;
- a_{ijs}: número de itens do tipo i no padrão de corte j para objeto do tipo s, $i = 1,...,m$, $s = 1,...,S+R$, $j \in J_s$;
- a_{ijsk}: número de itens do tipo i no padrão de corte j para o objeto s gerando sobra do tipo $S+k$, $i = 1,...,m$, $k = 1,...,R$, $s = 1,...,S$, $j \in J_s(k)$;

- c_{js}: perda em unidade de comprimento por cortar o objeto/sobra tipo s no padrão de corte j, $s = 1,...,$ S + R, $j \in J_s$;
- c_{jsk} perda em unidade de comprimento por cortar o objeto s no padrão de corte j gerando uma sobra do tipo S + k, $s = 1,...,$ S, $k = 1,...,$ R, $j \in J_s(k)$;
- U: número máximo de sobras;
- α': prioridade para a geração de sobras, $\alpha' \in [0,1]$;
- α'': prioridade para o uso de sobras, $\alpha'' \in [0,1]$;

Variáveis:

- x_{js}: número de objetos do tipo s cortados de acordo com o padrão de corte j, $s = 1,...,$S + R, $j \in J_s$;
- x_{jsk}: número de objetos do tipo s cortados de acordo com o padrão de corte j e gerando uma sobra do tipo S+k, $s = 1,...,$S, $k = 1,\ldots,$R, $j \in J_s(k)$.

$$\text{Min } f(x) = \sum_{s=1}^{S} \sum_{j \in J_s} c_{js} x_{js} + \alpha' \sum_{s=1}^{S} \sum_{k=1}^{R} \sum_{j \in J_s(k)} c_{jsk} x_{jsk} + \alpha'' \sum_{s=S+1}^{S+R} \sum_{j \in J_s} c_{js} x_{js} \tag{2.8}$$

Sujeito a:

$$\sum_{s=1}^{S} \sum_{j \in J_s} a_{ijs} x_{js} + \sum_{s=1}^{S} \sum_{k=1}^{R} \sum_{j \in J_s(k)} a_{ijsk} x_{jsk} + \sum_{s=S+1}^{S+R} \sum_{j \in J_s} a_{ijs} x_{js} = d_i, \qquad i = 1,...,m \tag{2.9}$$

$$\sum_{j \in J_s} x_{js} + \sum_{k=1}^{R} \sum_{j \in J_s} x_{jsk} \leq e_s, \qquad s = 1,...,S \tag{2.10}$$

$$\sum_{j \in J_s} x_{js} \leq e_s, \qquad s = S+1,...,S+R \tag{2.11}$$

$$\sum_{s=1}^{S} \sum_{k=1}^{R} \sum_{j \in J_s(k)} x_{jsk} - \sum_{s=S+1}^{S+R} \sum_{j \in J_s} x_{js} \leq U - \sum_{s=S+1}^{S+R} e_s \tag{2.12}$$

$$x_{js} \geq 0,\ s = 1,...,S+R,\ j \in J_s \text{ e inteiro.} \tag{2.13}$$

$$x_{jsk} \geq 0,\ k = 1,...,R,\ s = 1,...,S,\ j \in j_s(k) \text{ e inteiro.} \tag{2.14}$$

No modelo (2.8) - (2.14), a função objetivo minimiza a perda total proveniente do corte de objetos padronizados, objetos padronizados que geram sobras e de sobras disponíveis em

estoque. As restrições (2.9) buscam atender a demanda, enquanto que as restrições (2.10) e (2.11) limitam, respectivamente, a utilização de objetos padronizados e de sobras existentes em estoque. A restrição (2.12) limita a quantidade máxima de sobras geradas e as restrições (2.13) e (2.14) caracterizam o domínio das variáveis.

Devido ao elevado número de variáveis e as condições de integralidade nas restrições (2.13) e (2.14) é inviável resolver o modelo (2.8) - (2.14) na otimalidade até mesmo para instâncias pequenas. Para resolver o problema, os autores relaxaram a condição de integralidade e uma solução ótima contínua foi obtida utilizando o método simplex com geração de colunas de Gilmore e Gomory (1963).

De acordo com o modelo (2.8) - (2.14), são obtidos 3 tipos de padrões de corte. Padrões de corte a partir de objetos padronizados sem a geração de sobra, com a geração de sobra e a partir de retalhos. Desta forma, uma grande variedade de colunas precisam ser geradas e testadas durante a resolução deste problema. Com a finalidade de melhorar o desempenho do método de geração de colunas, quatro estratégias para a geração das K-soluções na resolução do subproblema (problema da mochila) foram propostas.

2.4 Método de geração de colunas

A técnica de geração de colunas é aplicada frequentemente em problemas clássicos da literatura, como problemas de corte de estoque, problemas de roteamento de veículos com janela de tempo, e problemas de dimensionamento de lotes (GONDZIO; GONZÁLEZ-BREVIS; MUNARI, 2013). A geração de colunas é um método iterativo que encontra a solução para a relaxação linear de um problema. Nesta estratégia, a coluna mais atrativa para a solução é descoberta, e isso implica em resolver um problema da mochila.

Utilizar o algoritmo de geração de colunas para resolver um problema de otimização linear consiste em dividir o problema em duas partes: o problema mestre (PM) e o subproblema (SP). No caso dos PCE, uma solução para o problema mestre também é uma solução para o problema de corte relaxado, enquanto o subproblema fornece novos padrões de corte (colunas) que melhoram esta solução. Este procedimento é iterativo e ocorre até não haver mais colunas que melhorem a solução do problema mestre.

A escolha da coluna mais atrativa é usualmente feita através do critério de Dantzig em

que se escolhe a coluna com o menor custo relativo, conforme 2.15.

$$c_k - \pi^{\mathrm{T}} * a_k = min\left\{c_j - \pi^{\mathrm{T}} * a_j, \forall j\right\} \tag{2.15}$$

Em (2.15), c_k corresponde as perdas de matéria prima, sendo calculadas por: L$-\sum_{i=1}^{m} a_i l_i$, em que L e l_i representam respectivamente o tamanho do objeto e do item i. O vetor π^{T} representa as variáveis duais do problema e a_k é a coluna gerada pelo subproblema (problema da mochila). As colunas são geradas pelo subproblema respeitando a restrição do problema da mochila:

$$\begin{aligned} &\sum_{i=1}^{m} l_i a_i \leq \mathrm{L}_s, && s = 1, ..., \mathrm{S}+\mathrm{R} \\ &0 \leq a_i \leq d_i, \quad i \in \mathbb{Z}_+, && i = 1, ..., m \end{aligned} \tag{2.16}$$

em que a_i corresponde a quantidade de itens do tipo i na coluna, l_i representa o comprimento do item $i, i = 1, ..., m$, L_s o comprimento do objeto tipo $s, s = 1, ..., \mathrm{S}+\mathrm{R}$ a ser cortado e d_i corresponde a demanda do item i. Assim, a solução do problema da mochila corresponde ao padrão $(a_1, a_2, a_3, ..., a_m)$.

Considerando que π_i é o valor de utilidade do item $i, i = 1, ..., m$, para assegurar que o padrão de corte mais atrativo seja selecionado para compor o problema mestre, o seguinte subproblema deve ser resolvido:

$$\text{Max } f(a) = \sum_{i=1}^{m} \pi_i a_i \tag{2.17}$$

Sujeito a:

$$\sum_{i=1}^{m} l_i a_i \leq \mathrm{L}_s, \qquad s = 1, ..., \mathrm{S}+\mathrm{R} \tag{2.18}$$

$$0 \leq a_i \leq d_i, \qquad a_i \in \mathbb{Z}_+, \quad i = 1, ..., m \tag{2.19}$$

No modelo (2.17)-(2.19), a função objetivo (2.17) maximiza a utilidade dos itens, enquanto a restrição (2.18) garante que o conjunto dos itens a serem cortados não ultrapasse a capacidade da mochila. O conjunto de restrições (2.19) garante que a quantidade dos itens no padrão de corte não ultrapasse a demanda permitida.

O problema mestre de minimizar o número de objetos cortados é então dado pelo mo-

delo (2.20) - (2.22)

$$\text{Min } f(x) = \sum_{j=1} x_j \tag{2.20}$$

Sujeito a:

$$\sum_{j=1}^{n} a_{ij} x_j \geq d_i, \qquad i = 1, ..., m \tag{2.21}$$

$$x_j \geq 0 \tag{2.22}$$

No modelo (2.20)-(2.22), a função objetivo minimiza o número de objetos a serem cortados, enquanto a restrição (2.21) garante o atendimendo da demanda. A restrição 2.22 garante a não negatividade das variáveis. A variável x_j representa o número de objetos cortados.

2.4.1 Algoritmo de geração de colunas

O algoritmo de geração de colunas está representado no Algoritmo 1. O vetor *price* corresponde ao vetor das variáveis duais associadas a solução do PM.

Algoritmo 1: Geração de colunas

Etapa 1: colunas iniciais
  Gere as colunas iniciais;
  Resolva o PM com o conjunto de colunas iniciais;
Etapa 2: novas colunas
  Faça π = vetor *price* do PM e gere o SP;
  Resolva o SP e obtenha a coluna mais atrativa da iteração atual para melhorar a solução do PM;
  Faça custoRel = custo relativo da coluna mais atrativa;
  Se $custo\text{R}el < 0$
    Armazene a coluna e insira no problema mestre;
    Resolva o PM e volte ao início da Etapa 2;
  Senão
    Fim. A solução contínua ótima do PM foi encontrada.

As colunas iniciais utilizadas na Etapa 1 para um PCE podem ser os padrões de corte homogêneos que são padrões gerados com baixo esforço computacional.

O custo relativo indica se a coluna analisada tem potencial para melhorar a solução obtida pelo problema mestre, enquanto o vetor *price*, ou π, corresponde aos coeficientes da função objetivo (2.17).

De acordo com o modelo (2.8) - (2.14) a cada iteração do algoritmo de geração de colunas 3 tipos de subproblemas precisam ser resolvidos para considerar: objetos padronizados, objetos padronizados gerando sobra (aqui, um subproblema é resolvido para cada possibilidade de sobra a ser gerada) e sobras em estoque. Devido a grande quantidade de colunas que devem ser analisadas, para problemas de grande porte este processo pode ser demorado.

2.5 Múltiplas soluções para o problema da mochila

Resolver um PCE utilizando o método de geração de colunas consiste em a cada iteração do método obter uma coluna com o problema da mochila, a qual será inserida no problema de corte. Alternativamente, múltiplas colunas podem ser inseridas a cada iteração do método. Desta forma, diminuir o número de vezes que se resolve o problema da mochila, pode diminuir o tempo de processamento total do método de geração de colunas. Uma estratégia que contribui para esta redução, consiste na obtenção de múltiplas soluções do problema da mochila com critério de seleção, em apenas uma iteração.

Yanasse, Soma e Maculan (2000) apresentaram um algoritmo para resolver o problema da mochila considerando a possibilidade de extrair as K-melhores soluções obtidas e citam a possível aplicação em problemas de corte. Ao realizar variações no número de K-soluções os autores verificaram que o processo de busca das soluções ótimas através de *backtrack* apresentou uma curva com aparência exponencial.

Em Leão, Cherri e Arenales (2014) também foi desenvolvido um algoritmo para as K-melhores soluções para o problema mochila. Para o caso de encontrar uma única solução, foram realizadas comparações entre o algoritmo desenvolvido e o CPLEX, revelando que o algoritmo reduz o tempo computacional de resolução do problema na maioria das instâncias testadas. Ainda, o tempo computacional do algoritmo proposto cresce pouco com o aumento do numero K de soluções requisitadas.

O *software* de otimização de propósito geral IBM CPLEX Optimization Studio também fornece meios de se obter múltiplas soluções para modelos de programação linear inteira em geral, embora não sejam garantidamente as melhores. A seguir, apresentam-se duas

possibilidades de obtenção de múltiplas soluções para o problema da mochila a partir do *solver* CPLEX.

Solve e Populate

O *solver* CPLEX também pode ser utilizado para encontrar K soluções para o problema da mochila por dois meios distintos: 1) armazenar soluções incumbentes encontradas durante a resolução do modelo matemático com o comando *solve*, 2) gerar múltiplas soluções alternativas com a função *Populate*.

Os dois comandos se diferenciam com relação ao critério de parada. O comando *Solve* termina sua execução assim que encontra a solução ótima para o problema, retornando o conjunto de soluções factíveis encontradas nessa busca. Já o comando *Populate* executa uma heurística, que por sua vez tem outros critérios de parada, como tempo computacional, limite de soluções armazenadas, ou número de nós, não garantindo a otimalidade de nenhuma das soluções encontradas.

Assim, caso se deseje obter a solução ótima no conjunto de soluções encontradas, o comando *Solve* é o mais indicado. Caso se deseje uma maior diversidade de soluções, o comando *Populate* deve ser aplicado.

Os dois métodos de solução podem ser executados em conjunto ou separadamente. A execução em conjunto se dá pelo *Solve* em primeira instância e, assim que a melhor resposta for encontrada, é iniciada a busca por múltiplas soluções através do *Populate*.

A execução separadamente difere na diversidade dos resultados gerados. Embora com o comando *Solve* seja possível armazenar um conjunto de respostas durante sua execução, este número pode ser inferior a um número K pré-estabelecido. Com o *Populate* é possível usar como critério de parada o limite de respostas geradas, ou seja, obtêm-se K soluções (sem a garantia de otimalidade).

Os comandos *Solve* e *Populate*, embora tenham sido descritos para o problema da mochila, podem ser aplicados em outros problemas de Programação linear inteira mista (PLIM).

Callback

Callbacks são funções capazes de interromper o processo de otimização do CPLEX e substituir ou aprimorar os métodos do *software* por comandos dados pelo usuário. As *callbacks* podem ser usados para rejeitar soluções factíveis de baixa qualidade, inserir restrições

adicionais, escolher o próximo nó a ser explorado e etc. Elas também podem ser utilizadas como um método auxiliar para obter múltiplas soluções para o problema da mochila. Nesta dissertação foram utilizados a *callback Incumbent* e *Lazy Constraint*.

A *callback Incumbent* é executada sempre que uma solução incumbente, isto é, factível, é encontrada durante o processo de otimização. É possível rejeitar a solução atual para forçar o CPLEX a continuar a processo de seleção e tentar encontrar uma nova solução.

Para se obter múltiplas soluções utilizando a *callback Incumbent* é necessário armazenar uma solução factível e rejeitá-la na sequência. Logo, para se obter K soluções é necessário executar o procedimento de armazenamento e posterior rejeição da solução por pelo menos K vezes. Note que o fato do CPLEX rejeitar uma solução não garante que ela não seja encontrada novamente e, desta forma, encontrar K soluções distintas pode levar a um aumento do número de vezes que o método é executado.

A *Lazy constraint callback*, assim como a *callback Incumbent*, também é executada sempre que uma solução factível é encontrada. No entanto, a *callback Lazy constraint* permite a inserção de um conjunto de restrições especiais que inicialmente não fazem parte do modelo matemático que está sendo resolvido, mas que são verificadas somente se uma solução factível for encontrada. Estas restrições especiais são chamadas de *lazy constraints*.

Uma *lazy constraint* só é adicionada ao conjunto de restrições do modelo matemático quando a solução encontrada não a satisfazer, mesmo que esta solução seja factível para as restrições originais do modelo que está sendo resolvido.

Para se obter múltiplas soluções para o problema da mochila utilizando *lazy constraints* é necessário repetir o processo de encontrar uma solução + inserção de uma *lazy constraint* por K vezes. Para se garantir que as soluções a serem obtidas em seguida sejam diferentes, basta inserir uma *lazy constraint* que restrinja a solução atual. Desta forma K soluções distintas podem ser obtidas, o que garante que as soluções sejam diversificadas.

Capítulo 3

ESTRATÉGIA DE SOLUÇÃO

Neste capítulo são apresentadas as quatro estratégias propostas para a geração das K-soluções para o problema da mochila, a cada iteração do método de geração de colunas. Estas estratégias foram aplicadas na resolução do modelo (2.8) - (2.14) de Arenales et al. (2015) com a finalidade de reduzir o tempo computacional de resolução deste modelo.

Para resolver o modelo (2.8) - (2.14), Arenales et al. (2015) relaxaram a condição de integralidade das variáveis e utilizaram o método simplex com geração de colunas. Durante a resolução do problema, os autores optaram por inserir apenas uma coluna a cada iteração do método, garantindo que o problema mestre encontre uma melhor solução, se houver.

A diversidade de tipos de objetos e sobras em estoque implica na resolução de subproblemas para os objetos e para as sobras. Gerando até K padrões de corte para cada problema da mochila resolvido, o número de padrões inseridos a cada iteração no problema mestre é superior a K.

A obtenção das K soluções para o problema da mochila foi realizada por quatro estratégias distintas. Duas destas estratégias obtêm K-soluções para o problema da mochila, com foco na qualidade das soluções, sendo uma com o auxílio de um *callback* e outra por um método já proposto na literatura que utiliza o método *Branch and Bound*. As duas estratégias restantes não buscam as melhores soluções do problema da mochila, mas sim um conjunto de K-soluções. Uma delas obtém as K-soluções factíveis rapidamente, com o comando *solve* e a estrutura *solution pool*, enquanto outra estratégia emprega um *callback* para priorizar a diversidade entre as próprias soluções.

3.1 Estratégia K-*Solve*

Nesta estratégia a obtenção de K-soluções do problema da mochila durante a resolução do modelo (2.8) - (2.14) foi realizada pelo comando *Solve* e, para armazenamento destas soluções, a estrutura *solution pool* do CPLEX foi utilizada. Estas K-soluções são as de melhor qualidade encontradas pelo CPLEX durante sua busca pela solução ótima e não necessariamente as K-melhores do problema da mochila.

Uma particularidade desta estratégia está no valor de K. É possível que o CPLEX encontre a resposta ótima para o problema da mochila antes de encontrar K-soluções devido a eficiência das heurísticas do próprio *solver*, ou a própria característica do problema. Desativar as heurísticas possivelmente aumentariam o número de K-soluções obtidas a cada problema da mochila, mas prejudicariam a eficiência do *solver* como um todo.

Uma outra alternativa para aumentar o número de padrões de corte obtidos a cada iteração desta estratégia é o uso do comando *populate* junto com o comando *solve*. No entanto, testes preliminares publicados em Barreto et al. (2018) mostraram que as duas funções juntas geram um aumento indesejável no tempo computacional gasto para a resolução das instâncias do PCESA, e nem sempre possibilitarão redução no tempo computacional em comparação com a geração de colunas padrão, em que K = 1.

3.2 Estratégia K-Diversificação

A estratégia de diversificação consiste em duas etapas. A primeira etapa consiste em obter a solução ótima para o problema da mochila. A segunda etapa é iterativa, de forma que a cada solução encontrada uma restrição é adicionada, forçando o CPLEX a encontrar uma nova solução.

O parâmetro Q corresponde a soma do número de itens do padrão de corte e é utilizado como limitante nas restrições do problema da mochila na segunda etapa. A soma dos itens utilizados deve ser menor que $Q \cdot p$, em que p é chamado de coeficiente de diversificação e é representado por $p \in \mathbb{R}_+$ e $p \leq 1$. A restrição de diversificação é dada por 3.1.

$$\sum i = 1^{m} y_i \leq Q * p \tag{3.1}$$

O problema da mochila se torna mais restrito a medida que se diminui o valor de p.

Desta forma, um novo padrão de corte é obtido e o valor de Q atualizado. O procedimento se repete até se obter um total de K padrões. Caso a função objetivo obtida seja inferior ao melhor limitante conhecido, este padrão é rejeitado pela *callback Incumbent*.

A *callback Lazy Constraint* é utilizada para adicionar ao modelo a restrição de diversificação ao fim de cada resolução do problema da mochila.

3.3 Estratégia K-*Branch and Bound*

Este método utilizado para obter as K-melhores soluções para o problema da mochila foi baseado no trabalho de Leão, Cherri e Arenales (2014), em que o problema da mochila foi resolvido pelo *Branch and Bound* e as K-melhores soluções do problema da mochila foram obtidas em poucos segundos.

Essa estratégia é composta por 3 etapas: ordenação dos itens, obtenção de K-soluções e aperfeiçoamento do conjunto de soluções obtidas. A ordenação dos itens é realizada por uma classificação de atratividade dos próprios itens. Por exemplo, considere o modelo (2.17) - (2.19). Ordene os itens em ordem não-crescente de acordo com $\frac{\pi_1}{l_1} \geq \frac{\pi_2}{l_2} \geq \frac{\pi_3}{l_3} \geq \ldots \geq \frac{\pi_m}{l_m}$. Esse procedimento é rápido e facilita à próxima etapa obter soluções razoavelmente boas.

A segunda etapa consiste em aproveitar a ordenação anterior e obter um conjunto de K-soluções. Uma solução inicial para o problema da mochila é obtida incluindo o item mais atrativo o maior número inteiro possível de vezes. O espaço restante no problema da mochila é ocupado pelo item subsequente o tanto quanto for possível. Isto acontece até o último item caso ainda exista espaço vago para ser preenchido.

A obtenção das $K-1$ soluções restantes é realizada diminuindo em uma unidade o último item pertencente à solução inicial iterativamente para cada uma das soluções restantes. Por exemplo, suponha que a melhor solução obtida seja representada por um padrão de corte da forma $a_1 = (a_{11}, a_{21}, \ldots, a_{r1}, 0, 0)$ em que a_{ij} representa o item i no padrão de corte j e r refere-se ao índice do último item diferente de zero. A segunda solução diminui em uma unidade o valor de r, logo $a_2 = (a_{12}, a_{22}, \ldots, a_{r1} - 1, 0, 0)$ e assim por diante.

A terceira etapa compreende o aperfeiçoamento do conjunto de soluções obtidas. Se a primeira solução for um padrão de corte da forma $a_1 = (0, 0, \ldots, 0)$, a estratégia se encerra. Caso contrário, é possível melhorar o conjunto de soluções obtidas a partir de dois métodos: *short backtracking* e *long backtracking*.

O *short backtracking* consiste em diminuir o valor de r em uma unidade, encontrar uma solução melhor que a_k e atualizar o conjunto de soluções excluindo a última solução. O processo de diminuir em x unidades, $x > 0$ o valor de r da solução $a_1 = (a_{11}, a_{21}, \ldots, a_{r1}, 0, 0)$ e alocar itens no espaço restante é chamado de problema residual, e é representado por $\overline{f}(a^{-1})$ se $x = 1$. O *short backtracking* somente é aplicado caso a inclusão de itens posteriores ao item r na solução $a_2 = (a_{12}, a_{22}, \ldots, a_{r1} - 1, 0, 0)$ resulte em uma solução melhor que a k-ésima solução ($f(a_k)$) do conjunto de soluções obtido na segunda etapa. Se $\overline{f}(a^{-1}) < f(a_k)$, o *long backtracking* é aplicado e se retorna à terceira etapa. O *long backtracking* equivale a tornar nulo o valor de r e, de acordo com Leão, Cherri e Arenales (2014), os nós intermediários podados não fazem perder a otimalidade, uma vez que estes não produziriam as soluções desejadas.

A condição de parada desta estratégia é a etapa 3. Resolver o problema residual faz com que se caminhe em direção ao vetor nulo $(0, 0, \ldots, 0, 0, 0)$, que é nó inicial da árvore do *branch and bound*.

3.4 Estratégia K-Incumbentes

Esta estratégia consiste em obter K-soluções para o problema da mochila utilizando o método *solve* com o auxílio da *callback* Incumbente. Esta *callback* foi utilizada para rejeitar soluções factíveis de baixa qualidade e forçar a busca por soluções melhores.

O processo de resolução foi dividido em 3 etapas: obter a solução inicial, obter K-soluções e melhorar o conjunto de soluções obtidas. A primeira etapa desta estratégia consiste em armazenar a primeira solução incumbente encontrada, independente do valor da sua função objetivo.

Na segunda etapa, K-soluções factíveis são encontradas durante a resolução do problema da mochila. Todas as vezes que uma solução factível é encontrada, deve ser verificado se esta já foi encontrada ou se é uma nova solução. Se uma nova solução é encontrada, ela é armazenada. Caso contrário, é rejeitada utilizando o método *reject*, que força a busca por uma nova solução. Esta etapa acontece até que K-soluções sejam obtidas.

A terceira etapa do método consiste em refinar o conjunto de soluções obtidas. Caso o valor da função objetivo da pior solução entre as K armazenadas seja maior ou igual que o melhor limitante conhecido das soluções não exploradas, a otimização é abortada e as K

soluções já foram obtidas. Caso contrário, o processo de resolução continua, pois ainda é possível melhorar o conjunto de soluções encontradas. Nesta etapa, soluções incumbentes mais promissoras para o problema de corte, isto é, com melhor valor da função objetivo, substituem as soluções menos promissoras até se obter as K soluções incumbentes de melhor valor.

3.5 Exemplo numérico

Para ilustrar as quatro estratégias de solução propostas para o problema da mochila, um exemplo com cinco tipos de itens e dois objetos em estoque é apresentado. Neste exemplo, não é permitido gerar padrões de corte a partir de sobras.

As Tabelas 3.1 e 3.2 apresentam a quantidade de objetos em estoque e o número máximo de sobras que podem ser geradas.O parâmetro $p = 0,4$ foi utilizado na estratégia "Diversificação"e o exemplo fo resolvido utilizando K = 2.

Tabela 3.1: Dados dos itens

Item	**Tamanho**	**Demanda**
1	18	25
2	27	17
3	25	12
4	19	19
5	50	8

Tabela 3.2: Objetos, sobras e novas sobras

Objetos		**Sobras**		**Novas sobras**	
Tamanho	Estoque	Tamanho	Estoque	Tamanho	Máx. permitido
100	100	40	3	40	10
		50	6	50	

As Tabelas 3.3, 3.4, 3.5 e 3.6 exibem os padrões de corte obtidos a cada iteração para cada uma das estratégias. Os padrões de corte obtidos na iteração são apresentados juntamente com o valor do custo relativo (coluna CR) e a sobra gerada (se existir). A coluna FO apresenta o valor da função objetivo após a inserção dos padrões obtidos a cada iteração.

As 4 estratégias iniciaram a resolução do exemplo a partir de uma FO de 6130, obtida com a inclusão somente de padrões homogêneos. A resolução das estratégias termina com a exibição do último padrão de corte obtido pelo problema da mochila e inserido no problema

mestre. Os padrões de corte em negrito pode ter sido responsáveis pela diferença entre a convergência das 4 estratégias nas primeiras iterações.

Tabela 3.3: Estratégia K-*Solve*

Iteração	FO	Padrões	CR	Sobra
1	2610	**(5, 0, 0, 0, 0)**	-400	
		(1, 0, 0, 2, 0)	-240	40
		(1, 0, 0, 1, 0)	-150	50
2	1710	(0, 0, 4, 0, 0)	-300	
		(0, 0, 2, 0, 0)	-150	50
		(0, 0, 2, 0, 0)	-140	40
3	463,3	(1, 3, 0, 0, 0)	-220	
		(0, 2, 0, 0, 0)	-140	40
		(1, 1, 0, 0, 0)	-70	50
4	63,3	(0, 0, 0, 0, 2)	-100	
		(0, 0, 0, 0, 1)	-50	50
		(0, 0, 0, 0, 1)	-40	40
5	45,45	(4, 1, 0, 0, 0)	-6,6	
6	8	(1, 0, 1, 3, 0)	-5,9	
		(0, 0, 0, 5, 0)	-4,5	
		(0, 0, 0, 3, 0)	-2,7	40
7	3,8	(3, 1, 0, 1, 0)	-0,7	
8	0	(0, 3, 0, 1, 0)	-1,19	

Tabela 3.4: Estratégia K-Diversificação

Iteração	FO	Padrões	CR	Sobra
1	2610	**(5, 0, 0, 0, 0)**	-400	
		(1, 0, 0, 4, 0)	-400	
		(3, 0, 0, 0, 0)	-240	40
		(0, 0, 0, 3, 0)	-240	40
		(2, 0, 0, 0, 0)	-150	50
		(0, 0, 0, 2, 0)	-150	50
2	463,3	(0, 0, 4, 0, 0)	-300	
		(1, 3, 0, 0, 0)	-220	
		(0, 0, 2, 0, 0)	-150	50
		(0, 0, 2, 0, 0)	-140	40
		(0, 2, 0, 0, 0)	-140	40
		(1, 1, 0, 0, 0)	-70	50
3	28,18	(0, 0, 0, 0, 2)	-100	
		(0, 0, 0, 0, 1)	-50	50
		(0, 0, 0, 0, 1)	-40	40
		(4, 1, 0, 0, 0)	-6,6	
4	5	(1, 0, 1, 3, 0)	-3,2	
		(0, 3, 0, 1, 0)	-1,8	
5	0	(3, 1, 0, 1, 0)	-0,6	

Tabela 3.5: Estratégia K-*Branch and Bound*

Iteração	FO	Padrões	CR	Sobra
1	3130	**(5, 0, 0, 0, 0)**	-400	
		(4, 0, 0, 1, 0)	-400	
		(3, 0, 0, 0, 0)	-240	40
		(2, 0, 0, 1, 0)	-240	40
		(2, 0, 0, 0, 0)	-150	50
		(1, 0, 0, 1, 0)	-150	50
2	1910	(0, 0, 0, 5, 0)	-400	
		(0, 0, 0, 4, 0)	-300	
		(0, 0, 0, 3, 0)	-190	40
		(0, 0, 0, 2, 0)	-100	50
		(0, 0, 1, 1, 0)	-100	50
		(0, 0, 0, 2, 0)	-90	40
3	1060	(0, 0, 4, 0, 0)	-300	
		(1, 0, 3, 0, 0)	-220	
		(0, 0, 2, 0, 0)	-80	50
		(0, 0, 2, 0, 0)	-70	40
		(0, 1, 1, 0, 0)	-70	40
4	463,3	(1, 3, 0, 0, 0)	-220	
		(0, 3, 0, 1, 0)	-220	
		(0, 2, 0, 0, 0)	-75	40
		(1, 1, 0, 0, 0)	-5	50
		(0, 1, 0, 1, 0)	-5	50
5	63,3	(0, 0, 0, 0, 2)	-100	
		(0, 0, 2, 0, 1)	-50	
		(0, 0, 0, 0, 1)	-50	50
		(0, 0, 0, 0, 1)	-40	40
6	7,7	(4, 1, 0, 0, 0)	-6,6	
		(3, 1, 0, 1, 0)	-6,6	
7	0	(1, 0, 1, 3, 0)	-2,7	
		(2, 0, 1, 2, 0)	-0,5	

Neste exemplo, foram obtidos no máximo 6 padrões de corte a cada iteração. No entanto, apenas a estratégia K-*Solve* na tabela 3.3 não apresentou 6 padrões de corte na primeira e segunda iterações. Este comportamento já era esperado uma vez que esta estratégia não prioriza buscar K soluções, mas sim a melhor solução e, somente se for possível, armazenar soluções auxiliares ao longo do caminho.

Embora a estratégia K-*Solve* tenha obtido metade do número de padrões de corte na primeira iteração em relação as demais, esta apresentou uma função objetivo na primeira iteração de 2610, contra 3130 da estratégia K-*Branch and Bound*. Isto acontece devido a diferença dos padrões de corte inseridos. Repare que as 2 estratégias inserem o padrão (A)

Tabela 3.6: Estratégia K-Incumbentes

Iteração	FO	Padrões	CR	Sobra
1	2275	**(5, 0, 0, 0, 0)**	-400	
		(4, 1, 0, 0, 0)	-400	
		(3, 0, 0, 0, 0)	-240	40
		(1, 0, 0, 2, 0)	-240	40
		(2, 0, 0, 0, 0)	-150	50
		(1, 1, 0, 0, 0)	-150	50
2	466,6	(0, 0, 4, 0, 0)	-300	
		(0, 1, 2, 1, 0)	-255	
		(0, 0, 2, 0, 0)	-100	50
		(0, 0, 2, 0, 0)	-90	40
		(0, 2, 0, 0, 0)	-90	40
		(0, 1, 0, 1, 0)	-55	50
3	57,5	(0, 0, 0, 0, 2)	-100	
		(0, 1, 0, 1, 1)	-50	
		(0, 0, 0, 0, 1)	-50	50
		(0, 0, 0, 0, 1)	-40	40
		(0, 0, 0, 3, 0)	-3,3	40
4	21,6	(0, 3, 0, 1, 0)	-10	
		(1, 3, 0, 0, 0)	-7,5	
5	0	(3, 1, 0, 1, 0)	-0,7	

$(5,0,0,0,0)$, mas a estratégia K-*Solve* insere o padrão (B) $(1,0,0,2,0)$, enquanto a estratégia K-*Branch and Bound* insere o padrão (C) $(2,0,0,1,0)$. Embora ambos os padrões contenham itens do mesmo tipo, o padrão (C) inserido pela estratégia K-*Branch and Bound* apresenta mais itens do tipo 4 em relação ao padrão (B), o que melhora a convergência para o ótimo uma vez que uma grande quantidade de itens do tipo 1 já estava contida no padrão (A).

As estratégias K-Incumbentes, K-*Branch and Bound* e K-Diversificação inseriram 6 padrões de corte com o mesmo custo relativo na primeira iteração e atingiram valores diferentes para a função objetivo, conforme pode ser visto nas tabelas 3.4, 3.5 e 3.6. A estratégia K-Incumbentes apresentou a menor FO (2275) pois além de inserir itens do tipo 1 e 4 com os padrões (A) e (B), também inseriu itens do tipo 2 com o padrão (D) $(4,1,0,0,0)$ e (E) $(1,1,0,0,0)$, enquanto as estratégias K-*Branch and Bound* e K-Diversificação inseriram apenas itens do tipo 1 e 4. Este comportamento evidencia a teoria de que diversificar os padrões de corte provoca uma melhor convergência em relação ao ótimo.

3.6 Solução inteira

Soluções inteiras para o PCESA foram obtidas para cada uma das estratégias apresentadas utilizando o procedimento heurístico proposto em Wäscher e Gau (1996). Este procedimento consiste em resolver o problema contínuo pelo método de geração de colunas e, em seguida, utilizar as colunas para resolver o problema considerando a integralidade das variáveis de decisão. Estas colunas correspondem a todas as colunas utilizadas durante a resolução do problema continuo.

Para garantir a factibilidade utilizando este procedimento também é necessário incluir colunas que representam os padrões de corte homogêneos unitários, ou seja, padrões de corte que possuem apenas 1 unidade de cada tipo de item. Havendo necessidade, esses padrões são utilizados para completar a demanda restante após a utilização dos padrões mais atrativos.

Utilizando este procedimento, soluções inteiras para instâncias de pequenas dimensões são obtidas em tempos computacionais aceitáveis e podem apresentar boa qualidade. No entanto, para problemas de dimensões maiores é natural que o tempo de resolução aumente devido ao aumento do número de colunas, sendo inviável considerar todas as colunas utilizadas durante a geração de colunas. Para contornar esta dificuldade, na obtenção de soluções inteiras foram utilizados os padrões de corte unitários homogêneos, os padrões de corte pertencentes à solução ótima contínua e alguns padrões de corte selecionados durante a resolução do problema. Esta seleção é realizada de acordo com a ordem de obtenção dos padrões na geração de colunas. A cada padrão de corte selecionado um conjunto de padrões de corte de tamanho igual é excluído. A quantidade de padrões de corte a ser excluída é determinada de forma que o número final de padrões de corte sem exclusão corresponda a um número pré-determinado. Com esta forma de seleção é possível garantir que padrões de todo o conjunto gerado seja selecionado.

Mesmo com a redução no número de colunas, a resolução do problema inteiro pode demorar muito para garantir a otimalidade. Para estes casos, o tempo de resolução do problema inteiro foi limitado.

Capítulo 4

TESTES COMPUTACIONAIS

Neste capítulo são apresentados os resultados da implementação das quatro estratégias descritas no Capítulo 3 e a solução da geração de colunas quando apenas uma coluna é inserida a cada iteração (SCG). Os resultados estão divididos em três seções. A Seção 4.1 apresenta os critérios utilizados no geração de instâncias e a Seção 4.2 compara as quatro estratégias de solução e a SCG quando a condição de integralidade do modelo (2.8) - (2.14) está relaxada. Na Seção 4.3, são apresentados os resultados obtidos após a aplicação da heurística de integralidade descrita em 3.6. Por fim, na Seção 4.4, é discutido o comportamento das quatro estratégias e a SCG quando aplicadas em uma única instância.

As instâncias geradas foram baseadas no artigo de Arenales et al. (2015). Os testes foram realizados em um computador com processador Intel Core i7-5200U, com 8GB de memória RAM e sistema operacional Windows 10. Todos os algoritmos foram desenvolvidos em linguagem C++ usando a biblioteca *concert* do *solver* IBM ILOG CPLEX Optimization Studio v. 12.9.

4.1 Gerador de instâncias

Os parâmetros utilizados no gerador de instâncias desenvolvido, baseado em Arenales et al. (2015), são apresentados a seguir.

- Quantidade de tipos de objetos padronizados: 1;
- Quantidade de tipos de itens: 50;

- Quantidade de objetos em estoque: Número muito grande, suficiente para atender a demanda;
- Tamanho dos objetos padronizados: 1000;
- Tamanho dos itens: Itens Médios(M), gerados no intervalo [140, 400] e itens Grandes(G), gerados no intervalo [400, 700];
- Tamanho das sobras para estoque: 400, 500 e 600;
- Demanda dos itens: As demandas podem ser Baixa (B), Média(M) e Alta(A) e são geradas nos intervalos [1,10],[10,50] e [50,300] respectivamente;
- Quantidade de sobras que podem ser geradas: $U \in \{0,3,6,9\}$;
- Quantidade de soluções do problema da mochila: K = [1,2,4,6,8];
- $\alpha' = \alpha'' = 1$;
- Número de instâncias por classe: 50.

Em Arenales et al. (2015) foi utilizado 15 tipos de itens em estoque, o parâmetro U foi definido como $U \in \{0,3,6,9,12\}$ e foi permitido utilizar sobras para atender a demanda. O restante dos parâmetros é igual para esta dissertação.

Neste trabalho, os testes permitindo a geração de sobras foram suficientes para demonstrar a eficiência das estratégias de inserção de K-soluções no problema mestre, não havendo a necessidade de considerar sobras em estoque.

As classes geradas combinam a demanda dos itens e seus tamanhos. Por exemplo, a classe [MB] representa as instâncias de itens Médios com Baixa demanda. Desta forma, 6 classes de testes foram geradas e testadas para situações sem estoque inicial de sobras. A seguir são apresentados os resultados, os quais representam a média das 50 instâncias de cada uma das classes.

4.2 Solução contínua

Para comparar os diferentes valores de K, foram utilizados dois indicadores capazes de medir o esforço computacional: número médio de iterações e tempo computacional médio,

em segundos. Outros fatores como quantidade de padrões gerados e perda para diferentes valores de U também foram analisados.

4.2.1 Tempo computacional

As Tabelas 4.1 - 4.12 exibem o tempo computacional para a resolução das classes de instâncias [MB], [MM], [MA], [GB], [GM] e [GA] para valores de U $\in \{0,3,6,9\}$. Nestas tabelas a primeira coluna apresenta a estratégia utilizada, enquanto as demais colunas representam o tempo médio computacional obtido em segundos para cada valor de K. Valores em negrito representam os menores tempos para cada um dos valores de K.

Tabela 4.1: Classe [MB]: tempo computacional médio(s) para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	5.32	4.42	4.34	4.40	11.80	9.84	9.54	9.40
K-Diversificação	39.10	34.34	35.68	37.64	47.86	41.82	44.16	44.84
K-*Branch and Bound*	**0.22**	**0.26**	**0.28**	**0.24**	**0.36**	**0.34**	**0.38**	**0.32**
K-Incumbentes	46.06	46.42	45.50	44.52	53.30	52.82	53.84	55.02
SCG	7.28				17.46			

Tabela 4.2: Classe [MB]: tempo computacional médio (s) para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-Solve	11.40	9.44	9.30	9.24	10.98	9.14	9.02	9.04
K-Diversificação	47.52	40.74	41.12	41.24	44.56	38.64	40.10	39.74
K-*Branch and Bound*	**0.34**	**0.30**	**0.32**	**0.24**	**0.40**	**0.28**	**0.32**	**0.34**
K-Incumbentes	50.88	49.66	52.48	52.76	47.14	46.58	46.66	49.24
SCG	17.38				17.42			

Tabela 4.3: Classe [MM]: tempo computacional médio (s) para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	5.34	4.54	4.62	4.38	12.04	11.10	10.46	10.42
K-Diversificação	38.18	31.86	34.30	34.76	48.98	42.30	41.12	44.78
K-*Branch and Bound*	**0.38**	**0.26**	**0.30**	**0.20**	**0.40**	**0.36**	**0.42**	**0.36**
K-Incumbentes	43.58	43.34	43.68	43.72	55.42	52.24	53.78	60.86
SCG	7.02				17.02			

De acordo com os valores apresentados nas tabelas (4.1) - (4.12), a estratégia K-*Branch and Bound* foi responsável pelos menores tempos de processamento nas seis classes testadas. Isso ocorre porque o algoritmo proposto em Leão, Cherri e Arenales (2014) é dedicado

Tabela 4.4: Classe [MM]: tempo computacional médio (s) para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	12.64	10.78	10.44	10.26	12.14	10.28	10.10	10.06
K-Diversificação	47.88	40.66	41.48	43.58	47.04	39.96	42.04	42.42
K-*Branch and Bound*	**0.32**	**0.44**	**0.34**	**0.38**	**0.34**	**0.34**	**0.40**	**0.38**
K-Incumbentes	54.46	52.5	53.54	55.84	54.16	51.84	52.08	57.30
SCG	16.86				17.00			

Tabela 4.5: Classe [MA]: tempo computacional médio (s) para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	5.26	4.64	4.50	4.36	14.28	11.92	11.90	12.04
K-Diversificação	34.34	28.86	32.26	32.22	47.52	40.78	41.42	43.30
K-*Branch and Bound*	**0.16**	**0.18**	**0.22**	**0.32**	**0.46**	**0.36**	**0.48**	**0.50**
K-Incumbentes	45.00	44.58	45.74	46.50	54.98	56.10	58.84	59.62
SCG	6.88				16.72			

Tabela 4.6: Classe [MA]: tempo computacional médio (s) para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	14.72	12.56	12.06	12.08	14.66	12.28	12.12	11.96
K-Diversificação	47.66	40.70	41.30	42.76	45.68	41.46	40.68	42.46
K-*Branch and Bound*	**0.44**	**0.92**	**0.54**	**0.50**	**0.92**	**0.60**	**0.62**	**0.74**
K-Incumbentes	56.16	56.44	58.64	62.56	58.50	55.38	56.76	58.82
SCG	16.70				16.60			

Tabela 4.7: Classe [GB]: tempo computacional médio (s) para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	0.48	0.50	0.56	0.58	0.90	0.98	0.96	0.94
K-Diversificação	0.54	0.28	0.26	0.24	1.02	0.56	0.46	0.42
K-*Branch and Bound*	**0.08**	**0.08**	**0.06**	**0.10**	**0.06**	**0.08**	**0.06**	**0.06**
K-Incumbentes	0.42	0.32	0.50	0.54	0.78	0.62	0.84	0.98
SCG	1.00				1.06			

Tabela 4.8: Classe [GB]: tempo computacional médio (s) para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	1.06	1.02	0.92	0.84	1.00	1.02	0.96	0.98
K-Diversificação	1.00	0.54	0.52	0.46	0.96	0.60	0.44	0.42
K-*Branch and Bound*	**0.10**	**0.12**	**0.10**	**0.10**	**0.06**	**0.10**	**0.06**	**0.04**
K-Incumbentes	0.70	0.72	0.88	0.96	0.72	0.74	0.86	0.98
SCG	1.04				1.06			

Tabela 4.9: Classe [GM]: tempo computacional médio (s) para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	0.44	0.56	0.54	0.58	1.02	0.92	1.04	0.98
K-Diversificação	0.58	0.36	0.30	0.28	1.08	0.70	0.48	0.48
K-*Branch and Bound*	**0.06**	**0.06**	**0.06**	**0.10**	**0.12**	**0.12**	**0.12**	**0.10**
K-Incumbentes	0.42	0.44	0.52	0.56	0.80	0.66	0.88	0.98
SCG	0.98				1.06			

Tabela 4.10: Classe [GM]: tempo computacional médio (s) para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	1.00	0.98	1.00	1.02	0.90	0.96	1.06	1.04
K-Diversificação	1.02	0.64	0.46	0.46	1.10	0.64	0.52	0.44
K-*Branch and Bound*	**0.10**	**0.12**	**0.08**	**0.10**	**0.12**	**0.10**	**0.10**	**0.08**
K-Incumbentes	0.78	0.68	0.80	0.96	0.78	0.68	0.80	0.96
SCG	1.14				1.04			

Tabela 4.11: Classe [GA]: tempo computacional médio (s) para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	0.52	0.60	0.50	0.56	1.02	1.04	1.06	1.00
K-Diversificação	0.64	0.32	0.26	0.26	1.12	0.58	0.54	0.44
K-*Branch and Bound*	**0.10**	**0.10**	**0.06**	**0.08**	**0.10**	**0.10**	**0.08**	**0.10**
K-Incumbentes	0.46	0.36	0.46	0.58	0.82	0.72	0.82	0.96
SCG	1.02				1.06			

Tabela 4.12: Classe [GA]: tempo computacional médio (s) para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	1.12	1.04	0.88	1.08	1.02	1.08	1.00	0.98
K-Diversificação	1.08	0.58	0.48	0.44	1.06	0.60	0.52	0.42
K-*Branch and Bound*	**0.16**	**0.10**	**0.08**	**0.08**	**0.14**	**0.12**	**0.06**	**0.06**
K-Incumbentes	0.82	0.70	0.84	0.90	0.84	0.70	0.78	0.98
SCG	1.10				1.08			

somente a resolução do problema da mochila, e desta forma consegue resolvê-lo em um tempo muito menor em relação ao CPLEX.

Nas classes de itens médios a estratégia K-Incumbentes apresentou os piores tempos para todos os valores de U comparados. Nas classes de itens grandes a estratégia SCG apresentou o pior desempenho na maioria das classes, o que evidencia o potencial das estratégias desenvolvidas.

Como esperado, em todas as classes, quando U = 0 o tempo computacional médio é bas-

tante inferior comparado a qualquer situação em que U $\neq$ 0, devido ao aumento do número de instâncias da mochila que devem ser resolvidas a cada iteração da geração de colunas.

4.2.2 Número médio de iterações

As Tabelas 4.13 - 4.24 apresentam o número médio de iterações do algoritmo de geração de colunas para as 6 classes testadas.

Tabela 4.13: Classe [MB]: número médio de iterações para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	131.04	119.58	115.16	116.26	115.60	103.04	100.50	100.44
K-Diversificação	**106.42**	**69.10**	**55.60**	**46.40**	**97.18**	**62.80**	**49.26**	**41.40**
K-*Branch and Bound*	122.18	87.22	71.94	63.38	109.74	77.94	64.52	57.04
K-Incumbentes	118.14	86.02	70.96	61.18	109.94	78.86	64.86	56.18
SCG	169.84				169.34			

Tabela 4.14: Classe [MB]: número médio de iterações para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	109.60	96.34	94.16	93.24	102.84	91.88	90.90	89.98
K-Diversificação	**91.02**	**58.16**	**46.90**	**38.44**	**85.34**	**55.30**	**43.86**	**35.78**
K-*Branch and Bound*	102.58	72.88	59.36	52.36	96.42	68.74	56.82	49.44
K-Incumbentes	103.24	75.46	62.72	53.84	98.72	73.68	60.16	52.26
SCG	169.58				169.62			

Tabela 4.15: Classe [MM]: número médio de iterações para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	127.62	114.58	112.36	111.38	120.10	107.04	104.48	103.70
K-Diversificação	**103.54**	**67.70**	**55.20**	**45.00**	**100.62**	**66.20**	**53.68**	**44.40**
K-*Branch and Bound*	117.92	85.26	71.56	62.44	115.46	83.28	69.14	61.24
K-Incumbentes	113.56	82.94	68.20	57.18	111.04	80.50	65.72	58.00
SCG	163.94				163.84			

Observando os valores das Tabelas 4.13 - 4.24, nota-se que o número de iterações diminui a medida que se aumenta o valor de K. Quanto maior o valor de K, mais discrepante é o número de iterações em relação à estratégia SCG, que precisou da maior quantidade de iterações para todos os valores de U em todas as classes.

Na classe [MB] para U = 0, a estratégia K-Diversificação apresentou reduções de 37%, 59%, 67% e 73% no número de iterações para K $\in \{2,4,6,8\}$ em relação a SCG. Para U = 3 as

Tabela 4.16: Classe [MM]: número médio de iterações para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	117.98	104.36	102.68	101.88	115.78	101.88	100.08	99.10
K-Diversificação	**98.70**	**63.90**	**52.38**	**42.94**	**96.72**	**63.44**	**51.14**	**42.32**
K-*Branch and Bound*	113.32	82.08	68.34	59.78	110.66	80.48	66.90	59.64
K-Incumbentes	109.58	79.58	64.74	57.00	108.48	79.24	63.88	55.18
SCG	163.48				163.72			

Tabela 4.17: Classe [MA]: número médio de iterações para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	127.26	114.70	112.74	112.54	121.98	107.90	106.48	105.34
K-Diversificação	**103.16**	**67.40**	**53.86**	**45.36**	**101.32**	**67.14**	**53.94**	**45.70**
K-*Branch and Bound*	116.90	85.16	70.80	62.88	116.32	84.66	69.92	62.86
K-Incumbentes	112.56	80.88	67.34	56.24	111.62	81.22	65.68	56.04
SCG	163.06				162.18			

Tabela 4.18: Classe [MA]: número médio de iterações para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	121.82	109.24	105.38	105.38	120.48	107.44	105.10	104.64
K-Diversificação	**100.84**	**66.76**	**53.48**	**44.90**	**100.30**	**66.38**	**53.58**	**45.10**
K-*Branch and Bound*	115.56	84.56	70.02	62.62	114.30	84.28	69.52	62.20
K-Incumbentes	111.38	81.24	66.48	56.88	111.32	80.40	65.04	57.24
SCG	162.12				162.08			

Tabela 4.19: Classe [GB]: número médio de iterações para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	49.86	53.82	54.10	54.10	57.34	57.34	57.34	57.34
K-Diversificação	44.36	22.72	14.76	11.50	43.72	21.88	14.68	11.40
K-*Branch and Bound*	56.38	33.62	26.36	22.64	48.52	31.44	24.88	21.60
K-Incumbentes	**28.82**	**15.10**	**12.84**	**11.04**	**28.38**	**15.02**	**12.72**	**10.78**
SCG	91.04				60.54			

Tabela 4.20: Classe [GB]: número médio de iterações para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	55.98	55.98	55.98	55.98	54.96	54.96	54.96	54.96
K-Diversificação	41.60	21.42	14.30	11.44	41.42	21.36	14.14	11.20
K-*Branch and Bound*	44.98	30.78	23.78	21.18	43.86	29.16	23.54	21.14
K-Incumbentes	**28.30**	**15.24**	**12.70**	**10.88**	**28.34**	**15.08**	**12.62**	**10.86**
SCG	61.50				56.46			

Tabela 4.21: Classe [GM]: número médio de iterações para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	51.78	55.92	56.54	56.84	60.34	60.34	60.34	60.34
K-Diversificação	45.60	23.70	14.90	11.88	45.80	23.56	15.00	11.78
K-*Branch and Bound*	58.5	35.58	27.58	23.84	55.08	34.68	27.60	23.62
K-Incumbentes	**29.00**	**15.22**	**12.82**	**11.32**	**28.80**	**15.30**	**12.82**	**11.26**
SCG	92.28				62.68			

Tabela 4.22: Classe [GM]: número médio de iterações para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	60.24	60.24	60.24	60.24	60.08	60.08	60.08	60.08
K-Diversificação	45.24	23.48	14.96	11.78	45.16	23.38	14.86	11.74
K-*Branch and Bound*	53.42	33.80	27.32	23.52	51.02	33.72	26.86	23.28
K-Incumbentes	**28.76**	**15.06**	**12.78**	**11.28**	**28.76**	**15.06**	**12.78**	**11.28**
SCG	62.84				62.96			

Tabela 4.23: Classe [GA]: número médio de iterações para U = 0 e U = 3

	U = 0				U = 3			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	51.18	55.80	55.86	56.14	60.82	60.82	60.82	60.82
K-Diversificação	44.14	22.30	15.12	11.58	44.16	22.36	15.14	11.56
K-*Branch and Bound*	58.68	35.86	27.82	23.72	58.16	35.68	27.84	23.70
K-Incumbentes	**29.90**	**15.34**	**12.46**	**11.16**	**29.98**	**15.34**	**12.48**	**11.18**
SCG	94.06				62.06			

Tabela 4.24: Classe [GA]: número médio de iterações para U = 6 e U = 9

	U = 6				U = 9			
	K = 2	K = 4	K = 6	K = 8	K = 2	K = 4	K = 6	K = 8
K-*Solve*	60.82	60.82	60.82	60.82	60.80	60.80	60.80	60.80
K-Diversificação	44.20	22.34	15.00	11.52	44.08	22.30	14.98	11.52
K-*Branch and Bound*	58.02	35.58	27.74	23.64	57.70	35.48	27.74	23.54
K-Incumbentes	**29.98**	**15.34**	**12.48**	**11.16**	**29.98**	**15.34**	**12.48**	**11.16**
SCG	62.08				62.10			

reduções foram de 43%, 63%, 71% e 76% em relação a SCG, o que leva a crer que quanto maior o valor de U e o valor de K, menor o número de iterações. Isto ocorre pois o aumento de U implica no aumento da diversidade dos padrões de corte, o que melhora a convergência das estratégias já que mais instâncias da mochila são resolvidas a cada iteração. No entanto, para itens grandes essa afirmação não é tão precisa. Na classe [GA] para K = 8 a estratégia K-Diversificação apresentou redução de 88% para U = 0 em relação a SCG, mas para U = $[3,6,9]$ a diferença permaneceu em 81%. Isto ocorre por 2 motivos: a geração de sobras apresenta

menos benefícios para itens grandes devido a dificuldade de combinação dos itens, e para atender altas demandas de itens seria mais benéfico permitir gerar uma maior quantidade de sobras.

Á estratégia K-Diversificação apresentou o menor número de iterações para cada um dos valores de K em todas as classes de itens médios. Nas classes de itens grandes a estratégia K-Incumbentes apresentou os menores valores, seguido pela estratégia K-Diversificação.

Um fato interessante ocorreu nas classes de itens grandes. A diferença percentual entre o número de iterações das estratégias K-Incumbentes e K-Diversificação se torna menor a medida que se aumenta o valor de K. Por exemplo, na classe [GA] com U = 6 para K = 2 a diferença percentual é de 32%. Esta diferença cai para 31%, 17% e 3% para K = 4, 6 e 8 respectivamente. Isso leva a conclusão de que quanto maior o K, maior a eficiência da estratégia K-Diversificação visto que mais padrões de corte estarão diversificados.

4.2.3 Média de padrões gerados

O número médio de padrões gerados a cada iteração durante a geração de colunas ($\overline{\text{KR}real}$), influencia diretamente no número de iterações necessárias para se atingir o ótimo. É natural presumir que quanto mais padrões inseridos, mais provável de se convergir mais rápido para a solução ótima em comparação com a SCG.

As Tabelas 4.25 e 4.26 apresentam o $\overline{\text{KR}real}$ das 5 estratégias para a classe [MM] e [GM] quando U = 9. Repare que se $\overline{\text{KR}real} = 1$, tem-se a geração de colunas padrão.

Tabela 4.25: Classe [MM]: Número médio de padrões gerados para U = 9

	K = 2	K = 4	K = 6	K = 8
K-*Solve*	2.77	3.35	3.51	3.58
K-Diversificação	4.14	7.81	10.70	13.43
K-*Branch and Bound*	4.57	8.79	12.69	16.33
K-Incumbentes	3.29	5.27	7.60	9.87
SCG	1.00	1.00	1.00	1.00

Tabela 4.26: Classe [GM]: Número médio de padrões gerados para U = 9

	K = 2	K = 4	K = 6	K = 8
K-*Solve*	1.14	1.14	1.14	1.14
K-Diversificação	2.25	4.67	7.36	10.03
K-*Branch and Bound*	2.96	5.43	7.78	10.04
K-Incumbentes	2.62	5.36	7.69	10.18
SCG	1.00	1.00	1.00	1.00

A estratégia K-*Solve* apresentou os menores valores para $\overline{\text{KR}real}$, tanto para a classe [MM] quanto para a classe [GM]. Este comportamento já era esperado devido à característica intrínseca do método de não procurar por soluções adicionais para o problema da mochila, mas sim aproveitar as já encontradas durante a busca pelo ótimo.

Ainda na estratégia K-*Solve*, $\overline{\text{KR}real}$ teve o mesmo valor para todos os valores de K da classe [GM]. Itens grandes possuem menos combinações que itens médios, o que gera uma queda abrupta no valor de $\overline{\text{KR}real}$ quando comparado com a classe [MM]. As demais estratégias também tiveram decaimento do valor de $\overline{\text{KR}real}$ entre as classes [MM] e [GM] devido a essa particularidade do número de combinações dos itens grandes.

A estratégia K-*Branch and Bound* apresentou os maiores valores de KR$real$. Esta estratégia busca K soluções, o que justifica os altos valores. As estratégias K-Incumbentes e K-Diversificação também buscam K soluções, mas apresentaram valores menores pois é possível que uma solução rejeitada não volte a ser encontrada, o que reduz o tamanho do conjunto das K soluções. Isto pode ser observado com mais clareza na Tabela 4.25 em que a diferença entre os valores de $\overline{\text{KR}real}$ é mais discrepante entre as estratégias propostas.

4.2.4 Perda média

A Tabela 4.27 apresenta a perda média para todas as classes quando não se tem sobras em estoque inicialmente. A perda é apresentada em termos de comprimento dos objetos.

Tabela 4.27: Perda média sem sobras em estoque inicial

	U=0	U=3	U = 6	U = 9
[MB]	0.41	0.24	0.18	0.17
[MM]	2.34	2.12	1.92	1.74
[MA]	0.00	0.00	0.00	0.00
[GB]	41995.50	40955.50	39987.50	39099.50
[GM]	231729	230679	229653	228627
[GA]	1374110	1373080	1372050	1371020

A redução da perda média está intimamente ligada ao aumento do número de sobras em estoque. Este comportamento já era esperado pois o aumento da quantidade de sobras provoca um aumento da diversidade dos padrões de corte, o que implica em um menor desperdício de materiais.

4.3 Solução inteira

As quatro estratégias propostas foram comparadas entre si com relação a solução inteira obtida. Foram considerados: qualidade da solução inteira e aproveitamento de padrões gerados durante a geração de colunas para a solução inteira.

A obtenção de soluções inteiras foi realizada a partir do procedimento heurístico descrito na Seção (3.6). Vale mencionar que os padrões homogêneos foram gerados da forma a = (1, 0, 0, ...,0) para garantir a obtenção de soluções factíveis com o uso da heurística. Devido a esses padrões apresentarem apenas um tipo de item em sua composição, eles podem atender pequenas demandas sem exceder a produção de outro item.

O tempo máximo para obtenção da solução inteira para cada instância foi limitado em 100 segundos. Além dos padrões de corte homogêneos e dos padrões pertencentes à base ótima, foram selecionados mais 50 padrões de corte gerados durante a resolução do problema.

4.3.1 Qualidade das soluções inteiras

As Tabelas 4.28 - 4.33 apresentam o valor da função objetivo obtida por cada estratégia. Note que o valor da função objetivo corresponde a perda do PCESA após a aplicação da heurística de integralização das variáveis de decisão.

Tabela 4.28: Classe [MB] - FO da solução inteira

	U = 0				U = 3				U = 6				U = 9			
	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8
K-Solve	4177	3977	4257	3757	4595	3903	4209	3645	3969	3693	3135	3023	3373	3135	3003	2685
K-Diversificação	**3637**	**2997**	**2257**	2737	**3799**	**3137**	**2929**	**2533**	**3331**	**2837**	**2289**	**2187**	2967	**2441**	**1859**	**1675**
K-Branch and Bound	4377	3417	3037	**2517**	4189	3793	3469	3095	3701	3257	3007	2575	3065	2683	2247	2419
K-Incumbents	3997	3477	3037	2877	4189	4041	3323	3375	3345	3003	3009	2411	**2917**	2521	2179	2157
SCG	5036				4989				4162				3669			

Tabela 4.29: Classe [MM] - FO da solução inteira

	U = 0				U = 3				U = 6				U = 9			
	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8
K-Solve	1389	1289	1409	1349	1541	1187	1039	1133	1137	921	897	809	941	1109	719	771
K-Diversificação	**1129**	**909**	**909**	**769**	1237	**655**	**593**	**409**	**875**	**573**	539	**299**	**919**	**655**	477	**341**
K-Branch and Bound	1349	1169	1029	909	1317	1373	929	969	1191	967	727	649	1249	853	735	683
K-Incumbents	1429	1169	1029	929	**1189**	871	845	677	1327	699	**525**	391	977	723	**413**	447
SCG	1809				1769				1303				1293			

Tabela 4.30: Classe [MA] - FO da solução inteira

	U = 0				U = 3				U = 6				U = 9			
	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8
K-Solve	1541	1541	1341	1241	1481	785	869	1091	1235	1069	1011	931	1073	923	833	771
K-Diversificação	**981**	**861**	**861**	781	**991**	**601**	**497**	**497**	**843**	681	**481**	**465**	**803**	**639**	**459**	431
K-Branch and Bound	1381	1101	1021	861	1143	989	595	775	935	1009	611	693	749	705	557	527
K-Incumbents	1441	1321	921	**781**	1155	723	921	603	1169	**659**	527	559	1207	695	633	**427**
SCG	1541				1685				1541				1685			

Tabela 4.31: Classe [GB] - FO da solução inteira

	U = 0				U = 3				U = 6				U = 9			
	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8
K-Solve	43036	43056	43076	43076	42060	42060	42060	42060	41066	41064	41064	41064	40284	40284	40284	40284
K-Diversificação	**42536**	42536	42376	**42256**	41456	41342	41264	**41154**	40522	40322	40286	**40162**	39564	39468	39374	**39310**
K-Branch and Bound	42556	**42336**	**42256**	42296	**41426**	**41242**	**41136**	41172	**40424**	**40312**	**40222**	40254	**39542**	**39460**	**39332**	39332
K-Incumbentes	43216	42596	42416	42336	41984	41424	41356	41280	40956	40528	40354	40326	39912	39524	39436	39364
SCG	43136				42030				41048				40182			

Tabela 4.32: Classe [GM] - FO da solução inteira

	U = 0				U = 3				U = 6				U = 9			
	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8
K-Solve	232799	232919	232899	232899	231903	231903	231903	231903	230851	230851	230851	230851	229839	229839	229839	229839
K-Diversificação	**232319**	232259	232099	**232059**	231127	231095	230961	230967	**230101**	230073	229917	229931	**229097**	229077	228909	228903
K-Branch and Bound	232399	**232059**	**232039**	232099	**231097**	**230989**	**230871**	230973	230107	**229951**	**229875**	229931	229113	**228945**	**228833**	228903
K-Incumbentes	232979	232459	232219	**232059**	231777	231305	231101	**230947**	230699	230323	230055	**229921**	229651	229315	229005	**228891**
SCG	232939				231895				232939				231895			

Tabela 4.33: Classe [GA] - FO da solução inteira

	U = 0				U = 3				U = 6				U = 9			
	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8	K=2	K=4	K=6	K=8
K-Solve	1375080	1375120	1375120	1375120	1374240	1374240	1374240	1374240	1373190	1373190	1373190	1373190	1372170	1372170	1372170	1372170
K-Diversificação	1374720	1374640	1374520	1374480	**1373620**	1373530	1373460	1373390	**1372590**	1372500	1372430	1372360	1371550	1371460	1371400	1371330
K-Branch and Bound	1374840	**1374520**	**1374460**	1374440	1373650	**1373340**	**1373340**	**1373300**	1372600	**1372310**	**1372310**	**1372290**	**1371540**	**1371300**	**1371280**	**1371250**
K-Incumbentes	**1372040**	1374780	1374560	**1374380**	1374130	1373700	1373550	1373390	1373070	1372670	1372510	1372350	1372040	1371640	1371470	1371320
SCG	1375080				1374180				1373160				1372130			

Pelas tabelas apresentadas, pode-se observar que a estratégia K-Diversificação foi a melhor estratégia na classe de itens médios, obtendo a melhor solução inteira em 83% das instâncias. A estratégia K-Incumbentes foi a melhor em 11% das situações restantes, contra 6% da estratégia K-*Branch and Bound*. Para as classes de itens grandes a estratégia K-Diversificação tem a melhor solução inteira em apenas 22% dos testes realizados. A estratégia K-*Branch and Bound* lidera a classificação com 69%, enquanto a estratégia K-Incumbentes possui 4%. A estratégia SCG não obteve a melhor solução inteira em nenhuma das comparações realizadas.

As 4 estratégias apresentaram tempo de resolução similares para a obtenção da solução inteira. A divergência entre o tempo computacional não é significativa e por isso foi omitida dos resultados. Na classe de itens grandes, por exemplo, os problemas foram resolvidos quase que instantaneamente, e as flutuações de tempo entre as estratégias não são relevantes para serem analisadas.

As Tabelas (4.28) - (4.33) mostram a diminuição no valor da perda a medida que se aumenta o valor de U. A perda média das 4 estratégias propostas para cada valor de K ($\overline{FO}$) apresentou reduções de 28,21%, 23,89% e 22,85% de U = 0 para U = 9 nas classes [MA], [MB] e [MM] respectivamente.

Embora seja esperado, vale ressaltar que aumentar o valor de U não significa necessariamente reduzir a perda para a solução inteira. Por exemplo, a SCG teve perda na classe [MA] de 1540 para U = 0 e 1684 para U = 3. Isso ocorreu devido a limitação do tempo imposto para a resolução dos problemas.

4.3.2 Aproveitamento dos padrões gerados

Os padrões de corte gerados durante a geração de colunas foram selecionados para serem utilizados na resolução do problema inteiro conforme a heurística descrita na Seção 3.6.

A Tabela 4.34 apresenta a relação em porcentagem entre o número de padrões gerados durante a geração de colunas e quantos desses padrões foram utilizados na solução inteira. Por uma questão de brevidade apenas a classe [MM] para U = 9 foi representada, entretanto, o mesmo comportamento das soluções é verificado para as demais classes.

A estratégia SCG apresentou a maior utilização dos padrões gerados durante a geração de colunas, com 21%. A estratégia K-*Solve* foi a segunda estratégia com as melhores taxas de

Tabela 4.34: Classe [MM] - Aproveitamento dos padrões para U = 9

	K = 2	K = 4	K = 6	K = 8
K-*Solve*	**0.18**	**0.17**	**0.16**	**0.16**
K-Diversificação	0.14	0.11	0.10	0.10
K-*Branch and Bound*	0.11	0.08	0.07	0.06
K-Incumbentes	0.16	0.14	0.12	0.11
SCG	**0.21**			

aproveitamentos dos padrões de corte para todos os valores de K.

A heurística para obtenção da solução inteira tem influência no resultado apresentado nesta seção. Caso o número de padrões de corte permitidos para a solução contínua fosse maior, o aproveitamento de padrões poderia ser mais elevado para as demais estratégias.

4.4 Discussões

De acordo com as soluções apresentadas na Seção (4.2), as quatro estratégias de solução propostas obtêm a solução ótima para a relaxação linear do modelo matemático independente do valor de K. No entanto, o caminho percorrido em relação à solução ótima do PCESA é diferente para cada uma das 4 estratégias propostas.

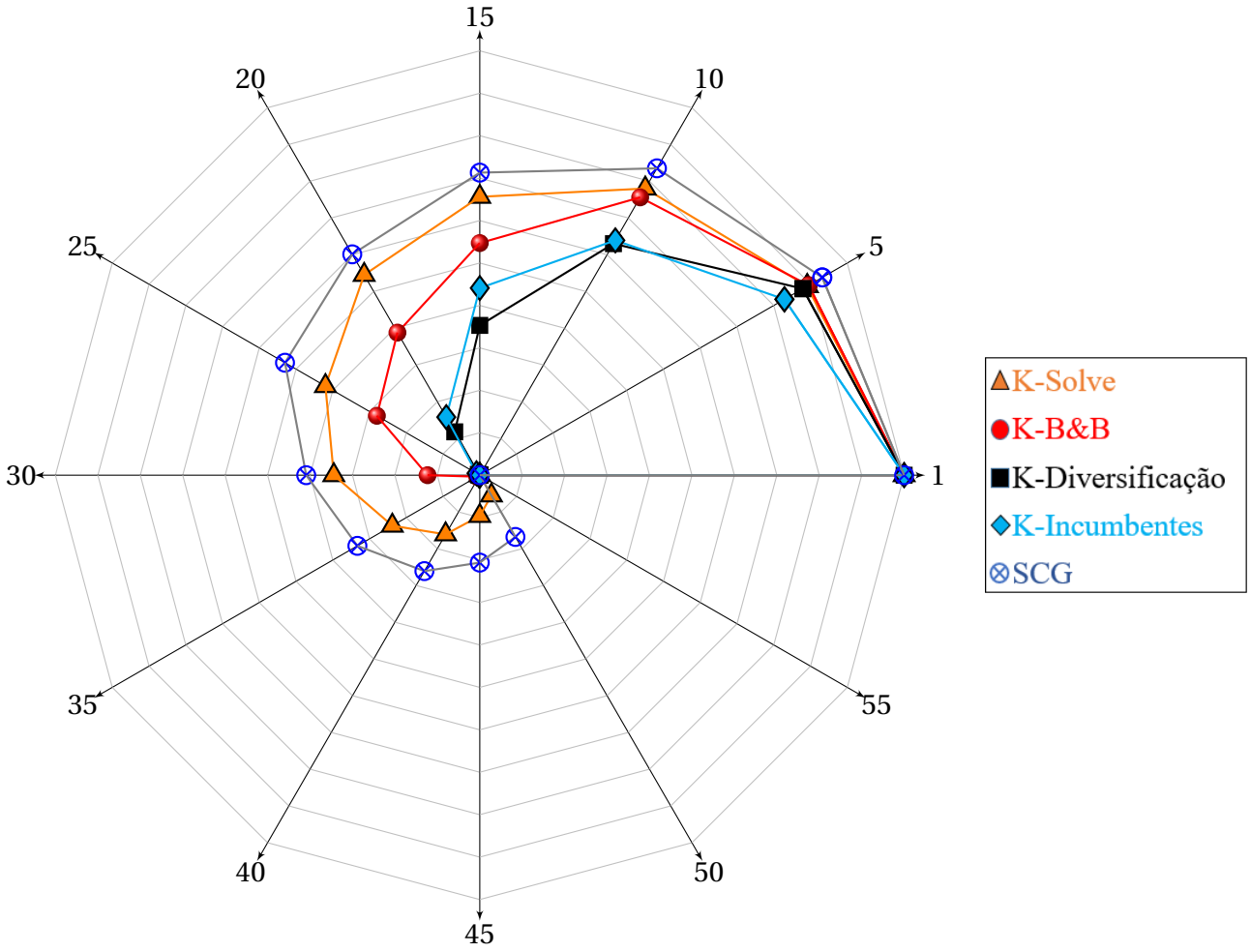
Com os testes realizados, foi possível observar que o potencial de convergência da estratégia K-Diversificação superou as demais estratégias em grande parte dos experimentos na obtenção da solução ótima para o PCESA, mostrando que diversificar os padrões de corte pode impulsionar a convergência da geração de colunas.

Na Figura 4.1, a comparação entre a convergência das 4 estratégias e a SCG é apresentada para a obtenção da solução contínua de uma instância aleatória da classe [MM] quando K = 8 e U = 9. O parâmetro $p = 0,4$ foi utilizado na estratégia K-Diversificação.

Na Figura 4.1 o centro do gráfico representa o ponto ótimo da instância analisada. Quanto mais afastado do centro, maior o valor da função objetivo. Os números em volta da Figura representam o número de iterações até aquele momento. As estratégias iniciam juntas a resolução da instância na iteração 1 e caminham em direção a solução ótima do problema relaxado no centro do gráfico. As iterações superiores a 55 foram omitidas para uma melhor visualização dos dados. Na figura, a estratégia K-*Branch and Bound* foi abreviada para K-B&B.

A estratégia K-Diversificação apresentou a melhor taxa de convergência, com 41 itera-

Figura 4.1: Convergência das 5 estratégias



ções necessárias para se atingir o ótimo, contra 62, 74, 98 e 174 das estratégias K-*Branch and Bound*, K-Incumbentes, K-*Solve* e SCG, respectivamente. Note que as estratégias K-*Solve* e SCG permaneceram desde o início da otimização sendo as mais afastadas em relação ao centro da Figura 4.1.

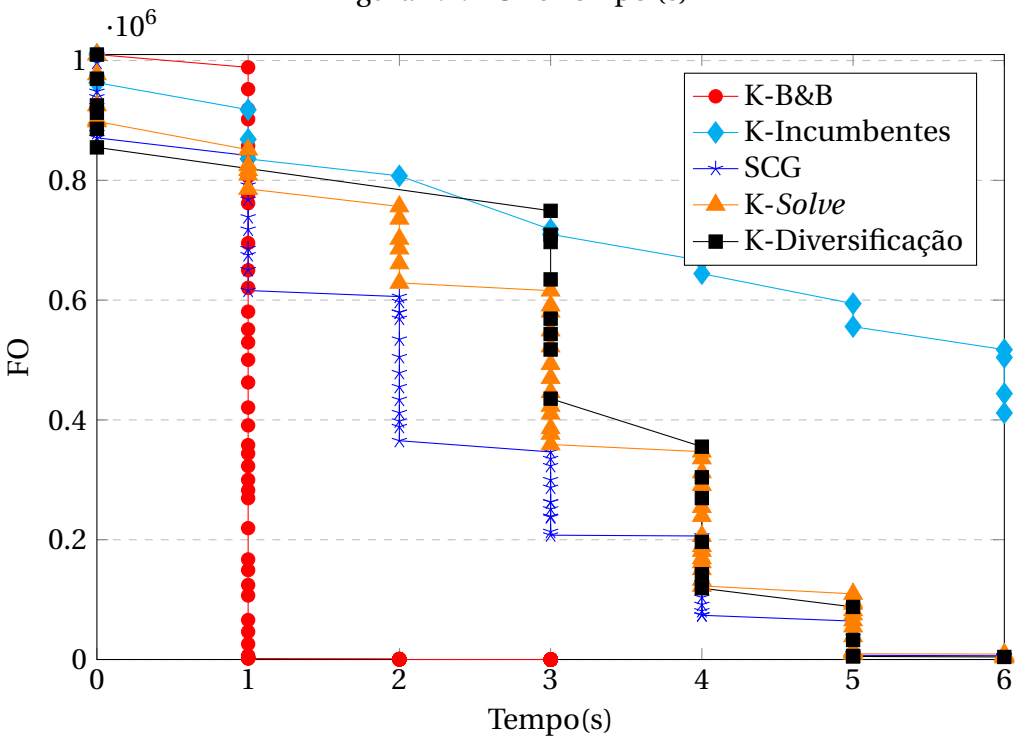
A estratégia K-Incumbentes foi responsável pelo menor valor da função objetivo até a iteração de número 8. Após isso a estratégia K-Diversificação a supera e caminha mais rápido em relação ao ótimo. Isto reforça o comportamento do exemplo apresentado na Seção 3.5. Mesmo que outra estratégia encontre bons padrões de corte iniciais e caminhe mais rápido para o ótimo, a longo prazo a estratégia K-Diversificação tem maior potencial de convergência.

A comparação entre as estratégias K-*Solve* e SCG permite concluir que é mais favorável fornecer múltiplas soluções a cada iteração ao problema mestre ao invés de uma única solução a cada iteração. A estratégia K-Diversificação evidencia a teoria de que diversificar as

soluções do problema da mochila a cada iteração é mais promissor em relação a fornecer as K soluções do problema da mochila para o problema mestre considerando apenas os custos relativos.

A Figura 4.1 mostra a relação entre o valor da função objetivo (FO) em relação ao tempo de processamento (Tempo) por iteração até atingir a solução ótima contínua para a mesma instância. A estratégia K-*Branch and Bound* foi novamente abreviada para K-B&B.

Figura 4.2: FO vs Tempo (s)



De acordo com a Figura 4.2, pode-se observar que a estratégia K-*Branch and Bound* foi a mais rápida para atingir o ótimo, terminando a otimização com apenas 3 segundos. As estratégias K-*Solve*, SCG, K-Diversificação e K-Incumbentes demoraram 11, 20 e 31 e 53 segundos respectivamente.

A estratégia K-Diversificação foi capaz de resolver múltiplas iterações da geração de colunas em menos de um segundo, conforme pode ser visto em Tempo = 0s, 6 iterações foram resolvidas. No entanto, em alguns momentos esta estratégia ficou por um longo período na mesma iteração (entre 0s e 3s a estratégia apenas uma iteração foi realizada). Este comportamento ocorre devido a característica do *callback Incumbent* utilizado nesta estratégia.

O *callback Incumbent*, também utilizado na estratégia K-Incumbentes, tem a finalidade de rejeitar soluções. Entretanto, mesmo que uma solução seja rejeitada, ela ainda pode ser encontrada inúmeras vezes pelo CPLEX devido a heurísticas do próprio *solver* ou devido a

relaxação de um nó, que resulta em uma solução inteira. Para que uma solução não seja encontrada repetidamente, é possível desativar as heurísticas ou inserir uma *Lazy constraint* que exclua a solução do conjunto solução. Esta estratégia não foi utilizada, pois, desativar as heurísticas do CPLEX influenciaria na performance destas estratégias em relação as demais e ainda não garantiria que a solução não fosse encontrada após ser rejeitada. Além disso, para restringir especificamente uma solução da mochila seria necessário o uso de cortes combinatoriais, o que implica na adição de diversas restrições para restringir todas as soluções desejadas.

Por padrão, o CPLEX possui um mecanismo chamado *presolve*. O *presolve* diminui o tamanho do problema original (por exemplo o problema da mochila) e, consequentemente, reduz o número de soluções a serem analisadas. O *presolve* não exclui a solução ótima do problema da mochila, porém, pode excluir algumas soluções incumbentes subótimas que façam parte do conjunto de K soluções desejado. Isto reforça que a estratégia K-*Branch and Bound* tenha um $\overline{\text{KR}eal}$ superior em relação as demais estratégias, uma vez que ela utiliza um *Branch and Bound* especializado para resolver o problema da mochila, e não o próprio CPLEX. Desativar o *presolve* também resultaria em uma menor eficiência do *solver*.

Vale citar que as estratégias K-*Branch and Bound* e K-Incumbentes podem obter as K-melhores soluções para o problema da mochila, mas sem a garantia de que isso vá ocorrer. Na estratégia K-Incumbentes essa perda da garantia é causada pelo *presolve* e a função de rejeição da *Incumbent callback*. Na K-*Branch and Bound* são encontradas as K soluções para o problema da mochila considerando apenas itens de coeficientes positivos da função objetivo. O vetor *price* fornecido pelo problema mestre possui itens de coeficientes negativos quando não é vantajoso realizar a produção deste item. Isto não reflete na obtenção da melhor solução para o problema da mochila.

Vale ressaltar que a solução ótima do modelo matemático 2.8 - 2.14 corresponde a uma solução inteira, sendo necessário o uso de heurísticas após a obtenção da soluçao contínua.

Capítulo 5

CONCLUSÕES

Nesta dissertação foram propostas quatro estratégias para resolver o problema de corte de estoque com sobras aproveitáveis (PCESA), utilizando o algoritmo de geração de colunas. Não foram encontradas aplicações destas estratégias em instâncias do problema de corte. A geração de colunas é um método iterativo no qual a cada iteração novas colunas são geradas com a finalidade de encontrar a melhor solução para a relaxação linear do problema. Esta técnica utiliza o problema da mochila para gerar as colunas que serão inseridas a cada iteração no problema de corte. As estratégias propostas também podem ser aplicadas em outros problemas que façam uso desse algoritmo. A formulação matemática utilizada para o PCESA neste trabalho foi proposta por Arenales et al. (2015).

As abordagens propostas consistem em inserir múltiplos padrões de corte no problema mestre a cada iteração do algoritmo de geração com o intuito de aproximar mais rapidamente da otimalidade. Para medir o desempenho das estratégias propostas, foram analisados a quantidade média de iterações e o tempo médio computacional de resolução em classes de instâncias geradas aleatoriamente.

Basicamente as estratégias consistem em obter K soluções do problema da mochila por diferentes procedimentos. Uma estratégia faz o uso do *Branch and Bound* para resolver o problema da mochila, enquanto em outra estratégia as K soluções são obtidas pelo próprio *solver* utilizado, sem nenhum esforço adicional. As duas estratégias restantes fazem o uso de funções especiais do *solver* CPLEX, chamadas de *callbacks*.

Os resultados demonstraram que o uso de múltiplas soluções do problema da mochila para inserção no PCESA diminui o tempo computacional e o número de iterações quando comparado com a geração de colunas padrão em que apenas uma coluna é inserida no pro-

blema mestre a cada iteração. Também foi possível verificar que diversificar as múltiplas soluções do problema da mochila pode acelerar o caminho em relação à solução contínua. Padrões de corte diversificados beneficiam o problema mestre, pois atendem múltiplas demandas simultaneamente. As estratégias propostas também superaram o método tradicional de geração de colunas após a aplicação da heurística de arredondamento na obtenção de soluções inteiras.

Para estudos futuros, propõe-se aprimorar estratégias de diversificação de soluções, uma vez que esta se mostrou mais promissora. Outra possibilidade é verificar se a aplicação das estratégias propostas em outros problemas que utilizam a técnica de geração de colunas provoca alterações no tempo de solução e na velocidade de convergência.

Referências Bibliográficas

ABUABARA, A.; MORABITO, R. Cutting optimization of structural tubes to build agricultural light aircrafts. *Annals of Operations Research*, 169, n. 1, p. 149–165, 2009.

ARENALES, M. N.; CHERRI, A. C.; NASCIMENTO, D. N. d.; VIANNA, A. A new mathematical model for the cutting stock/leftover problem. *Pesquisa Operacional*, SciELO Brasil, v. 35, n. 3, p. 509–522, 2015.

BARRETO, A.; NASCIMENTO, D.; CHERRI, L.; CHERRI, A. Cutting stock problem with usable leftovers: an approach using k-best solutions. 2018.

CHERRI, A. C.; ARENALES, M. N.; YANASSE, H. H. The one-dimensional cutting stock problem with usable leftover–a heuristic approach. *European Journal of Operational Research*, Elsevier, v. 196, n. 3, p. 897–908, 2009.

CHERRI, A. C.; ARENALES, M. N.; YANASSE, H. H. The usable leftover one-dimensional cutting stock problem—a priority-in-use heuristic. *International Transactions in Operational Research*, Wiley Online Library, v. 20, n. 2, p. 189–199, 2013.

CHERRI, A. C.; ARENALES, M. N.; YANASSE, H. H.; POLDI, K. C.; VIANNA, A. C. G. The one-dimensional cutting stock problem with usable leftovers - a survey. *European Journal of Operational Research*, v. 236, n. 2, p. 395–402, 2014.

CHU, C.; ANTONIO, J. Approximation algorithms to solve real-life multicriteria cutting stock problems. *Operations Research*, INFORMS, v. 47, n. 4, p. 495–508, 1999.

COELHO, K. R.; CHERRI, A. C.; BAPTISTA, E. C.; JABBOUR, C. J. C.; SOLER, E. M. Sustainable operations: The cutting stock problem with usable leftovers from a sustainable perspective. *Journal of Cleaner Production*, Elsevier, v. 167, p. 545–552, 2017.

CUI, Y.; YANG, Y. A heuristic for the one-dimensional cutting stock problem with usable leftover. *European Journal of Operational Research*, Elsevier, v. 204, n. 2, p. 245–250, 2010.

CUI, Y.; ZHONG, C.; YAO, Y. Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost. *European Journal of Operational Research*, Elsevier, v. 243, n. 2, p. 540–546, 2015.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting-stock problem. *Operations research*, v. 9, n. 6, p. 849–859, 1961.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problem — part II. *Operations research*, v. 11, n. 6, p. 863–888, 1963.

GONDZIO, J.; GONZÁLEZ-BREVIS, P.; MUNARI, P. New developments in the primal–dual column generation technique. *European Journal of Operational Research*, Elsevier, v. 224, n. 1, p. 41–51, 2013.

GRADIŠAR, M.; JESENKO, J.; RESINOVIČ, G. Optimization of roll cutting in clothing industry. *Computers & Operations Research*, Elsevier, v. 24, n. 10, p. 945–953, 1997.

HAESSLER, R. W. Controlling cutting pattern changes in one-dimensional trim problems. *Operations Research*, INFORMS, v. 23, n. 3, p. 483–493, 1975.

HAESSLER, R. W. A note on computational modifications to the gilmore-gomory cutting stock algorithm. *Operations Research*, INFORMS, v. 28, n. 4, p. 1001–1005, 1980.

HINXMAN, A. The trim-loss and assortment problems: A survey. *European Journal of Operational Research*, Elsevier, v. 5, n. 1, p. 8–18, 1980.

JAHROMI, M. H.; TAVAKKOLI-MOGHADDAM, R.; MAKUI, A.; SHAMSI, A. Solving an one-dimensional cutting stock problem by simulated annealing and tabu search. *Journal of Industrial Engineering International*, Springer, v. 8, n. 1, p. 24, 2012.

KANTOROVICH, L. V. Mathematical methods of organizing and planning production. *Management Science*, Informs, v. 6, n. 4, p. 366–422, 1960.

LEãO, A. A.; CHERRI, L. H.; ARENALES, M. N. Determining the k-best solutions of knapsack problems. *Computers & Operations Research*, Elsevier, v. 49, p. 71–82, 2014.

LONGHI, A. L. Modelos matemáticos para o problema integrado de dimensionamento de lotes e corte de estoque undimensional. Universidade Estadual Paulista (UNESP), 2013.

LONGHI, A. L.; MELEGA, G. M.; ARAUJO, S. A. d. Modelos matemÁticos para o problema integrado de dimensionamento de lotes e corte de estoque unidimensional. *Pesquisa Operacional para o Desenvolvimento*, v. 7, n. 1, p. 82–104, 2014.

LÜBBECKE, M. E.; DESROSIERS, J. Selected topics in column generation. *Operations research*, INFORMS, v. 53, n. 6, p. 1007–1023, 2005.

POLDI, K. C.; ARENALES, M. N. Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths. *Computers & Operations Research*, Elsevier, v. 36, n. 6, p. 2074–2081, 2009.

ROODMAN, G. M. Near-optimal solutions to one-dimensional cutting stock problems. *Computers & operations research*, Elsevier, v. 13, n. 6, p. 713–719, 1986.

SCHEITHAUER, G. A note on handling residual lengths. *Optimization*, Taylor & Francis, v. 22, n. 3, p. 461–466, 1991.

SINUANY-STERN, Z.; WEINER, I. The one dimensional cutting stock problem using two objectives. *Journal of the Operational Research Society*, Springer, v. 45, n. 2, p. 231–236, 1994.

STADTLER, H. A one-dimensional cutting stock problem in the aluminium industry and its solution. *European Journal of Operational Research*, Elsevier, v. 44, n. 2, p. 209–223, 1990.

TOMAT, L.; GRADIŠAR, M. One-dimensional stock cutting: optimization of usable leftovers in consecutive orders. *Central European Journal of Operations Research*, v. 25, n. 2, p. 473–489, 2017.

TRKMAN, P.; GRADISAR, M. One-dimensional cutting stock optimization in consecutive time periods. *European Journal of Operational Research*, Elsevier, v. 179, n. 2, p. 291–301, 2007.

WÄSCHER, G.; GAU, T. Heuristics for the integer one-dimensional cutting stock problem: A computational study. *Operations-Research-Spektrum*, Springer, v. 18, n. 3, p. 131–144, 1996.

WÄSCHER, G.; HAUSSNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. *European journal of operational research*, Elsevier, v. 183, n. 3, p. 1109–1130, 2007.

YANASSE, H. H.; SOMA, N. Y.; MACULAN, N. An algorithm for determining the k-best solutions of the one-dimensional knapsack problem. *Pesquisa Operacional*, SciELO Brasil, v. 20, n. 1, p. 117–134, 2000.