Pesquisa Operacional para o Desenvolvimento
Vol. 18, e18003, p. 1-24, 2024

ISSN 1984-3534

O USO DE K-SOLUÇÕES PARA O PROBLEMA DE CORTE DE ESTOQUE COM SOBRAS APROVEITÁVEIS[1]

Arthur Medeiros Figueiredo Barreto[a] *, Adriana Cristina Cherri[a,c], Luiz Henrique Cherri[b], Douglas Nogueira do Nascimento[a]

[a]Universidade Estadual Paulista "Júlio de Mesquita Filho", UNESP, Bauru-SP, Brasil

[b]Newfoundland Capital Management, São Paulo-SP, Brasil

[c]INESC TEC, Portugal

Recebido 30/05/2023, aceito 10/05/2024

RESUMO

Este trabalho aborda o problema de corte de estoque com sobras aproveitáveis, que consiste em atender a demanda de produção de itens a partir do corte de objetos disponíveis em estoque. O planejamento da produção deve ser feito de modo a minimizar o desperdício de material considerando que sobras podem ser geradas para o estoque, não sendo contabilizadas como perda. Este trabalho estuda o impacto da inserção de K-soluções a cada iteração do método de geração de colunas, que é um método iterativo no qual novas colunas são geradas a partir da resolução do problema da mochila e inseridas no problema de corte. Testes computacionais foram realizados variando o valor de K para verificar seu impacto no número de iterações e no tempo computacional em relação à geração de colunas padrão. Os resultados demonstraram o impacto positivo dessa estratégia para a obtenção de soluções contínuas e inteiras.

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

ABSTRACT

This paper deals with the problem of cutting stock with usable leftovers, which consists of meeting the demand for the production of items by cutting objects available in stock. Production planning must minimize material waste by considering that leftovers can be generated for the stock, not being accounted as a loss. This paper studies the impact of inserting K-solutions at each iteration of the column generation method, which is an iterative method in which new columns are generated from the solution of the knapsack problem and inserted into the cutting problem. Computational tests were performed varying the value of K to verify its impact on the number of iterations and computational time compared to the standard column generation. The results demonstrated the positive influence of this strategy in obtaining continuous and integer solutions.

Keywords: Cutting stock problem with usable leftovers, K-solutions, Column generation, Knapsack problem.

* Autor para correspondência. E-mail: arthur.medeiros@unesp.br
DOI: https://doi.org/10.4322/PODes.2024.005

[1]*Todos os autores assumem a responsabilidade pelo conteúdo do artigo.*

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) 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 padrões de corte, que representam a quantidade que cada item demandado será cortado dos objetos disponíveis em estoque, e definir a frequência na qual estes padrões serão utilizados 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 dos problemas de corte foram apresentados na década de 60 por Kantorovich (1960) e Gilmore e Gomory (1961, 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.

Wäscher et al. (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.

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 (Gradišar et al., 1997), no corte de tubos (Chu e Antonio, 1999; 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 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 et al. (2000) e Leão et al. (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, optamos por utilizar o modelo matemático proposto por Arenales et al. (2015) como base, pois o método de resolução aplicado a este modelo realiza a resolução de múltiplos problemas da mochila a cada iteração do método de geração de colunas. Em Arenales et al. (2015), são criados três tipos de padrões de corte: *i)* para objetos padronizados sem a geração de sobra, *ii)* para objetos padronizados com a geração de sobra, e *iii)* para sobras aproveitáveis. Desta forma, uma ampla variedade de colunas precisa ser gerada e testada durante a resolução do problema, o que nos permite analisar com mais clareza o impacto das K-soluções na geração de colunas.

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.

As próximas seções deste trabalho estão divididas da seguinte forma: a Seção 2 apresenta uma revisão da literatura referente ao PCE e ao PCESA unidimensionais. Na Seção 3, o PCESA unidimensional é definido juntamente com a apresentação do modelo matemático proposto por Arenales et al. (2015), utilizado nesta pesquisa. A Seção 4 apresenta a base teórica da estratégia de obtenção de múltiplas soluções para o problema da mochila, bem como formas de seleção de K-soluções com critério de seleção. Na Seção 5, são apresentadas as quatro estratégias propostas para a seleção de K-soluções. Para fins didáticos, a resolução de uma pequena instância descrevendo a comparação entre as quatro estratégias é apresentada passo a passo. A heurística de arredondamento utilizada é apresenda na Seção 6. A Seção 7 contém os resultados obtidos a partir de instâncias derivadas da literatura com a aplicação das estratégias propostas. Por fim, na Seção 8 são apresentadas as conclusões e as expectativas para trabalhos futuros.

2. Revisão da Literatura

A primeira abordagem para o PCE foi proposta em 1939 por Kantorovich e posteriormente traduzida para o inglês por Kantorovich (1960). 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. Haessler (1980) modificou 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. Hinxman (1980) apresentou uma revisão sobre a literatura do PCE unidimensional. O autor aponta que o método de geração de colunas apresentado em Gilmore e Gomory (1961, 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, 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. Poldi e Arenales (2009) apresentaram heurísticas construtivas e residuais da literatura para resolver o PCE, assim como foram propostas heurísticas residuais para resolver o problema.

Jahromi et al. (2012) resolveram o PCE unidimensional através de duas metaheurísticas: busca tabu e *simulated annealing*. Os autores comparam as metaheurísticas 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 et al. (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.

Referente ao PCESA, Roodman (1986) propôs uma abordagem com o objetivo principal de minimizar a perda e objetivo secundário de 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 heurístico.

Sinuany-Stern e Weiner (1994) 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 et al. (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 et al. (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.

Cherri et al. (2009) propuseram 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 et al. (2013) modificaram as heurísticas desenvolvidas por Cherri et al. (2009) priorizando o corte de sobras disponíveis em estoque. Em sua revisão de literatura, Cherri et al. (2014) abordaram artigos que resolvem o PCESA. Nessa análise, Cherri et al. (2014) apresentaram a modelagem matemática, quando proposta, estratégias de solução adotadas, comentários sobre os resultados obtidos e propostas sobre futuros estudos envolvendo os PCESA. Observou-se, de modo geral, que as sobras são tratadas de forma implícita nos problemas.

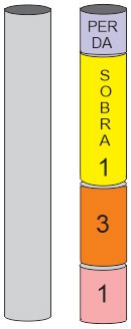
Arenales et al. (2015) propuseram um modelo matemático para o PCESA. Diferentemente dos trabalhos anteriores, as sobras foram tratadas de forma explícita, sendo possível pré-definir seus comprimentos e limitar suas quantidades quando geradas. Além disso, as sobras em estoque podem ser utilizados prioritariamente, 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 por Gilmore e Gomory (1963) .

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. Coelho et al. (2017) propuseram 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 sustentáveis.

3. Problema de Corte de Estoque Unidimensional 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 com comprimentos previamente definidos podem ser geradas em quantidades limitadas, retornando ao estoque para atenderem demandas futuras. No caso de uma abordagem que considera apenas um período, a contribuição inicial das sobras geradas é reduzir a perda da solução para o período atual, uma vez que a área das sobras aproveitáveis não é computada como perda. Em uma perspectiva multiperíodo, a possibilidade de gerar sobras para uso futuro aumenta o número de tipos de objetos em estoque, ampliando a diversidade de possíveis padrões de corte e, consequentemente, permitindo que soluções melhores sejam encontradas. A Figura 1 ilustra um padrão de corte que gera uma sobra aproveitável.

Figura 1: Padrão de corte para o PCESA unidimensional.



Fonte: Autores.

Devido às particularidades do modelo matemático proposto por Arenales et al. (2015) o mesmo foi utilizado neste trabalho para analisar o comportamento da geração de colunas quando

diferentes estratégias são utilizadas para a geração das K-soluções na resolução do subproblema. O modelo 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 $S + s$, $s \in \{1, ..., R\}$.
- e_s: número de objetos/sobras tipo s disponíveis em estoque, 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, . . . , R, $j \in J_s(k)$.

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

Sujeito a:

$$\sum_{s=1}^{S} \sum_{j \in J_s} a_{ijs} x_{js} + \sum_{s=1}^{S} \sum_{k=1}^{R} \sum_{j \in J_s(k)} a_{ijsk} x_{jsk} + \sum_{s=S+1}^{S+R} \sum_{j \in J_s} a_{ijs} x_{js} = d_i \qquad i = 1, ..., m \quad (2)$$

$$\sum_{j \in J_s} x_{js} + \sum_{k=1}^{R} \sum_{j \in J_s} x_{jsk} \leq e_s, \qquad s = 1, ..., S \quad (3)$$

$$\sum_{j \in J_s} x_{js} \leq e_s, \qquad s = S+1, ..., S+R \quad (4)$$

$$\sum_{s=1}^{S} \sum_{k=1}^{R} \sum_{j \in J_s(k)} x_{jsk} - \sum_{s=S+1}^{S+R} \sum_{j \in J_s} x_{js} \leq U - \sum_{s=S+1}^{S+R} e_s \quad (5)$$

$$x_{js} \geq 0,\ s = 1, ..., S+R, j \in J_s \text{ e inteiro.} \quad (6)$$

$$x_{jsk} \geq 0,\ k = 1, ..., R,\ s = 1, ..., S,\ j \in J_s\ (k) \text{ e inteiro.} \quad (7)$$

No modelo (1) - (7), 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) buscam atender a demanda, enquanto que as restrições (3) e (4) limitam, respectivamente, a utilização de objetos padronizados e de sobras existentes em estoque. A restrição (5) limita a quantidade máxima de sobras geradas e as restrições (6) e (7) caracterizam o domínio das variáveis.

Devido ao elevado número de variáveis e às condições de integralidade das variáveis de decisão (restrições (6) e (7)) é inviável resolver o modelo (1) - (7) 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 (1) - (7), são obtidos três tipos de padrões de corte, que cortam *i)* objetos padronizados sem a geração de sobra, *ii)* objetos padronizados com a geração de sobra, e *iii)* as sobras. 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.

4. 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 o problema da mochila é resolvido pode reduzir 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 et al. (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.

Leão et al. (2014) também desenvolveram 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.

4.1. *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).

4.2. *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 a obtenção de 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.

5. Estratégias de Solução

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.

5.1. Estratégia *K-Solve*

Nesta estratégia, a obtenção de K-soluções do problema da mochila durante a resolução do modelo (1) - (7) 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.

Esta estratégia consiste em duas etapas. Inicialmente o comando solve determina a busca pela solução ótima. Em uma segunda etapa, até K-soluções são armazenadas paralelamente ao longo da otimização. Caso uma solução com função objetivo melhor seja encontrada, esta substitui a pior solução armazenada anteriormente.

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 mostraram que as duas funções juntas geram um aumento indesejável no tempo computacional gasto para a resolução das instâncias do PCESA, e nem sempre possibilitam uma redução no tempo computacional em comparação com a geração de colunas padrão, em que $K = 1$. Devido a isto, apenas o comando *Solve* foi utilizado.

5.2. Estratégia K-Diversificação

A estratégia de diversificação consiste em duas etapas. Na primeira etapa, a solução ótima para o problema da mochila é obtida. A segunda etapa é iterativa, de forma que a cada solução encontrada uma restrição é adicionada, forçando o CPLEX a encontrar uma nova solução. Considere os itens do padrão de corte $a_i(a_1, a_2, ..., a_m)$ em que m é a quantidade de itens do problema de corte. O parâmetro Q corresponde a soma do número de itens do padrão de corte, 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 (8).

$$\sum_{i=1}^{m} a_i \leq Q * p \tag{8}$$

O somatório de todos os itens do padrão $a_i, i = 1, , , .m$ (Q) da solução ótima para o problema da mochila é armazenada em Q e multiplicada por p. Este resultado é utilizado para criar um novo problema da mochila com a restrição adicional de que o somatório dos itens seja menor que $Q * p$. Desta forma, um novo padrão de corte é obtido e o valor de Q é 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 função *callback Incumbent*. A função *callback Lazy Constraint* é utilizada para adicionar ao modelo a restrição de diversificação ao fim de cada resolução do problema da mochila.

5.3. 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 et al. (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 de acordo com $\frac{\pi_1}{l_1} \geq \frac{\pi_2}{l_2} \geq \frac{\pi_3}{l_3} \geq \ldots \geq \frac{\pi_m}{l_m}$, em que π_i é o valor de utilidade e l_i é o comprimento do item i.

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. Isto gera espaço para os demais itens na terceira etapa.

A terceira etapa compreende o aperfeiçoamento do conjunto de soluções obtidas. Se a primeira solução for um padrão de corte nulo 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 (índice do último item diferente de zero) 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.

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*.

5.4. 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.

5.5. Exemplo Numérico

A estratégia de solução K-Diversificação com parâmetro $p = 0,4$ para o problema da mochila é apresentada a seguir em um exemplo com cinco tipos de itens e dois tipos de sobras. O exemplo foi resolvido utilizando K = 2 e não foi permitido gerar padrões de corte a partir de sobras.

Os itens possuem tamanho $\{18, 27, 25, 19, 50\}$ e demanda $\{25, 17, 12, 19, 8\}$, respetivamente. Os objetos possuem 100 unidades de tamanho 100, enquanto as sobras possuem tamanho $\{40, 50\}$ com estoque de $\{3, 6\}$ unidades. Adicionalmente, é permitido gerar novas sobras de tamanho $\{40, 50\}$ até um total de 10 unidades.

A Tabela 1 exibe os padrões de corte obtidos a cada iteração para a estratégia K-Diversificação. 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 do exemplo 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.

Neste exemplo, a estratégia K-Diversificação necessitou de apenas 5 iterações para a convergência devido a diversidade dos padrões gerados, enquanto as estratégias K-*Solve*, K-*Branch and Bound* e K-Incumbentes necessitaram de 8, 7 e 5 iterações respectivamente.

O número máximo de padrões de corte a cada iteração é 6: 2 padrões de corte sem sobra, 2 padrões com sobra do tipo 1 e 2 padrões com sobra do tipo 2. Todas as estratégias apresentaram este número nas iterações 1 e 2, com exceção da estratégia K-*Solve*. Este comportamento já era

Tabela 1: 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	

Fonte: Autores.

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.

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. Isto ocorre pois enquanto as estratégias K-*Branch and Bound* e K-Diversificação inseriram apenas itens do tipo 1 e 4, a estratégia K-Incumbentes também inseriu itens do tipo 2 com os padrões $(4, 1, 0, 0, 0)$ e $(1, 1, 0, 0, 0)$. Este comportamento evidencia a teoria de que diversificar os padrões de corte provoca uma melhor convergência em relação ao ótimo. O padrão $(4, 1, 0, 0, 0)$ foi inserido na iteração 3, o que contribuiu para atingir o mesmo número de iterações em relação a estratégia K-Incumbentes.

6. Heurística de Arredondamento

As soluções inteiras para o PCESA foram obtidas utilizando a heurística de arredondamento proposta por Wäscher e Gau (1996). Este procedimento consiste em resolver o problema contínuo pelo método de geração de colunas e, em seguida, utilizar os padrões de corte gerados para resolver o problema considerando a integralidade das variáveis de decisão.

A adaptação desta heurística para o PCESA está descrita no Algoritmo 1. O algoritmo possui como dados de entrada $collumns$ e $limit\_aux$. O parâmetro $collumns$ se refere ao conjunto ordenado de todos os padrões de corte inseridos no problema mestre do problema contínuo. Esta ordenação respeita a ordem de inserção dos padrões no problema mestre através do algoritmo de geração de colunas. O parâmetro $limit\_aux$ é dividido pela quantidade de padrões de corte pertencentes ao conjunto $collumns$ e resulta em $limit$. Posteriormente, o parâmetro $limit$ é utilizado para remover padrões da conjunto $collumns$.

O algoritmo 1 permite a resolução do problema inteiro através da redução no número de padrões de corte gerados durante o algoritmo de geração de colunas. Para tal, 1 a cada $limit$ padrões de corte é removido do conjunto $collumns$ caso o padrão seja não-homogêneo e não

Algoritmo 1: Algoritmo da heurística de arredondamento

Entrada: $collumns$, $limit\_aux$;
Definir $size$ como a quantidade de padrões de corte pertencentes ao conjunto $collumns$;
Definir $nonH()$ como vetor de tamanho $size$ que especifica como 1 padrões de corte não-homogêneos;
Definir $nonContinuous()$ como vetor de tamanho $size$ que especifica como 1 os padrões de corte não-presentes na solução contínua;
$limit = (size/limit\_aux) + 1$;
Considerar as condições de integralidade para as variáveis de decisão do modelo matemático (1) - (7) ;
for $i = 1...size$ **do**
    if *nonH(i) == 1 AND i % limit == 0 AND nonContinuous(i) == 1* **then**
        remover o padrão de corte i do conjunto $collumns$;
    end
end
Resolver o problema inteiro com o conjunto $collumns$ atualizado.

tenha sido utilizado na solução contínua.

A garantia da factibilidade pelo algoritmo 1 ocorre devido ao conjunto de padrões de corte $collumns$ incluir os padrões homogêneos unitários. Desta forma, esses padrões podem ser utilizados para completar a demanda restante após a utilização dos padrões mais atrativos.

7. Testes Computacionais

Nesta seção, são descritos e apresentados os resultados da implementação das quatro estratégias propostas e a solução da geração de colunas quando apenas uma coluna é inserida a cada iteração (SCG). Os resultados estão divididos em duas subseções. A Subseção 7.1 apresenta os critérios utilizados no geração de instâncias e a Subseção 7.2 compara as quatro estratégias de solução e a SCG quando a condição de integralidade do modelo (1) - (7) está relaxada.

As instâncias geradas para os testes computacionais 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.

7.1. 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: $U \in \{3, 6, 9\}$;
- 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) foram considerados 15 tipos de itens demandados, 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. A descrição completa das instâncias geradas e os resultados obtidos para cada instância podem ser acessados a partir do link *https://github.com/arthurmedeirosfb/PODES_k_solucoes*.

7.2. 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.

7.2.1. Tempo Computacional

As Tabelas 2 e 3 exibem os valores médios do tempo computacional gasto na resolução das classes de instâncias [MB], [MM], [MA], [GB], [GM] e [GA] a partir das quatro estratégias descritas na seção anterior, com K = 2, 4, 6 e 8. Esses valores são as médias de três casos (U $\in \{3, 6, 9\}$), e valores em negrito representam os menores tempos para cada classe com cada K.

Tabela 2: Classes [MB], [MM] e [MA]: tempo médio (s) para K = 2, 4, 6 e 8.

	MB				MM				MA			
	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	11,39	9,47	9,29	9,23	12,27	10,72	10,33	10,25	14,55	12,25	12,03	12,03
K-Diversificação	46,65	40,40	41,79	41,94	47,97	40,97	41,55	43,59	46,95	40,98	41,13	42,84
K-B&B	**0,37**	**0,31**	**0,34**	**0,30**	**0,35**	**0,38**	**0,39**	**0,37**	**0,61**	**0,63**	**0,55**	**0,58**
K-Incumbentes	50,44	49,69	50,99	52,34	54,68	52,19	53,13	58,00	56,55	55,97	58,08	60,33
SCG	17,42				16,96				16,67			

Fonte: Autores.

Tabela 3: Classes [GB], [GM] e [GA]: tempo computacional médio (s) para K = 2, 4, 6 e 8.

	GB				GM				GA			
	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	0,99	1,01	0,95	0,92	0,97	0,95	1,03	1,01	1,05	1,05	0,98	1,02
K-Diversificação	0,99	0,57	0,47	0,43	1,07	0,66	0,49	0,46	1,09	0,59	0,51	0,43
K-B&B	**0,07**	**0,10**	**0,07**	**0,07**	**0,11**	**0,11**	**0,10**	**0,09**	**0,13**	**0,11**	**0,07**	**0,08**
K-Incumbentes	0,73	0,69	0,86	0,97	0,79	0,67	0,83	0,97	0,83	0,71	0,81	0,95
SCG	1,05				1,08				1,08			

Fonte: Autores.

De acordo com os dados apresentados nas Tabelas 2 e 3, a estratégia K*-Branch and Bound* foi responsável pelos menores tempos de processamento nas seis classes testadas. Isso ocorre porque o algoritmo proposto por Leão et al. (2014) é dedicado 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, enquanto para as classes de itens grandes a estratégia SCG apresentou o pior desempenho, o que evidencia o potencial das estratégias desenvolvidas.

7.2.2. Número Médio de Iterações

As Tabelas 4 e 5 apresentam o número médio de iterações, das quatro estratégias testadas, durante a resolução das seis classes de instâncias com K = 2, 4, 6 e 8. Novamente, esses são valores médios para U $\in \{3, 6, 9\}$, e os menores números de iterações para cada classe com cada K estão destacados em negrito.

Tabela 4: Classes [MB], [MM] e [MA]: número médio de iterações para K = 2, 4, 6 e 8.

	MB				MM				MA			
	$K = 2$	$K = 4$	$K = 6$	$K = 8$	$K = 2$	$K = 4$	$K = 6$	$K = 8$	$K = 2$	$K = 4$	$K = 6$	$K = 8$
K-Solve	114,49	102,60	100,07	99,83	117,95	104,43	102,41	101,56	121,43	108,19	105,65	105,12
K-Diversificação	94,26	60,85	48,79	40,21	98,68	64,51	52,40	43,22	100,82	66,76	53,67	45,23
K-B&B	107,06	76,28	62,71	55,06	113,15	81,95	68,13	60,22	115,39	84,50	69,82	62,56
K-Incumbentes	106,70	78,39	64,61	55,76	109,70	79,77	64,78	56,73	111,44	80,95	65,73	56,72
SCG	169,48				169,48				162,13			

Fonte: Autores.

Tabela 5: Classes [GB], [GM] e [GA]: número médio de iterações para K = 2, 4, 6 e 8.

	GB				GM				GA			
	$K = 2$	$K = 4$	$K = 6$	$K = 8$	$K = 2$	$K = 4$	$K = 6$	$K = 8$	$K = 2$	$K = 4$	$K = 6$	$K = 8$
K-Solve	56,09	56,09	56,09	56,09	60,22	60,22	60,22	60,22	60,81	60,81	60,81	60,81
K-Diversificação	42,25	21,55	14,37	11,35	45,40	23,47	14,94	11,77	44,15	22,33	15,04	11,53
K-B&B	45,79	30,46	24,07	21,31	53,17	34,07	27,26	23,47	57,96	35,58	27,77	23,63
K-Incumbentes	28,34	15,11	12,68	10,84	28,77	15,14	12,79	11,27	29,98	15,34	12,48	11,17
SCG	59,50				62,83				62,08			

Fonte: Autores.

Observando os valores das Tabelas 4 e 5, nota-se que o número de iterações tende a diminuir na medida em que K aumenta. 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 todas as classes. Na classe [MB], por exemplo, a estratégia K-Diversificação apresentou reduções de 44%, 64%, 71% e 76% no número de iterações para $K \in \{2, 4, 6, 8\}$ em relação a SCG. Na classe [GA] para $K = 8$, a estratégia K-Diversificação apresentou redução de 81% em relação a SCG

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] 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.

7.2.3. Média de Padrões Gerados

Um outro fator de análise das soluções ótimas contínuas obtidas por cada estratégia proposta é o número médio de padrões gerados a cada iteração durante a geração de colunas, aqui denotado por $\overline{KReal}$. Esse parâmetro se diferencia de K na medida em que, a cada iteração do método de geração de colunas, são gerados K padrões de corte para objetos padronizados sem sobra e K padrões de corte para objetos padronizados com sobra. O número total de padrões de corte gerados

para todos esses casos compõe o valor de $\overline{KReal}$. No entanto, apenas os K melhores padrões são inseridos no problema mestre.

O aumento no valor de $\overline{KReal}$ traz benefícios para o problema mestre pois um conjunto de K padrões pode ser escolhido entre uma gama maior de padrões de corte. As Tabelas 6 e 7 apresentam o $\overline{KReal}$ das 5 estratégias para a classe [MM] e [GM] quando U = 9, respectivamente. Repare que se $\overline{KReal} = 1$, tem-se a geração de colunas padrão.

Tabela 6: 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

Fonte: Autores.

Tabela 7: 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

Fonte: Autores.

A estratégia K-*Solve* apresentou os menores valores para $\overline{KReal}$, 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*, $\overline{KReal}$ 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 $\overline{KReal}$ quando comparado com a classe [MM]. As demais estratégias também tiveram decaimento do valor de $\overline{KReal}$ 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 $\overline{KReal}$. 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 na Tabela 6 em que para $K = 8$, a estratégia K-*Branch and Bound* apresenta $\overline{KReal}$ superior em 21% em relação a K-Diversificação e 65% em relação à estratégia K-Incumbentes.

7.2.4. Perda Média

A Tabela 8 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.

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.

Tabela 8: Perda média sem sobras em estoque inicial.

	U=3	U = 6	U = 9
[MB]	0,24	0,18	0,17
[MM]	2,12	1,92	1,74
[MA]	0,00	0,00	0,00
[GB]	40955,50	39987,50	39099,50
[GM]	230679	229653	228627
[GA]	1373080	1372050	1371020

Fonte: Autores.

7.3. Solução Inteira

Os testes computacionais para a obtenção de soluções inteiras a partir da heurística de arredondamento descrita na Seção 6 foram realizados com as mesmas instâncias utilizadas nos experimentos referentes às soluções contínuas. Também foram mantidos os valores de K = 2, 4, 6 e 8, e $U \in \{0, 3, 6, 9\}$. A Tabela 9 apresenta os valores médios da função objetivo (Coluna "FO") e da porcentagem de perda (Coluna "Perda (%)") entre todos os cenários referentes ao parâmetro U para as classes de instâncias com itens médios. Nesses testes, o parâmetro $limit\_aux$ da heurística de arredondamento foi definido como 50.

Tabela 9: Classes [MB], [MM] e [MA]: solução inteira.

	MB							
	K = 2		K = 4		K = 6		K = 8	
	FO	Perda (%)	FO	Perda (%)	FO	Perda (%)	FO	Perda (%)
K-Solve	4028,14	5,47	3676,64	5,03	3650,64	5,00	3277,14	4,51
K-Diversificação	3433,14	4,72	2852,64	3,95	2333,14	3,26	2282,64	3,20
K-B&B	3832,64	5,22	3287,14	4,51	2939,64	4,07	2651,14	3,67
K-Incumbentes	3611,64	4,95	3260,14	4,49	2886,64	4,01	2704,64	3,76
SCG	FO = 4464,23 , Perda (%) = 6,02							
	MM							
	K = 2		K = 4		K = 6		K = 8	
	FO	Perda (%)	FO	Perda (%)	FO	Perda (%)	FO	Perda (%)
K-Solve	1252,08	0,32	1126,58	0,29	1016,08	0,26	1015,58	0,26
K-Diversificação	1040,08	0,26	698,08	0,18	629,58	0,16	454,58	0,12
K-B&B	1276,58	0,32	1090,58	0,28	855,08	0,22	802,58	0,20
K-Incumbentes	1230,58	0,31	865,58	0,22	703,08	0,18	611,08	0,16
SCG	FO = 1543,58 , Perda (%) = 0,38							
	MA							
	K = 2		K = 4		K = 6		K = 8	
	FO	Perda (%)	FO	Perda (%)	FO	Perda (%)	FO	Perda (%)
K-Solve	1332,10	0,06	1079,10	0,05	1013,10	0,04	1008,10	0,04
K-Diversificação	904,10	0,04	695,10	0,03	574,10	0,02	543,10	0,02
K-B&B	1051,60	0,05	950,60	0,04	695,60	0,03	713,60	0,03
K-Incumbentes	1242,60	0,05	849,10	0,04	750,10	0,03	592,10	0,03
SCG	FO = 1529,60 , Perda (%) = 0,06							

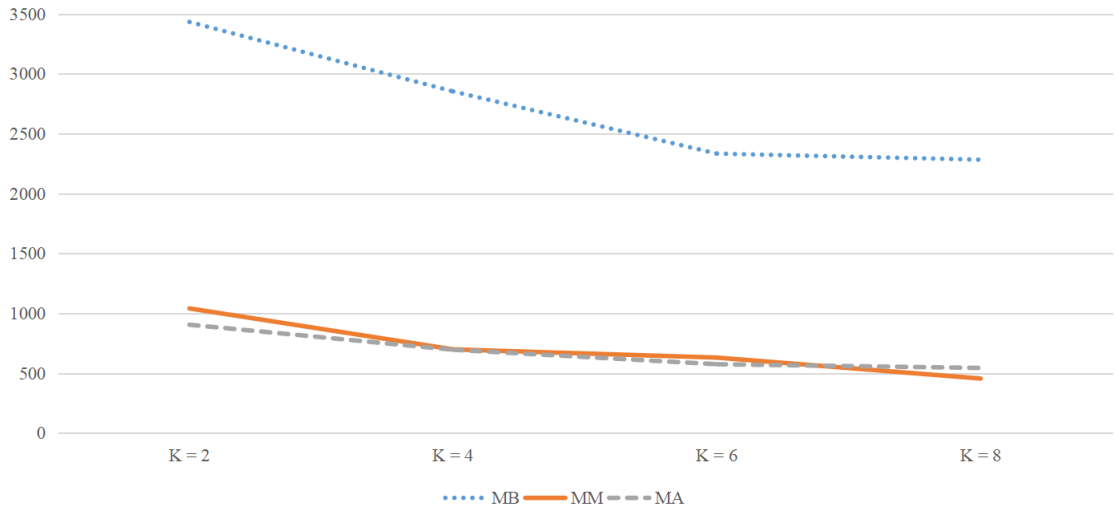
Fonte: Autores.

Como visto na Tabela 8, as soluções contínuas para as instâncias com itens médios apresentaram perdas muito próximas de zero, sendo que, para a classe MA, não houve perda. Em um primeiro momento, isso pode causar uma distorção na comparação com os resultados na Tabela 9, podendo indicar que as soluções inteiras obtidas pela heurística de arredondamento são de baixa qualidade, uma vez que os valores da função objetivo variaram entre 900 e 4028. Entretanto, ao analisarmos a porcentagem da perda que esses valores representam, é possível notar que a perda gerada se mantém em patamares consideravelmente baixos, não passando de 0,06% para a classe MA, 0,32% para a classe MM e 5,47% para a classe MB.

Na análise do impacto dos diferentes valores de K, é visto que os valores da função objetivo e da porcentagem de perda diminuem em todos os casos para as classes MB e MM quanto maior

o número de K-soluções consideradas. Apenas para a classe MA essa redução não é significativa, uma vez que a perda gerada pela estratégia SCG (K = 1) já está em um patamar bastante baixo (0,06%). O impacto dos diferentes valores de K na qualidade das soluções inteiras pode ser visto com mais clareza no gráfico da Figura 2, que mostra os valores da função objetivo para cada K referentes às soluções obtidas pela estratégia K-Diversificação, que teve o melhor desempenho entre as estratégias propostas.

Figura 2: Redução no valor da função objetivo das soluções inteiras para instâncias com itens médios (Estratégia K-Diversificação).



Fonte: Autores.

Os mesmos testes foram realizados para as classes de instâncias com itens grandes, e os resultados estão apresentados na Tabela 10. Nessa tabela, são mostrados os valores médios da porcentagem de perda (Coluna "Perda (%)") e o gap entre os valores da função objetivo para as soluções ótimas contínuas e os valores da função objetivo para as soluções inteiras (Coluna "Gap (%)"). Esse gap é calculado da seguinte forma: $gap = [(sol^I - sol^C)/sol^C] \times 100$, em que sol^C é o valor da função objetivo para a solução contínua e sol^I é o valor da função objetivo para a solução inteira.

Diferente das soluções para as instâncias com itens médios, a porcentagem de perda para as instâncias com itens grandes atingiu valores mais altos, em torno de 22%, tanto para as soluções contínuas quanto para as soluções inteiras, visto que o gap entre elas não passou de 3%. Isso se deve ao menor número de possíveis padrões de corte existentes quando os itens possuem comprimentos maiores, entre 40% e 70% do comprimento dos objetos padronizados. Consequentemente, o aumento no número de K-soluções também não contribuiu para melhorar a qualidade das soluções inteiras, como pode ser observado no gráfico da Figura 3.

7.4. Discussões

De acordo com as soluções apresentadas na Seção 7.2, 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.

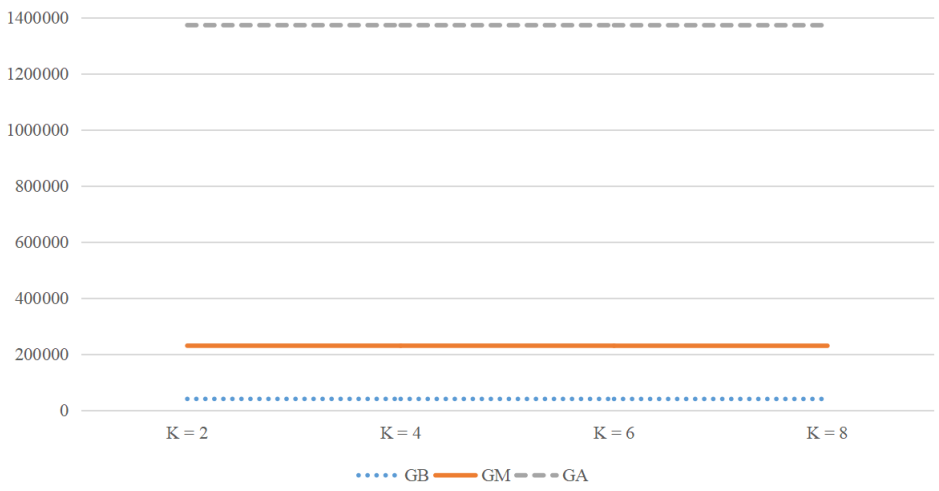
Com os testes realizados, foi possível observar que o potencial de convergência da estratégia K-Diversificação superou as demais estratégias em grande parte dos experimentos na obtenção da solução ótima para o PCESA, mostrando que diversificar os padrões de corte pode impulsionar a convergência da geração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

Tabela 10: Classes [GB], [GM] e [GA]: solução inteira.

	GB							
	$K = 2$		$K = 4$		$K = 6$		$K = 8$	
	Perda (%)	Gap (%)	Perda (%)	Gap (%)	Perda (%)	Gap (%)	Perda (%)	Gap (%)
K-Solve	22,48	2,72	22,49	2,73	22,49	2,75	22,49	2,75
K-Diversificação	22,22	1,26	22,18	1,00	22,13	0,78	22,09	0,52
K-B&B	22,21	1,18	22,15	0,81	22,10	0,56	22,11	0,63
K-Incumbentes	22,44	2,48	22,22	1,25	22,16	0,94	22,14	0,78
SCG	Perda (%) = 22,48 , Gap (%) = 2,69							
	GM							
	$K = 2$		$K = 4$		$K = 6$		$K = 8$	
	Perda (%)	Gap (%)	Perda (%)	Gap (%)	Perda (%)	Gap (%)	Perda (%)	Gap (%)
K-Solve	21,75	0,51	21,75	0,52	21,75	0,52	21,75	0,52
K-Diversificação	21,69	0,21	21,69	0,20	21,68	0,13	21,68	0,13
K-B&B	21,70	0,22	21,68	0,14	21,67	0,10	21,68	0,13
K-Incumbentes	21,74	0,48	21,71	0,29	21,69	0,18	21,68	0,12
SCG	Perda (%) = 21,74 , Gap (%) = 0,52							
	GA							
	$K = 2$		$K = 4$		$K = 6$		$K = 8$	
	Perda (%)	Gap (%)	Perda (%)	Gap (%)	Perda (%)	Gap (%)	Perda (%)	Gap (%)
K-Solve	22,01	0,08	22,02	0,08	22,02	0,08	22,02	0,08
K-Diversificação	22,01	0,04	22,01	0,03	22,01	0,03	22,01	0,02
K-B&B	22,01	0,04	22,01	0,02	22,01	0,02	22,01	0,02
K-Incumbentes	22,02	0,09	22,01	0,05	22,01	0,03	22,01	0,02
SCG	Perda (%) = 22,01 , Gap (%) = 0,08							

Fonte: Autores.

Figura 3: Redução no valor da função objetivo das soluções inteiras para instâncias com itens grandes (Estratégia K-Diversificação).



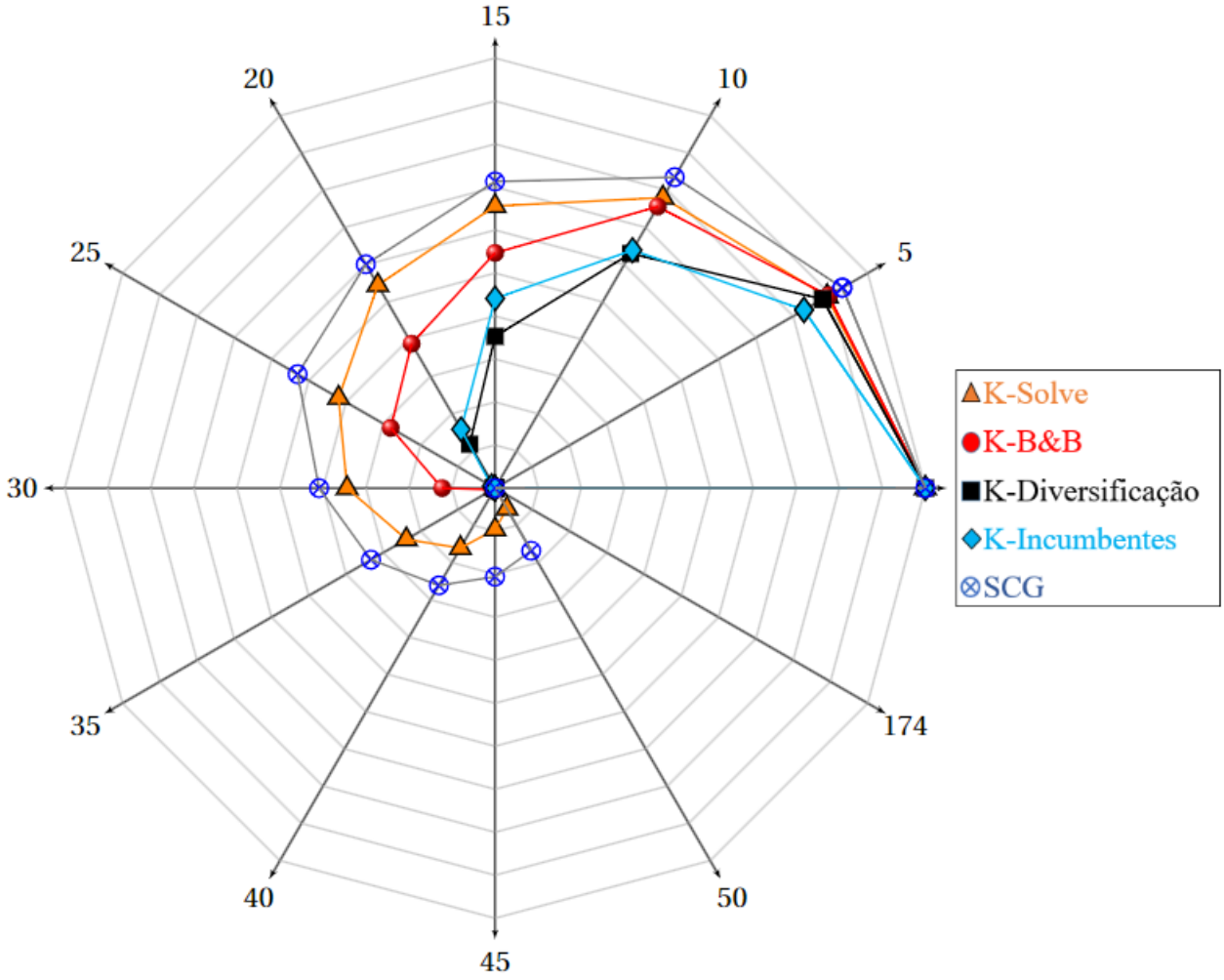
Fonte: Autores.

Na Figura 4, 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 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 iterações necessárias para se atingir o ótimo, contra 62, 74, 98 e 174 das estratégias K-*Branch and Bound*,

Figura 4: Convergência das 5 estratégias.



Fonte: Autores.

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.

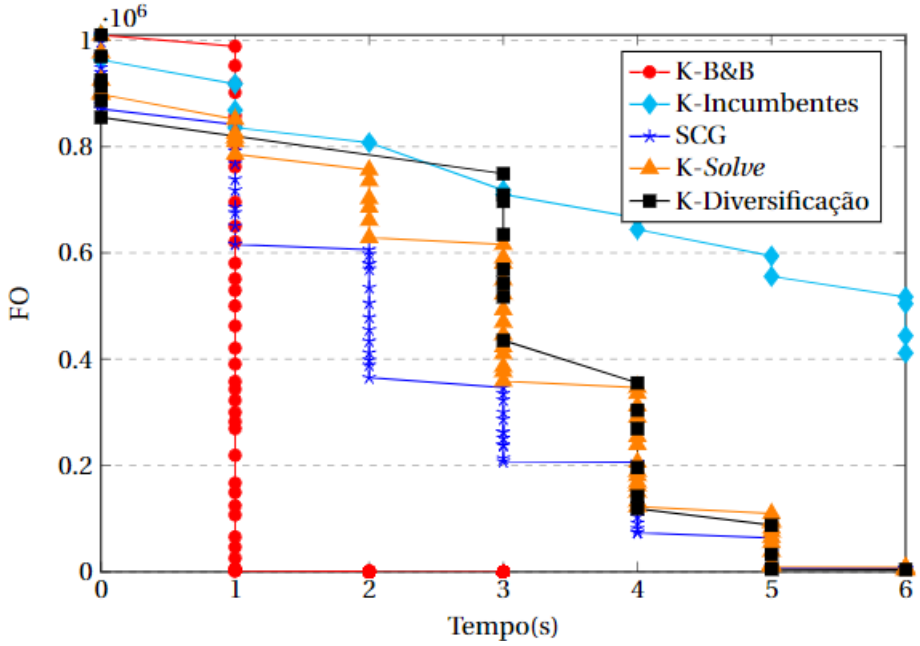
A estratégia K-Incumbentes foi responsável pelo menor valor da função objetivo até a iteração de número 8. Após isso a estratégia K-Diversificação a supera e caminha mais rápido em relação ao ótimo. Isto reforça o comportamento do exemplo apresentado na Seção 5.5. Mesmo que outra estratégia encontre 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 5 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.

De acordo com a Figura 5, 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

Figura 5: FO *vs* Tempo (s).



Fonte: Autores.

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 $\overline{KReal}$ 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.

8. Conclusão

Neste trabalho foram propostas quatro estratégias para resolver o problema de corte de estoque com sobras aproveitáveis (PCESA) utilizando o algoritmo de geração de colunas. Esta técnica utiliza o problema da mochila para gerar as colunas que serão inseridas a cada iteração no problema de corte. Essas estratégias também podem ser aplicadas em outros problemas que façam uso desse algoritmo. A formulação matemática utilizada neste trabalho para resolver o PCESA foi apresentada em 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, foram analisados o número de iterações e o tempo computacional de resolução para classes de instâncias geradas aleatoriamente. Uma heurística de arredondamento também foi utilizada para obter soluções inteiras a partir dos padrões de corte geradas por cada estratégia.

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 problema 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.

Com relação aos resultados dos experimentos com soluções inteiras, foi possível obter soluções de boa qualidade (baixa porcentagem de perda) para todas as classes de instâncias com itens médios. Além disso, aumentar o número de K-soluções contribuiu consideravelmente na diminuição da perda para essas instâncias. Entretanto, para instâncias com itens grandes, as soluções inteiras ficaram muito próximas das soluções ótimas contínuas, com uma perda média gerada de 22%.

Para estudos futuros, propõe-se a aplicação das estratégias propostas, e de outras estratégias de diversificação de soluções, para a resolução de instâncias maiores e/ou instâncias reais de problemas de corte com sobras aproveitáveis. Outra possibilidade, é verificar se a aplicação dessas estratégias em outros problemas que utilizam a técnica de geração de colunas contribui para a redução do tempo de resolução e da velocidade de convergência do método.

Agradecimentos. Os autores agradecem à CAPES (Código 001) e CAPES-PrInt-UNESP número de processo 88887.310463/2018-00, ao CNPq números de processo 317460/2021-8 e 402240/2023-5 e a FAPESP número de processo 2022/05803-3.

Referências

Abuabara, A. e Morabito, R. Cutting optimization of structural tubes to build agricultural light aircrafts. *Annals of Operations Research*, v. 169, n. 1, p. 149–165, 2009.

Arenales, M. N., Cherri, A. C., Nascimento, D. N. d. e Vianna, A. A new mathematical model for the cutting stock/leftover problem. *Pesquisa Operacional*, v. 35, n. 3, p. 509–522, 2015.

Cherri, A. C., Arenales, M. N. e Yanasse, H. H. The one-dimensional cutting stock problem with

usable leftover–a heuristic approach. *European Journal of Operational Research*, v. 196, n. 3, p. 897–908, 2009.

Cherri, A. C., Arenales, M. N. e Yanasse, H. H. The usable leftover one-dimensional cutting stock problem—a priority-in-use heuristic. *International Transactions in Operational Research*, v. 20, n. 2, p. 189–199, 2013.

Cherri, A. C., Arenales, M. N., Yanasse, H. H., Poldi, K. C. e Goncalves Vianna, A. C. 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. e Antonio, J. Approximation algorithms to solve real-life multicriteria cutting stock problems. *Operations Research*, v. 47, n. 4, p. 495–508, 1999.

Coelho, K. R., Cherri, A. C., Baptista, E. C., Jabbour, C. J. C. e Soler, E. M. Sustainable operations: The cutting stock problem with usable leftovers from a sustainable perspective. *Journal of Cleaner Production*, v. 167, p. 545–552, 2017.

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

Cui, Y., Zhong, C. e Yao, Y. Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost. *European Journal of Operational Research*, v. 243, n. 2, p. 540–546, 2015.

Gilmore, P. C. e 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. e Gomory, R. E. A linear programming approach to the cutting stock problem — part II. *Operations Research*, v. 11, n. 6, p. 863–888, 1963.

Gradišar, M., Jesenko, J. e Resinovič, G. Optimization of roll cutting in clothing industry. *Computers & Operations Research*, v. 24, n. 10, p. 945–953, 1997.

Haessler, R. W. Controlling cutting pattern changes in one-dimensional trim problems. *Operations Research*, 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*, v. 28, n. 4, p. 1001–1005, 1980.

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

Jahromi, M. H., Tavakkoli-Moghaddam, R., Makui, A. e Shamsi, A. Solving an one-dimensional cutting stock problem by simulated annealing and tabu search. *Journal of Industrial Engineering International*, v. 8, n. 1, p. 24, 2012.

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

Leão, A. A., Cherri, L. H. e Arenales, M. N. Determining the k-best solutions of knapsack problems. *Computers & Operations Research*, v. 49, p. 71–82, 2014.

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

Roodman, G. M. Near-optimal solutions to one-dimensional cutting stock problems. *Computers & Operations Research*, v. 13, n. 6, p. 713–719, 1986.

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

Sinuany-Stern, Z. e Weiner, I. The one dimensional cutting stock problem using two objectives. *Journal of the Operational Research Society*, 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*, v. 44, n. 2, p. 209–223, 1990.

Tomat, L. e 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. e Gradisar, M. One-dimensional cutting stock optimization in consecutive time periods. *European Journal of Operational Research*, v. 179, n. 2, p. 291–301, 2007.

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

Wäscher, G., Haußner, H. e Schumann, H. An improved typology of cutting and packing problems. *European Journal of Operational Research*, v. 183, n. 3, p. 1109–1130, 2007.

Yanasse, H. H., Soma, N. Y. e Maculan, N. An algorithm for determining the k-best solutions of the one-dimensional knapsack problem. *Pesquisa Operacional*, v. 20, n. 1, p. 117–134, 2000.