Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 10, n. 1, 2023.

Trabalho apresentado no XLII CNMAC, Universidade Federal de Mato Grosso do Sul - Bonito - MS, 2023

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Algoritmo Genético Adaptativo com Chaves Aleatórias Viciadas para um Problema de Corte de Estoque Multi-Período com Custos de Setup

Silvio A. de Araujo[1], Eduardo M. Silva[2]
IBILCE/UNESP, São José do Rio Preto, SP
Raf Jans[3]
HEC-Montreal, Quebec, Canadá

Resumo. Este trabalho apresenta um Algoritmo Genético Adaptativo com Chaves Aleatórias Viciadas (AGACAV) para resolver o problema de Corte de Estoque Multi-período com custos de *setup* nos padrões de corte. Uma aplicação determinística chamada *decoder* que mapeia soluções factíveis do problema é necessária para a inicialização do AGACAV. Um *decoder* baseado na geração de estoque por período e construção de padrões de corte é proposto e comparado com um procedimento de geração de colunas. Os métodos foram comparados em instâncias com diferentes tamanhos de itens e o resultados mostram que o AGACAV obtém melhores resultados para instâncias cujo custo de setup é maior que o custo dos objetos em estoque.

Palavras-chave. Problema de Corte de Estoque Unidimensional, Custo de *Setup* nos Padrões de Corte, Metaheurística.

1 Introdução

O Problema de Corte de Estoque (PCE) vem sendo estudado, dentre outros motivos, devido a sua importância em diversas aplicações de origem industrial, tais como em indústrias de metal, madeira, vidro e papel. O PCE consiste em determinar maneiras de cortar um conjunto de objetos, em pedaços menores (itens), visando satisfazer demandas de clientes ([2]). Um dos objetivos primários do PCE é minimizar o custo do material envolvido, no entanto, custos auxiliares surgem durante as etapas do processo de corte, um desses custos refere-se a quantidade de padrões de corte usados no plano de corte, por exemplo, o número de vezes que deve-se mudar a posição das lâminas para realizar um novo corte ([14]). Tais ajustes interrompem o processo de produção aplicando custos de *setup* como objetivo extra para o problema. O PCE que tem como objetivo minimizar a quantidade de padrões de cortes é conhecido como Problema da Minimização de Padrões e é classificado como um problema NP-*hard* ([10]).

O problema de corte de estoque multi-período (PCEMP) consiste em resolver, para cada período em um horizonte de planejamento, um PCE a fim de satisfazer a demandas em cada período. No entanto, nesta configuração, as demandas de alguns itens podem ser adiantadas ou não na forma de inventário para os próximos períodos ([12]). Uma revisão sobre o PCEMP pode ser encontrada em [11]. A partir de agora, o PCEMP com custos de *setups* será denotado por PCEMPs.

Devido a importância dos problemas de natureza combinatória do ponto de vista prático e teórico, o interesse em estudar abordagens exatas e heurísticas para esses problemas vem crescendo.

[1]silvio.araujo@unesp.br
[2]machado.silva@unesp.br
[3]raf.jans@hec.ca

O tempo computacional de algoritmos exatos aumenta exponencialmente com o tamanho das instâncias, e por isso o uso de abordagens heurísticas ou metaheurísticas para resolver grandes instâncias se torna quase inevitável. Recentemente, em [3], uma extensão do Algoritmo Genético de Chaves Aleatórias (AGCA), que foi inicialmente proposto por [1] e estendida em [8], é proposta. A versão dos autores lida com a "calibração" de parâmetros necessários para a inicialização de algoritmos evolutivos em geral e propõe uma configuração em que os parâmetros usados no AG são adaptativos. Este trabalho tem como objetivo propor uma versão do Algoritmo Genético Adaptativo de Chaves Aleatórias Viciadas (AGACAV), proposto por [3], para resolver o PCEMPs. A metaheuristica é testada em um conjunto de instâncias cujo tamanho dos itens é variado e então comparada com um procedimento heurístico baseado na geração de colunas de [5, 6] e resolvido por um *software* de otimização.

O restante do artigo se organiza da seguinte forma: na Seção 2, o problema é definido e um modelo matemático baseado na formulação de [5, 6] é apresentado. Na Seção 3, os métodos de solução baseados na geração de colunas e no trabalho de [3] são discutidos. Na Seção 4, os resultados computacionais e uma comparação dos métodos é apresentada. Por fim, na Seção 5, são apresentadas as conclusões e propostas futuras.

2 Definição do Problema e Modelo Matemático

Para formular matematicamente o PCEMPs supõe-se um conjunto T representando os períodos no horizonte de planejamento finito e um conjunto P dos itens cujas demandas são definidas *a priori*. Também assumimos que apenas um tipo objeto em estoque de tamanho L está disponível em quantidade ilimitada. Em cada período $t \in T$, itens do tipo $i \in P$ de tamanho l_i tem que ser cortado a fim de satisfazer sua demanda d_{it}. Sem perda de generalidade, assumimos que $l_i \leq L$, $l_i > 1$ e é inteiro para todo i. Definimos então o padrão de corte como uma maneira de obter itens menores de um objeto de tamanho L. Definimos o conjunto J como o conjunto de padrões de corte disponíveis. Toda vez que um padrão de corte diferente é usado, um custo de *setup* é imposto. Neste trabalho, não são consideradas restrições de capacidade e tempo de *setup*.

Também são definidas as seguintes notações para modelar o problema: $|M_t|$ indica a quantidade de padrões de corte necessária para satisfazer a demanda dos períodos t até $|T|$, a_{ijt} indica o número de vezes que o item i é cortado no padrão de corte j no período t. Para cada período t, todo padrão de corte $j \in J$ pode ser descrito pelo vetor $A_j = (a_{1jt}, a_{2jt}, ..., a_{|P|jt})^T$. O padrão de corte é factível se $a_{1jt}l_1 + a_{2jt}l_2 + ... + a_{|P|jt}l_{|P|} \leq L$.

As variáveis de decisão do problema são: o número de objetos cortados usando o padrão de corte j no período t (x_{jt}), a variável binária que vale 1 quando o padrão de corte j é usado no período t e 0 caso contrário (y_{jt}) e a quantidade de estoque do item i no período t (s_{it}).

A seguir a formulação Adaptada de Gilmore e Gomory (AGG) é apresentada. Esta formulação é a mais utilizada na literatura do PCEMP (veja artigos de [11]) e é dada por:

Modelo AGG

$$\textbf{Min} \sum_{t\in T}\sum_{i\in P} h_{it}s_{it} + \sum_{t\in T}\sum_{j\in J}(c_t y_{jt} + c x_{jt}) \tag{1}$$

s.a.

$$x_{jt} \leq |M_t| y_{jt} \qquad \forall j,\ \forall t \tag{2}$$

$$s_{i,t-1} + \sum_{j\in J} a_{ijt}x_{jt} = d_{it} + s_{it} \qquad \forall i,\ \forall t \tag{3}$$

$$x_{jt} \in \mathbb{Z}^+,\ y_{jt} \in \{0,1\} \qquad \forall j,\ \forall t \tag{4}$$

$$s_{it} \geq 0 \qquad \forall i,\ \forall t \tag{5}$$

DOI: 10.5540/03.2023.010.01.0017

A função objetivo (1) minimiza os custos de estoque dos itens, os custos de *setup* dos padrões de corte e os custo de cortar os objetos. As restrições (2) forçam y_{jt} a ser 1 caso algum objeto seja cortado de acordo com o padrão de corte j no período t, i.e., $x_{jt} > 0$. As restrições (3) representam a demanda e o balanceamento de estoque dos itens por período. Por fim, as condições (4) e (5) indicam o domínio das variáveis do modelo.

3 Métodos de Solução

Nesta seção são apresentados os métodos de solução aplicados ao PCEMPs. Primeiramente apresentamos o método de [3] e em seguida, um procedimento baseado na geração de colunas de [5, 6] é apresentado.

3.1 Algoritmo Genético Adaptativo com Chaves Aleatórias Viciadas (AGACAV)

O AGACAV é uma metaheurística baseada no Algoritmo Genético com Chaves Aleatórias Viciadas (AGCAV) de [8] e no Algoritmo Genético com Chaves Aleatórias (AGCA) de [1]. No AGCA, os indivíduos (também chamados de cromossomos) são representados como vetores compostos por números do intervalo $[0, 1]$ gerados aleatoriamente. Classificado como um algoritmo determinístico, o *decoder* associa a cada vetor uma solução factível do problema estudado.

No AGACAV, os parâmetros do algoritmo em comum com o AGCAV (dados pela Tabela 1) são atualizados de forma determinística durante as gerações do método. Além disso, o AGACAV introduz dois novos parâmetros, α e β, a fim de restringir a população de melhores indivíduos e introduzir técnicas de perturbação para soluções com *fitness* similares, respetivamente. O usuário precisa apenas fixar um valor para λ, $\lambda \in [0.999, 0.998, 0.997]$, que corresponde ao tempo computacional gasto pelo método.

Tabela 1: Valores recomendados para os Parâmetros. Fonte: [7].

Parâmetro	Descrição	Valor recomendado
p	tamanho da população	$p = an$ onde $1 \leq a \in \mathbb{R}$ é constante e n é a quantidade de genes do cromossomo
p_e	tamanho da população elite	$0.1p \leq p_e \leq 0.25p$
p_m	tamanho da população mutante	$0.1p \leq p_m \leq 0.3p$
ρ_e	probabilidade de *crossover*	$0.5 \leq \rho_e \leq 0.8$

O esquema de implementação do AGACAV é mostrado na Figura 1. Nota-se uma clara separação entre as componentes que dependem do problema (o passo de aplicar o *decoder*) e as componentes que independem do mesmo (demais passos).

DOI: 10.5540/03.2023.010.01.0017

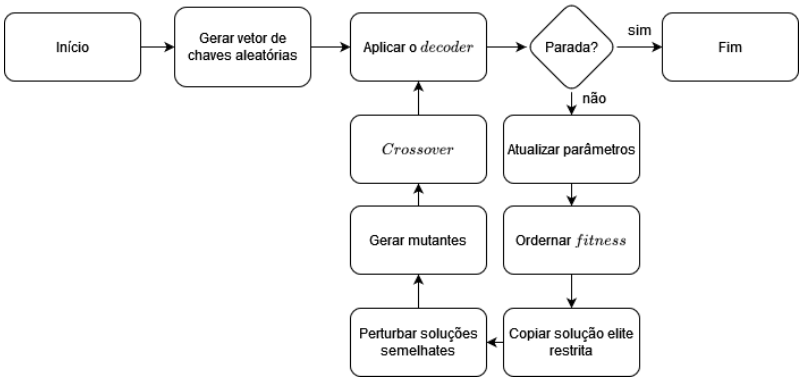

Figura 1: *Framework* do AGACAV. Fonte: autor.

3.1.1 Um *decoder* para o PCEMPs

O *decoder* proposto é baseado no trabalho de [9], onde os vetores de chaves aleatórias são múltiplos da quantidade de ordens do problema, no caso do PCEMPs, os vetores serão múltiplos de $|P||T|$. Essas parcelas são divididas em duas partes onde a primeira é chamada de Fase 1 (com $|P|(|T|-1)$ chaves aleatórias) e a segunda é a Fase 2 (com $2|P||T|$ chaves aleatórias). Dado um vetor de chaves aleatórias χ, usaremos a notação it, $1it$ e $2it$ para denotar as posições no vetor referentes aos itens i no período t na Fase 1 e nas duas parcelas da Fase 2, respectivamente.

Na Fase 1 de um período t, a quantidade de produção referente a cada item $i \in P$, X_{it}, é obtida como a soma das demandas do item em questão do período t até o período τ_{it}, tal que:

$$\tau_{it} = t + fint(\chi(it)(|T|-t)), \quad \forall i, \forall t \in [1, ..., |T|-1], \tag{6}$$

em que $fint$ é a função que arredonda para o inteiro mais próximo.

Na Fase 2 de um período t temos duas parcelas de chaves aleatórias. A primeira parcela vai fornecer para cada item i o valor W_{it}, indicando a quantidade com que o item i será inserido em algum padrão de corte no período t.

$$W_{it} = \lceil \lfloor L/l_i \rfloor \chi(1it) \rceil, \quad \forall i, \forall t \in T, \tag{7}$$

a segunda parcela gera o vetor RK_t, dado por:

$$RK_t = (\chi(21t), \chi(22t), ..., \chi(2|P|t)), \tag{8}$$

o vetor RK_t é então ordenado de forma decrescente aos valores das chaves aleatórias e a sequência obtida dita a ordem com que os itens serão usados.

Dado um período t, as informações obtidas pelas fases 1 e 2 são combinadas para gerar padrões de corte da seguinte maneira: os vetores W_t e RK_t são usados para obter o vetor de comprimentos $[l_{i_1}W_{i_1t}, W_{i_2}W_{i_2t}, ..., l_{i_{|P|}}W_{i_{|P|t}}]^T$, então, começando por $l_{i_1}W_{i_1t}$, os comprimentos são somados até que não haja mais itens que caibam no objeto sem exceder sua capacidade L, por exemplo, se $l_{i_1}W_{i_1t} \leq L$ e $l_{i_1}W_{i_1t} + l_{i_2}W_{i_2t} > L$, então o item i_2 não será considerado no padrão de corte e uma nova frquência factível W_{i_2t} é considerada, caso tal valor não exista, o algoritmo irá para o próximo item na sequência até que todos os itens sejam visitados e testados. Quando não há mais espaço no objeto ou items disponíveis, o padrão de corte é formado e os itens cortados são marcados como indisponíveis, assim que todos os itens do período se tornam indisponíveis o procedimento é interrompido. Salientamos que esse procedimento não designa o mesmo item em diferentes padrões de corte. De forma geral, a geração dos padrões de corte e seu cálculo de frequência difere de acordo com os períodos pelas seguintes regras:

- quando $t = 1$, a quantidade de vezes que o padrão de corte é usado (sua frequência) é obtida como o inteiro mais próximo à direita da maior das razões entre a demanda cumulativa e a quantidade de vezes que o item é cortado. Para um padrão de corte j no período 1, sua frequência é calculada como:

$$\max_{i \in P} \left\{ \left\lceil \frac{X_{i1}}{a_{ijt}} \right\rceil, a_{ijt} > 0 \right\}. \tag{9}$$

- para os períodos futuros, $t > 1$, para que um item i seja designado a um padrão de corte é necessário que $X_{it} - s_{it-1} > 0$, ou seja, a demanda cumulativa do item no período t não excede a quantidade do item estocada no período anterior. Desta forma, quando $t > 1$, a frequência dos padrões de corte j é calculada por:

$$\max_{i \in P} \left\{ \left\lceil \frac{X_{it} - s_{it-1}}{a_{ijt}} \right\rceil, a_{ijt} > 0 \right\}. \tag{10}$$

3.2 Procedimento de Geração de Colunas

A heurística é simples e direta. Ela consiste em resolver o problema mestre relaxado referente à formulação AGG pelo método de geração de colunas proposto por [5, 6], então o problema inteiro é resolvido.

O problema mestre relaxado é inicializado com colunas relacionadas aos padrões de corte homogêneos, i.e., colunas do tipo $(0, ..., a_{ii}, ..., 0)$, em que $a_{ii} = 1, \forall i$. Como parte da geração de colunas para problemas de corte, o subproblema é caracterizado por um problema da mochila. No PCEMP, o subproblema é resolvido para cada período t e a coluna com o menor custo reduzido é incluída ao problema mestre restrito. O subproblema de *pricing* para cada período t é dado por:

$$\textbf{Min} \quad c - \sum_{i \in P} a_i \pi_{it} \tag{11}$$

s.a.

$$a_1 l_1 + a_2 l_2 + ... + a_{|P|} l_{|P|} \leq L \tag{12}$$

$$a_i \in \mathbb{Z}^+ \qquad \forall i, \tag{13}$$

em que c é o custo associado ao objeto e π_{it} é a solução dual associada a restrição (3) referente ao item i no período t. A função objetivo (11) minimiza o custo reduzido das colunas candidatas para o problema mestre. A condição (12) representa a restrição de mochila e as restrições (13) determinam o domínio das variáveis.

O procedimento é executado até que não hajam mais colunas com custo reduzido atraente para serem introduzidas no problema em cada período.

4 Resultados Computacionais

Nesta seção os métodos descritos anteriormente são testados e comparados usando uma máquina com processador Intel Core i7 com 3.2 giga-hertz de CPU e 16 gigabyte de RAM. Os procedimentos heurísticos foram implementados pelo Visual Studio 2017 usando o IBM ILOG Cplex 22.1 como *software* de otimização. As instâncias usadas são propostas em [13] e possuem as seguintes características: custos de *setup*: $c_t = 100$ para todo t, quantidade de períodos: $|T| = \{3, 6\}$; quantidade de itens: $|P| = \{10, 20\}$; custo de estoque $h_{it} = 0.01 l_i$ e comprimento dos itens: $l_i \in \begin{cases} [0.01, 0.2]L, & \text{Pequeno;} \\ [0.01, 0.8]L & \text{Médio;} \\ [0.2, 0.8]L, & \text{Grande.} \end{cases}$. No total, foram consideradas 12 classes de 20 instâncias com demanda média de 10 itens por período.

A análise dos resultados foi feita com base nos valores do limitante inferior obtido pelo modelo *AGG*, o limitante superior (*fitness*), Z_{IP}, e o tempo até convergência dos métodos. O tempo limite foi adotado com 1800 segundos, o parâmetro λ foi considerado como 0.999 e a população inicial do AGACAV foi $p = 500$. Os resultados considerados foram os melhores dentre 15 inicializações do AGACAV. Para resolver o modelo AGG foi considerado $|M_t| = \max_{i,j,\ \overline{a}_{ijt} > 0} \left\{ \left\lceil \frac{\sum_{\tau=t}^{|T|} d_{i\tau}}{\overline{a}_{ijt}} \right\rceil \right\}$.

Na Tabela 2 são mostradas as médias dos resultados considerando cada classe com custo de objeto $c = 10$. Nota-se que o AGACAV apresenta os melhores resultados tanto para soluções inteiras quanto para tempo médio (com exceção da Classe 8 onde o AGG apresenta uma solução inteira 0.9% melhor). As melhorias das soluções do AGACAV sobre o AGG, por grupo de itens, foi a seguinte: 43.97% para o Grupo S, 2.07% para o Grupo M e 2.65% para o Grupo H. Os métodos também foram testados para os custos $c = 100$ e $c = 1000$. Quando $c = 100$, a qualidade dos *gaps* das soluções com base no limitante inferior melhora, porém as soluções inteiras do AGG são melhores que o AGACAV para algumas classes de instâncias. No geral, o AGACAV foi melhor 7.31% em valores *fitness* para o Grupo S, já o AGG foi melhor 2.47% em valores *fitness* para o Grupo M e 2.34% para o Grupo H. Em termos de tempo computacional, pouca variância em relação a Tabela 2 foi notada. Quando $c = 1000$, o AGG obteve melhores soluções inteiras em todas as classes, no geral, o AGG obteve uma média 0.63% melhor que a média do AGACAV. Para $c = 1000$, as soluções dos grupos M e H, considerando o AGG, obtiveram gaps médios de menos de 1%.

Tabela 2: Resultados computacionais considerando $c = 10$.

Classes	limitante inferior	(Z_{IP})		(T_{IP})	
$(\|T\|/\|P\|/l_i)$	Z_{LP}	$\boldsymbol{D_1}$	***AGG***	$\boldsymbol{D_1}$	***AGG***
Classe 1 (3/10/S)	613.97	892.48	1421.41	0.28	9.83
Classe 2 (3/20/S)	1121.28	1558.00	2861.51	0.69	991.26
Classe 3 (6/10/S)	1188.54	2006.11	2512.00	0.68	1689.00
Classe 4 (6/20/S)	2413.39	4083.94	5501.12	1.31	1764.78
Média (S)	**1334.29**	**2135.13**	**3074.00**	**10.89**	**1113.72**
Classe 5 (3/10/M)	2695.72	2872.69	3022.73	0.31	0.24
Classe 6 (3/20/M)	5187.74	5494.76	5762.9	1.04	88
Classe 7 (6/10/M)	5638.51	6055.15	6258.7	0.98	100.5
Classe 8 (6/20/M)	9737.59	10,865.63	10,767.5	2.12	1524.85
Média (M)	**5314.84**	**6322.06**	**6452.96**	**1.11**	**428.4**
Classe 9 (3/10/H)	3567.99	3762.85	3883.49	0.77	0.1
Classe 10 (3/20/H)	6352.66	6645.76	6888.89	5.35	14.17
Classe 11 (6/10/H)	6554.83	6918.18	7057.16	4.2	8.73
Classe 12 (6/20/H)	12,597.48	13,340.6	13,651.39	11.64	956.3
Média (H)	**7268.24**	**7666.85**	**7870.23**	**5.49**	**244.82**
Mean	**4639.12**	**5374.68**	**5799.06**	**2.45**	**595.64**

5 Considerações Finais

Neste trabalho, uma metaheurística baseada no Algoritmo Genético Adaptativo com Chaves Adaptativas Viciadas (AGACAV) é proposta para o problema de Corte de Estoque Multi-Período com Custos de *Setup* (PCEMPs). Foi proposto um *decoder* baseado na representação dos indivíduos do algoritmo como estocagem, frequências e ordenação aleatórias dos itens para cada período. O método foi então comparado, em termos de solução inteira e tempo computacional, com um procedimento de geração de colunas resolvido por *software* de otimização. Os métodos foram testados em 12 classes de instâncias diferenciadas pela quantidade de itens, quantidade de períodos e pelo tamanho dos itens. O AGACAV obtém bons resultados para custos menores de *setup* e um melhor tempo computacional para todas as instâncias quando comparado à heurística de geração de colunas.

Como proposta futura, um procedimento de busca local pode ser implementado no AGACAV. Além disso, a adaptação dos parâmetros do método pode ser melhorada usando técnicas de aprendizado de máquina como em [4].

DOI: 10.5540/03.2023.010.01.0017

Referências

[1] J. C. Bean. "Genetic Algorithms and Random Keys for Sequencing and Optimization". Em: **ORSA Journal on Computing** 6.2 (1994), pp. 154–160. DOI: 10.1287/ijoc.6.2.154.

[2] H. Ben Amor e J. M. Valério de Carvalho. "Cutting Stock Problems". English. Em: **Column Generation**. Ed. por G. Desaulniers, J. Desrosiers e M. M. Solomon. Springer US, 2005, pp. 131–161.

[3] A. Chaves, J. F. Gonçalves e L. Lorena. "Adaptive Biased Random-key Genetic Algorithm with Local Search for the Capacitated Centered Clustering Problem". Em: **Computers & Industrial Engineering** 124 (jul. de 2018), pp. 331–346. DOI: 10.1016/j.cie.2018.07.031.

[4] Antônio Augusto Chaves e L. H. N Lorena. **An adaptive and near parameter-free BRKGA using Reinforcement Learning**. Rel. técn. Universidade Federal de São Paulo (UNIFESP), 2021.

[5] P. C. Gilmore e R. E. Gomory. "A Linear Programming Approach to the Cutting Stock Problema - Part II". Em: **Operations Research** 11.6 (1963), pp. 863–888.

[6] P. C. Gilmore e R. E. Gomory. "A Linear Programming Approach to the Cutting-Stock Problem". Em: **Operations Research** 9.6 (1961), pp. 849–859.

[7] J. F. Gonçalves e M. G. C. Resende. "Random-key genetic algorithms". Em: **Handbook of heuristics** (2016), pp. 1–13.

[8] J. F. Gonçalves e M.G.C. Resende. "Biased random-key genetic algorithms for combinatorial optimization". Em: **J Heuristics** 17 (2011), pp. 487–525. DOI: 10.1007/s10732-010-9143-1.

[9] José Fernando Gonçalves e Mauricio G.C. Resende. "A biased random key genetic algorithm for 2D and 3D bin packing problems". Em: **International Journal of Production Economics** 145.2 (2013), pp. 500–510. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2013.04.019. URL: https://www.sciencedirect.com/science/article/pii/S0925527313001837.

[10] C. McDiarmid. "Pattern minimisation in cutting stock problems". Em: **Discrete Applied Mathematics** 98.1-2 (1999), pp. 121–130.

[11] G. M. Melega, de Araujo, S. A. e Jans, R. "Classification and literature review of integrated lot-sizing and cutting stock problems". Em: **European Journal of Operational Research** 271.1 (2018), pp. 1–19.

[12] K. C. Poldi e S. A. de Araujo. "Mathematical models and a heuristic method for the multiperiod one-dimensional cutting stock problem". Em: **Annals of Operations Research** 238.1 (2016), pp. 497–520.

[13] Eduardo M. Silva et al. "Formulations and theoretical analysis of the one-dimensional multi-period cutting stock problem with setup cost". Em: **European Journal of Operational Research** 304.2 (2023), pp. 443–460. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2022.04.023. URL: https://www.sciencedirect.com/science/article/pii/S0377221722003344.

[14] D. A. Wuttke e S. H. Heese. "Two-dimensional cutting stock problem with sequence dependent setup times". Em: **European Journal of Operational Research** 265.1 (2018), pp. 303–315.

