Trabalho apresentado no XLI CNMAC, Unicamp - Campinas - SP, 2022.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics
Preprint

Uma proposta de solução via Local Branching do Problema de Corte de Estoque com Data de Entrega e *Setup* de troca de Padrão de corte

Elisama de Araújo Silva Oliveira [1]
MMC/CEFET, Belo Horizonte, MG
Elisangela Martins de Sá
MMC/CEFET, Belo Horizonte, MG
Sérgio Ricardo de Souza
MMC/CEFET, Belo Horizonte, MG
Elizabeth Fialho Wanner MMC/CEFET, Belo Horizonte, MG

Resumo.

Este artigo apresenta o Problema de Corte de Estoque com Datas de entregas e *Setup* de troca de padrão de corte (PCE-DDS). Este problema é uma extensão do clássico Problema de Corte de Estoque (PCE), em que as datas de entrega são incorporadas aos pedidos e os *setups* de trocas de padrão na máquina de corte são adicionados ao tempo de conclusão do pedido. Apesar do PCE ser extensivamente estudado, não foram encontrados, na literatura, trabalhos que abordem, conjuntamente, a inclusão de datas de entrega e tempos de *setup* para trocade padrões. Este artigo propõe uma metodologia para a solução do PCE-DDS. Consideramos uma versão simples do PCE e, como hipótese, que o tempo para cortar qualquer padrão de corte é o mesmo. A metodologia proposta é dividida em três fases: na primeira fase, o problema relaxado é resolvido pelo método simplex com geração de colunas, obtendo uma solução viável fracionada; na segunda fase, a heurística *Relax-and-Fix* é aplicada às colunas encontradas na fase anterior para se encontrar uma solução inteira; na terceira fase, a matheurística *Local Branching* é usada para melhorar a solução obtida na segunda fase. Os resultados computacionais são comparados com a solução fornecida pelo *solver* CPLEX e mostram que a proposta de abordagem de resolução apresenta soluções de melhor qualidade que as fornecidas pelo *solver*.

Palavras-chave. Problema de corte, Data de Entrega, Atraso, Adiantamento, *Setup*, *Local Branching*.

1 Introdução

O problema de corte de estoque (PCE) consiste na otimização do processo de corte de unidades maiores (objetos) que estejam disponíveis para a produção de um conjunto de unidades menores (itens), com o objetivo de atender a demanda desses itens e satisfazer algum critério de otimização, como, por exemplo, minimizar a perda de material gerada pelo corte ou o custo total dos objetos cortados. As formas e as medidas dos objetos e dos itens são bem especificadas. O PCE tem uma ampla gama de aplicações industriais e tem sido estudado extensivamente nas últimas décadas. Os trabalhos de [6] e [5] são considerados os pioneiros e apresentaram uma técnica de geração de colunas para a solução, o que viabilizou a aplicação em problemas reais.

[1] elisamaoliveirasophia@gmail.com

DOI: 10.5540/03.2022.009.01.0325

O presente artigo aborda o PCE incluindo datas de entrega dos pedidos e *setup* de troca de padrões (PCE-DDS). O PCE-DDS é, portanto, uma extensão do PCE clássico, ao introduzir o tempo de *setup* para troca de padrões de corte, a existência de datas de entrega dos pedidos dos clientes e a possibilidade de adiantamento ou atraso na entrega destes pedidos. O objetivo é determinar uma sequência de cortes de padrões para atender aos pedidos dos clientes e satisfazer limitações de tempo, além de permitir estocar itens produzidos em excesso para ser usado em produções futuras. Apesar de haver estudos do PCE com datas de entrega feitos por [7], [2] e [1], não foram encontrados, na literatura, trabalhos que consideram, simultaneamente, datas de entrega e *setup*.

2 Definição e Modelagem do Problema

Suponha que exista em estoque um conjunto de objetos (grandes) de mesmo comprimento L e um conjunto com n tipos de itens menores de comprimento ℓ_i, $i = 1, \ldots, n$. As datas de entrega dos pedidos são organizadas segundo um horizonte de planejamento, que é dividido em períodos de tempo $t = 1, \ldots, M$. Em cada período t, deve ser atendida uma quantidade de demanda b_{ti} do tipo de item i com uma data de entrega d_{ti}. Além disso, seja a_{ij} a quantidade de itens do tipo i produzidos pelo padrão de corte j, para $j = 1, \ldots, N$.

Neste ambiente, o processamento de um pedido deve ser finalizado o mais próximo possível da sua data de entrega. Quando ocorre o adiantamento no processamento da tarefa, a produção é finalizada antes da data de entrega, devendo então ser armazenada, o que gera custo de estocagem. Já com o atraso na cadeia produtiva, ocorre o custo de atraso do pedido. Em problemas de programação, tanto a antecipação quanto o atraso na produção são considerados eventos prejudiciais e, portanto, devem ser minimizados.

Os parâmetros c_{tj} e p_{tj} denotam, respectivamente, o custo e o tempo para cortar cada objeto usando o padrão de corte j no período t. Os parâmetros f_{tj} e r_{tj} denotam, respectivamente, o custo e o tempo de *setup* referentes ao padrão j no período t. Por fim, para cada período t, u_{ti} representa o custo de estocagem do item i; β_t denota o custo por entregar itens depois de sua data de entrega; e γ_t representa o custo por entregar itens antes de sua data de entrega.

Para modelar o problema, definimos as seguintes variáveis de decisão. A variável x_{tj} representa a quantidade de objetos cortados no período t de acordo com o padrão de corte j; as variáveis T_t e E_t representam as penalizações por atraso e por antecipação referentes aos pedidos no período t; a variável tp_t representa o tempo gasto para cortar todos os padrões referentes ao período t; a variável e_{ti} representa a quantidade de itens i estocados no período t; a variável S_{tj} indica se ocorreu ou não *setup* de padrão de corte no período t para cortar o padrão de corte j; e a variável TI_t representa o tempo de início de corte dos pedidos referentes ao período t.

Usando a notação mencionada, a formulação matemática proposta para modelar o PCE-DDS

DOI: 10.5540/03.2022.009.01.0325

é dada por:

$$\min \quad \sum_{t=1}^{M}\sum_{j=1}^{N_t} c_{tj}x_{tj} + \sum_{t=1}^{M}\sum_{j=1}^{N_t} f_{tj}S_{tj} + \sum_{t=1}^{M}\beta_t E_t + \sum_{t=1}^{M}\gamma_t T_t + \sum_{t=1}^{M}\sum_{i=1}^{n} u_{ti}e_{ti} \tag{1}$$

$$\text{sujeito a:} \quad e_{ti} + \sum_{j=1}^{N} a_{itj}x_{tj} = b_{ti} + e_{(t-1)i},\ \forall t\ , \forall i \tag{2}$$

$$TI_t = d_{t-1} - E_{t-1} + T_{t-1},\ \forall t \tag{3}$$

$$TI_t + tp_t + E_t - T_t = d_t,\ \forall t \tag{4}$$

$$tp_t = \sum_{j=1}^{N} p_{jt}x_{jt} + \sum_{j=1}^{N} r_j S_{tj},\ \forall t \tag{5}$$

$$x_{tj} \leq BS_{tj}, \forall t, j \tag{6}$$

$$E_0 = 0, T_0 = 0 \tag{7}$$

$$e_{0i} = \bar{e}_i, ,\ \forall i \tag{8}$$

$$x_{tj} \in \mathbb{Z}_+,\ \forall j,\ \forall t \tag{9}$$

$$E_t, T_t, tp_t, TI_t \in \mathbb{Z}_+,\ \forall t \tag{10}$$

$$S_{tj} \in \{0, 1\}, \quad \forall t, j \tag{11}$$

$$e_{ti} \in \mathbb{Z}_+, \quad \forall t, i. \tag{12}$$

A função objetivo (1) minimiza o custo total dado pela soma dos custos referentes aos objetos cortados, *setups*, adiantamentos, atrasos e estoques. As restrições (2) modelam o balanço de estoque e garantem o atendimento da demanda do período. As restrições (3) modelam o cálculo do tempo de início de procedimento do período, em que assumimos que d_0 é igual a 0. As restrições (4) modelam o cálculo do valor do atraso ou do adiantamento que ocorram no período de corte, no qual pelo menos uma destas variáveis será nula. O conjunto de restrições (5) calcula o tempo a ser gasto na produção de pedidos em cada um dos períodos, somando os tempos de corte com os tempos de *setups*. O conjunto de restrições (6) garante que, em um determinado período, só poderá haver corte de um determinado padrão se ocorrer uma troca de padrão na máquina. As restrições (7) representam a condição de limite inicial de tempo, definindo que não há atrasos ou adiantamento de pedidos no início do processo de corte. As restrições (8) representam os valores iniciais de estoque de itens. As restrições (9)-(12) representam as condições de não negatividade e integralidade das variáveis de decisão.

3 Método de resolução

O procedimento de resolução proposto consiste em três fases executadas em sequência: na fase 1, o problema linear relaxado é resolvido pelo método simplex com geração de colunas, obtendo-se uma solução que pode não ser inteira viável para o PCE-DDS; na fase 2, determina-se uma solução inteira para o problema, aplicando a heurística Relax-and-Fix; a fase 3 busca melhorar a solução inicial inteira gerada pela fase 2. Estas duas últimas fases usam as colunas encontradas na fase 1.

3.1 Fase 1 : Problema de precificação

Neste trabalho, inicializamos o procedimento de geração de colunas usando padrões de cortes homogêneos para ajudar na composição de uma base inicial para o método simplex. Para a geração de novos padrões de cortes, com potencial para melhorar o valor da função objetivo do problema

relaxado e restrito, basta encontrar um padrão de corte j com custos relativos negativo. O custo relativo de uma variável associada a um padrão de corte j para o objeto em estoque e dada por:

$$\begin{aligned} CR &= c_{tj} - \Psi^t \alpha_{tij} \\ &= c_{tj} - \begin{bmatrix} \pi_{t1} & \dots & \pi_n & \theta_t & \zeta_t \end{bmatrix} \begin{bmatrix} \alpha_{t1} \\ \vdots \\ \alpha_{tn} \\ (p_{tj} + \frac{r_{tj}}{B}) \\ 0 \end{bmatrix} \\ &= \left(c_{tj} + \frac{f_{tj}}{B} \right) - \sum_{i=1}^{n} \pi_{ti} \alpha_{ti} - \theta_t (p_{tj} + \frac{r_{tj}}{B}), \end{aligned} \tag{13}$$

em que π_{ti} representa o valor da variável dual referente à restrição de atendimento da demanda; θ_t é o valor da variável dual referente à restrição de contagem de atraso ou adiantamento; e ζ_t é o valor da variável dual referente à restrição de tempo de início de corte. Além disso, α_{ti} indica quantos itens do tipo i têm o padrão de corte gerado para o período t. Considerando α_{ti} uma variável de decisão e usando a expressão (13), podemos formular o problema da mochila que minimize essa expressão:

$$(SP_t) \quad \min \quad \left(c_t + \frac{f_t}{B} - \sum_i \alpha_{it} \pi_{ti} + (p_t + \frac{r_t}{B}) \sigma_t \right) \tag{14}$$

$$\text{s. a} \quad \sum_i \alpha_{ti} \leq L \tag{15}$$

$$\alpha_{ti} \in \mathbb{Z}_+. \tag{16}$$

3.2 Fase 2: Encontrando uma solução inicial inteira com a Heurística Relax-and-Fix

Como nem todas as soluções na fase 1 serão inteiras, a fase 2 do algoritmo consiste na aplicação da heurística *Relax-and-Fix* para encontrar uma solução inteira viável, usando como colunas iniciais as mesmas colunas que foram geradas na fase 1. A heurística *Relax-and-Fix*, descrita por [8], consiste em um método de decomposição de um modelo de programação inteira ou de programação inteira mista em submodelos menores disjuntos, que podem ser resolvidos rapidamente, porém sem a garantia de resolução do problema original de forma ótima. Para isso, o conjunto de variáveis inteiras ou binárias do modelo é particionado e as partes da partição proposta são ordenadas. O método consiste em resolver um subproblema para cada parte, na ordem predeterminada, em que apenas as variáveis referentes a esta parte corrente, que denotaremos por parte h, devem ser inteiras, enquanto as variáveis referentes às partes $m > h$ ficam relaxadas e as variáveis referentes às partes $m < h$ ficam fixadas nos valores inteiros obtidos ao resolver o subproblema referente a esta parte. Após resolver este subproblema, o valor das variáveis referentes à parte corrente é fixado no valor ótimo obtido ao resolver este subproblema. Dessa maneira, são resolvidos submodelos com menor número de variáveis inteiras que o modelo original e, possivelmente, mais fáceis de serem resolvidos. Nesta fase, a partição dos conjuntos foi feita sobre os períodos de tempo. Ao final desse passo, obteve-se uma solução para todas as variáveis inteiras x_{tj} e e_{ti} do modelo. Dessa forma, como os atrasos e adiantamentos estão inteiramente ligados a essas variáveis, consegue-se determinar, para essas variáveis, uma solução inteira.

DOI: 10.5540/03.2022.009.01.0325

3.3 Fase 3 : Melhorando a solução inicial com a Matheurística *Local Branching*

Após a fase 2, de posse de uma solução inteira, aplicamos na fase 3 a matheurística *Local Branching* para tentar encontrar uma solução inteira viável melhor. A matheurística *Local Branching*, introduzida por [3], é baseada na exploração parcial ou completa da vizinhança de soluções viáveis. Considere que uma determinada solução de referência viável seja conhecida. Busca-se, então, encontrar uma solução aprimorada que não esteja muito longe desta solução de referência. Para encontrar soluções na vizinhança da solução, é adicionada uma restrição de *local branching* que restringe a busca, a cada iteração, a soluções que estejam em uma vizinhança de tamanho k. Neste trabalho, será apresentada a matheurística *Local Branching* aplicada ao problema usando apenas as variáveis inteiras x_{tj} para definir as restrições de *local branching*. Esta escolha se deve ao fato desta estratégia ter obtido melhores resultados em experimentos preliminares.

Seja $\Delta(x, x')$ uma medida de distância da solução x à solução x' e $I = \{(t, j) \ : \ t = 1, 2, \ldots, M, j = 1, 2, \ldots, N_t\}$. Supondo que as variáveis inteiras x_{tj} sejam limitadas, sendo $l_{tj} \leq x_{tj} \leq u_{tj}$, então, de acordo com [3], a distância $\Delta(x, x')$ pode ser definida da seguinte forma:

$$\Delta(x, x') \quad := \sum_{(t,j) \in I : x_{tj}{}' = l_{tj}} \mu_{ij}(x_{tj} - l_{tj}) + \sum_{(t,j) \in I : x_{tj}{}' = u_{tj}} \mu_{tj}(u_{tj} - x_{tj}) \tag{17}$$

em que μ_{tj} pode ser definido por:

$$\mu_{tj} = \frac{1}{u_{tj} - l_{tj}}.$$

Assim, dada uma solução de referência x', o espaço de solução pode ser particionado por meio da disjunção:

$$\Delta(x, x') \quad \leq \quad k \qquad \text{ou} \tag{18}$$

$$\Delta(x, x') \quad \geq \quad k + 1. \tag{19}$$

A proposta deste método é, assim, explorar o espaço de soluções, considerando as duas partes definidas por este particionamento. Inicialmente, é feita uma busca da melhor solução na vizinhança da solução corrente x', ou seja, considerando as soluções viáveis x que satisfazem à condição (18), busca-se encontrar uma solução que seja melhor que a solução corrente. Caso esta busca seja bem sucedida, a melhor solução encontrada será a nova solução de referência. Além disso, como as soluções x que satisfazem à condição (18) já foram exploradas, uma nova fase de exploração pode ser realizada nas soluções que satisfazem à condição (19). Da mesma forma, ao explorar o novo espaço de busca, é feito um novo particionamento do espaço de solução que ainda falta ser explorado. Logo o algoritmo realiza sistematicamente uma busca local na melhor solução conhecida.

4 Resultados Computacionais

Os testes computacionais foram executados em um computador com Sistema Operacional Linux-Ubuntu, Processador Intel Core i7–7^{a} geração, 8GB de RAM, 2TB de Disco Rígido. O pacote `Concert Technology CPLEX` 12.7.1 foi usado com a linguagem `C++` e o limite de tempo de execução de 1200 segundos foi imposto. Para os testes computacionais, foram geradas 20 instâncias para a realizar a análise da modelagem e do método de solução proposta. Para isto, foi desenvolvido um gerador de instâncias, na linguagem `C++`, baseado em [4]. As instâncias geradas tem as seguintes características:

(i) comprimento do objeto em estoque $L = 1000$;

DOI: 10.5540/03.2022.009.01.0325

(ii) número de tipos de itens $n = 50$;

(iii) número de períodos de tempo $t = 3$;

(iv) as demandas dos itens foram escolhidas aleatoriamente, segundo uma distribuição uniforme, no intervalo $[1, 50]$;

(v) os comprimentos dos itens foram escolhidos aleatoriamente, segundo uma distribuição uniforme, no intervalo $[0,01L; 0,6L]$;

(vi) custo por cortar um padrão de corte j em um período t: $c_{tj} = 1$;

(vii) tempo de corte do padrão de corte j em um período t: $p_{tj} = 1$;

(viii) custo de troca de padrões de corte, custo de atrasar ou adiantar a entrega dos pedidos foram escolhidos aleatoriamente segundo uma distribuição uniforme no intervalo de $[1, 10]$;

(ix) os custos de estocar itens foram escolhidos aleatoriamente, segundo uma distribuição uniforme, no intervalo $[0,1\ell_i; 0,3\ell_i]$;

(x) as datas de entrega foram geradas segundo uma distribuição uniforme no intervalo $[0,8 \times CAP; 1,1 \times CAP]$. O termo CAP é definido como a capacidade de corte da máquina em um determinado período, e pode ser calculado pela expressão:

$$CAP = \left\lfloor \frac{\sum_{i=1}^{n} l_i b_{ti}}{L} \right\rfloor . \tag{20}$$

A Tabela 1 apresenta os resultados para as 20 instâncias referentes a quatro abordagens: (i) resultado obtido na Fase 1; (ii) resultado obtido após a Fase 2; (iii) resultado obtido após a Fase 3; e (iv) resultado obtido ao resolver o problema inteiro com as colunas geradas usando o *solver* CPLEX. Esta tabela apresenta a quantidade média de *setups*, a quantidade média de objetos cortados, o tempo médio de atraso e de adiantamento, os estoques médios e o valor médio da função objetivo, o tempo computacional médio e quantidade média de problemas com atraso e adiantamento.

Tabela 1: Resultados referentes a cada Fase e à resolução via *solver* CPLEX para as 20 instâncias

Métodos de Solução	*Setup*	Objetos	Atraso	Adiantados	Estoque Anterior	Estoque Final	Função	Tempo - (#limite atingido)	Problemas com Atraso	Problemas com Adiantamento
GC (Fase 1) Limite Inferior	10,61	10615,24	14,01	305,18	0	0	11084,26	3,44	1	2
Heurística *Relax-and-Fix* (Fase 2)	155,3	10502,95	26,4	299,1	280,15	126,85	13828,85	19,21	1	2
Matheurística *Local Branching* (Fase 3)	153,05	10503,65	26,4	119,1	645,15	261,45	13827,35	118,51 - (1)	1	1
solver CPLEX com as colunas da Fase 1	154,2	10504,35	27,65	299,1	285,2	127,25	13846,85	183,53 - (3)	1	2

A linha GC (Fase 1) apresenta as médias das soluções para o problema na versão relaxado, sendo utilizada como Limitante Inferior para a comparação das soluções, uma vez que a função objetivo é minimizada. Com essa solução, não foi gerada nenhum item em estoque para ser usado em períodos posteriores.

A coluna *setup* se refere a média da soma das variáveis de *setup*. Nas Fases 2 e 3 e na resolução via *solver* CPLEX, em que as variáveis de *setup* são inteiras, a soma contabiliza as trocas de

padrões que foram feitas. Para todas as fases o valor é bem próximo, em torno de 150 padrões de corte. O mesmo acontece com o número de objetos que foram cortados, em média de 10500. Para esse conjunto de instâncias, apenas um apresentou atraso na entrega dos itens e dois apresentaram adiantamento; porém, com a matheuristica *Local Branching* (Fase 3) houve apenas um problema com adiantamento. Além disso, a Fase 3 conseguiu melhorar a solução inicial apresentada pela Heurística *Relax-and-Fix*. As soluções inteiras foram comparadas com a solução fornecida pelo *solver* CPLEX quando utilizam as colunas da Fase 1. A coluna Tempo - (#limite atingido) apresenta a média do tempo computacional acumulado até finalizar a fase, por exemplo, o tempo referente à Fase 2 é a soma dos tempos gastos nas Fases 1 e 2. A solução apresentada pela proposta de solução é de melhor qualidade que a solução encontrada pelo *solver*, uma vez que as soluções vindas do *solver* tiveram tempo médio de solução pior, e três dessas instâncias atingiram o tempo limite de 1200 segundos. Para o procedimento proposto, apenas um dos problemas atingiu o tempo limite. Todas os valores médios referentes à Fase 3 foram menores que os valores médios referentes às soluções encontradas pelo CPLEX, com exceção dos estoques.

5 Considerações Finais

Neste trabalho foi proposta uma modelagem para o PCE-DDS, além de apresentar um método de solução composto por três fases. Os resultados foram comparados com a solução fornecida pelo *solver* CPLEX. Os resultados computacionais mostram que as soluções encontradas pelo procedimento proposto foram superiores, em média, além de apresentar um tempo computacional inferior.

Agradecimentos

Este trabalho teve apoio da CAPES, CNPq e Fapemig.

Referências

[1] Claudio Arbib e Fabrizio Marinelli. “Maximum lateness minimization in one-dimensional bin packing”. Em: **Omega** 68 (2017), pp. 76–84.

[2] Claudio Arbib e Fabrizio Marinelli. “On cutting stock with due dates”. Em: **Omega** 46 (2014), pp. 11–20.

[3] Matteo Fischetti e Andrea Lodi. “Local branching”. Em: **Mathematical programming** 98.1-3 (2003), pp. 23–47.

[4] T. Gau e G. Wäscher. “CUTGEN1: A problem generator for the standard one-dimensional cutting stock problem”. Em: **European Journal of Operational Research** 84.3 (1995), pp. 572–579.

[5] Paul C. Gilmore e Ralph E. Gomory. “A linear programming approach to the cutting stock problem – Part II”. Em: **Operations research** 11.6 (1963), pp. 863–888.

[6] Paul C. Gilmore e Ralph E. Gomory. “A linear programming approach to the cutting-stock problem”. Em: **Operations research** 9.6 (1961), pp. 849–859.

[7] Harald Reinertsen e Thomas WM Vossen. “The one-dimensional cutting stock problem with due dates”. Em: **European Journal of Operational Research** 201.3 (2010), pp. 701–711.

[8] Laurence A. Wolsey. **Integer Programming**. New York, NY, EUA: John Wiley & Sons, 1998.

