UMA HEURÍSTICA GULOSA ITERADA PARA UM PROBLEMA DE SEQUENCIAMENTO E DIMENSIONAMENTO DE LOTES EM UM AMBIENTE FLOW SHOP

Harlem Mauricio Madrid Villadiego
Departamento de Informática - Universidade Federal de Viçosa
Viçosa - MG - Brasil
villadiego@ufv.br

José Elias Claudio Arroyo
Departamento de Informática - Universidade Federal de Viçosa
Viçosa - MG - Brasil
jarroyo@dpi.ufv.br

RESUMO

Neste artigo é abordado um problema integrado de dimensionamento e sequenciamento de lotes num ambiente de produção flow shop permutacional com múltiplas máquinas que possuem diferentes capacidades de produção e requerem tempos de preparação dependentes da sequência. Para um horizonte de planejamento com múltiplos períodos, o problema consiste em determinar, para cada período, as dimensões dos lotes de produtos e o sequenciamento dos mesmos de tal maneira que as demandas dos clientes sejam atendidas e as capacidades das máquinas sejam respeitadas. O objetivo é minimizar a soma dos custos de processamento, preparação e estoques. Dada a complexidade do problema, neste trabalho propõe-se uma heurística Gulosa Iterada (*Iterated Greedy*) que possui uma etapa de sequenciamento e outra de dimensionamento dos lotes. A heurística proposta é comparada com os melhores métodos disponíveis na literatura. A fim de validar o desempenho da heurística proposta, uma variedade de testes computacionais foram realizados e os resultados foram analisados através de testes estatísticos.

PALAVRAS CHAVE. Dimensionamento de lotes, flow shop scheduling, heurísticas.

Área Principal: Metaheurísticas

ABSTRACT

In this work, we consider the integrated lot-sizing and sequencing problem in a permutation ow shop with machine sequencedependent setups. The problem is to determine the lot sizes and the production sequence in each period of a planning horizon such that the customer demands must be met and the capacity of the machines must be respected. The objective is to determine the sum of the setup costs, the production costs and the inventory costs over the planning horizon. Due to the complexity of the problem, we propose a heuristic based on Iterated Greedy metaheuristic which uses sequencing and lot-sizing decisions. The proposed method is compared against the best heuristics available in the literature. Comprehensive computational and statistical analyses are carried out in order to validate the performance of the proposed heuristic.

KEYWORDS. Lot-sizing, flow shop scheduling, heuristics

1. Introdução

A programação da produção é um dos assuntos mais importantes nas indústrias de manufatura desde que é a atividade que permite coordenar todas as operações no processo produtivo a fim de cumprir os compromissos realizados com os clientes da empresa. Conforme Pochet e Wosley (2006), a programação da produção envolve decisões sobre dimensionamento e sequenciamento de lotes. Um lote indica uma quantidade específica de um produto a ser produzida (processada) em uma máquina continuamente sem interrupção. O sequenciamento envolve a decisão de estabelecer a ordem na qual os lotes serão processados, determinando seus tempos de início e finalização.

Na literatura várias pesquisas tratam sobre problemas de dimensionamento e sequenciamento de lotes (PDSL) simultaneamente. Problemas com características tais como: restrições de capacidade nas máquinas, múltiplos produtos e com tempos de preparação das máquinas dependentes da sequência, são mais comuns. Barany et al. (1984) propuseram formulações matemáticas fortes para o PDSL com restrições de capacidade (PDSLC) obtendo bons resultados em um ambiente de produção com uma máquina. Almada-Lobo et al. (2007) desenvolveram um método heurístico com formulações exatas para o PDSLC considerando tempos de preparação dependentes da sequência. Almada-Lobo et al. (2010) também estuda o PDSLC com tempos de preparação dependentes da sequência. Estes autores propõem uma heurística *Variable Neighbourhood Search* (VNS) e seus resultados foram comparados com a heurística proposta por Almada-Lobo et al. (2007). Uma revisão de literatura sobre o PDSLC pode ser encontrada in Drex et al. (1997). Na literatura é comum adicionar a característica multi nível ao PDSLC, que é considerada quando existe uma dependência na produção de vários produtos em diferentes níveis de produção. Os ambientes de produção encontrados no PDSLC multi nível (PDSLCM) podem ser: máquina simples, máquinas paralelas, flow shop, job shop entre outros. Modelos e algoritmos para o PDSLCM são discutidos em Karimi et al. (2003). Zhu e Wilhelm (2006) realizam uma revisão de literatura sobre problemas de dimensionamento e sequenciamento de lotes e ressaltam que pesquisas em ambiente produção do tipo flow shop são escassas.

Dado que o escopo deste trabalho é o problema de dimensionamento e sequenciamento de lotes em um flow shop (PDSLFS), em seguida são apresentadas as pesquisas, que para nosso conhecimento, são as mais relevantes.

Sikora et al. (1996a) trata o problema de dimensionamento e sequenciamento de lotes em um ambiente de produção flow shop. O problema existe em uma indústria de placas de circuito impresso, considerando, restrições de capacidade nas máquinas, tempos de preparação dependentes da sequência e estoques intermediários. Os autores propõem uma heurística baseada em dois métodos não integrados, um para resolver o dimensionamento e outro para resolver o sequenciamento. Para o mesmo problema, Sikora et al. (1996b), propuseram um algoritmo genético que supera a heurística proposta por Sikora et al. (1996a). Ponnambalam et al. (2003) propuseram uma heurística híbrida que combina um Algoritmo Genético (AG) com um Simulated Annealing (SA). Nessa heurística, o AG tenta resolver o dimensionamento de lotes enquanto o SA o sequenciamento. Os resultados obtido por este método híbrido são melhores em comparação com os obtidos por Sikora et al. (1996b). Smith et al. (1988) foram os primeiros em propor um modelo matemático para o PDSLFS, considerando: famílias de produtos, estoques intermediários, tempos de preparação dependentes da sequência e mantidos entre períodos. Dado que o modelo apresenta um grande número de variáveis decisões e restrições, este só resolve instâncias de pequeno tamanho, portanto, foi de pouco interesse na literatura.

Mohammadi et al. (2010a) propõem um modelo matemático para o PDSLFS, neste problema se considerou: tempos de preparação dependentes da sequência, demandas que devem ser atendidas, as capacidades das máquinas devem ser respeitadas e os tempos de preparação das máquinas são preservados entre períodos. Estes autores também apresentaram dois limitantes inferiores, duas heurísticas baseadas na estratégia de "horizonte rolante" e duas heurísticas baseadas na estratégia de "*relax and fix*". Os autores afirmam que as heurísticas baseadas na estratégia de

relax and fix geraram os melhores resultados. Mohammadi et al. (2010b) estende o trabalho realizado por Mohammadi et al. (2010a), usando o mesmo modelo matemático e os mesmos limitantes inferiores além de propor melhoras significativas nas heurísticas. Mohammadi et al. (2010c) considera o mesmo modelo matemático e os limitantes inferiores de Mohammadi et al. (2010b), e propõem duas novas heurísticas baseadas na estratégia de horizonte rolante e na clássica heurística NEH (Nawaz, 1983). Belo Filho et al. (2012) propuseram uma heurística *Asynchronous Teams* (*A-Teams*) que é comparada com os métodos propostos por Mohammadi et al. (2010c). Belo Filho et al. (2012) mostram o que a heurística *A-Teams* determina os melhores resultados. Ramezanian et al. (2012) propõem um novo é mais eficiente modelo matemático para o PDSLFS. Este modelo possui um menor número de variáveis de decisões e restrições. Além do novo modelo, estes autores propõem duas heurísticas baseadas na estratégias de horizonte rolante. Estas heurísticas superam os métodos propostos por Mohammadi et al. (2010c).

Neste artigo, é proposta uma heurística *Iterated Greedy* (IG) para resolver o PDSLFS. O problema PDSLFS abordado possui as mesmas características dos problemas estudados em Mohammdi et al. (2010c), Belo Filho et al. (2012) e Ramezanian et al. (2012). o IG é uma heurística simples e eficiente proposta por Ruiz et al. (2006) para o problema de sequenciamento de tarefas num ambiente flow shop. O IG procura a melhor sequência de produção (ordem de processamento dos produtos) em cada período, minimizando os custos de preparação. As decisões correspondentes às dimensões dos lotes são determinadas por métodos que adiantam e postergam a produção dos produtos entre os períodos. A heurística IG proposta é comparada com as duas melhores heurística disponível na literatura: o *A-Teams* proposto por Belo Filho et al. (2012) e a heurística baseada na estratégia de horizonte rolante proposto por Ramezanian et al. (2012).

2. Definição do PDSLFS

Neste artigo estuda-se o problema integrado de dimensionamento e sequenciamento de lotes. Neste problema existem N produtos independentes a serem produzidos. Cada lote é formada por uma quantidade especifica de um produto, os quais devem ser processados em um ambiente de produção flow shop permutacional, onde as máquinas são limitadas em capacidade e dispostas em serie em um horizonte de planejamento dividido em T períodos. Em cada período existe uma demanda para cada produto, a qual sempre deve ser atendida. As principais suposições/condições do problema são descritas a seguir.

Os T períodos são iguais. Os lotes de produtos devem ser processados na mesma ordem nas M máquinas que são dispostas em série. Cada máquina pode processar só um lote de cada vez. Em cada período, uma máquina possui uma capacidade de processamento. Os custos e tempos de preparação das máquinas são dependentes da sequência. A preparação da máquina deve iniciar e terminar no mesmo período. No início do horizonte de planejamento as máquinas estão preparadas para um determinado produto. As preparações de máquinas são preservadas entre períodos (*setup carryover*). Em cada período existe uma demanda de produtos acabados, a qual sempre deve ser atendida no mesmo período. Não existe estoque entre duas máquinas consecutivas. O tempo de transporte de produtos intermediários e outros processos entre as máquinas sucessivas é desprezível. Não há prioridade de produção nas máquinas entre os produtos.

O problema consiste em determinar, para cada período, a quantidade de produtos a serem produzidos num lote (tamanho ou dimensão do lote), assim como a ordem na qual os lotes serão processados nas *M* máquinas do ambiente. A função objetivo (f) do problema é minimizar a soma dos custos de processamento, preparação e estoque.

3. Heurística Iterated Greedy para o PDSLFS

A ideia básica da heurística IG é melhorar iterativamente uma solução através de três diferentes métodos: destruição-construção, busca local e critério de aceitação. Além destes métodos, neste trabalho é adicionado um método de melhoria para o dimensionamento de lotes (*MDL*). Então,

a heurística proposta é nomeada de *IG+MDL*. O *IG+MDL* começa gerando uma solução inicial por meio de um procedimento construtivo. Logo aplica iterativamente os métodos mencionados anteriormente até que a condição de parada seja satisfeita. Os métodos destruição-construção e busca local são usados para gerar uma boa sequência de produção. O método *MDL* é aplicado à melhor sequência de produção obtida pela busca local a fim de melhorar as quantidades a serem produzidas (tamanho dos lotes). A cada iteração, e por meio do critério de aceitação, a nova solução atual é atualizada com a solução obtida pelo método *MDL*.

O Algoritmo 1 apresenta o funcionamento da heurística e seus componentes. O Algoritmo recebe como parâmetros: O número máximo de iterações do algoritmo $(Iter_{Max})$, um parâmetro utilizado no método de Destruição-Construção (d), número de iterações da busca local $(Iter_{Bl})$ e o número máximo de iterações para o método de melhoria do dimensionamento de lotes $(Iter_{MDL})$. O algoritmo inicia gerando uma solução inicial (linha 1). Se é construída uma solução inviável (com relação ao dimensionamento), é usado o modelo matemático proposto por Ramezanian et al. (2012) para obter uma solução factível (linhas 2 - 4). As iterações do *IG+MDL* são calculadas nas linhas 6 a 15 até que um número máximo de iterações seja atingido. Nas linhas 7 e 8 são executados os procedimentos de Destruição-Construção e Busca Local, respectivamente. O *MDL* é executado na linha 9. Entre as linhas 10 e 14 testa-se o critério de aceitação e atualiza-se a solução atual. Finalmente a melhor solução obtida pelo algoritmo é retornada na linha 16.

Algorithm 1 : IG+MDL($Iter_{Max}$, d, $Iter_{BL}$, $Iter_{MDL}$)

1: SOL := Construção_Solução_Inicial();
2: **if** Solução_Infatível(SOL) **then**
3: SOL := Modelo_Matemático(SOL);
4: **end if**
5: $SOL^* := SOL$;
6: **for** *iterações* := 0 **to** $Iter_{Max}$ **do**
7: SOL := Destruição-Construção(SOL, d);
8: SOL := Busca_Local($SOL, Iter_{BL}$);
9: SOL := Melhoria_Dimensionamento_Lotes($SOL, Iter_{MDL}$);
10: **if** $f(SOL) < f(SOL^*)$ **then**
11: $SOL^* := SOL$;
12: **else**
13: $SOL := SOL^*$;
14: **end if**
15: **end for**
16: **return** SOL^*;

Nas próximas subseções apresentam-se a descrição dos métodos do Algoritmo *IG+MDL*, incluindo a representação de uma solução e a geração de uma solução inicial.

3.1. Representação de uma Solução

Uma solução do problema em estudo é representada por T matrizes de ordem $3 \times N$ (onde T é o numero de períodos e N é o número de produtos). A primeira linha de cada matriz representa a sequência de lotes; a segunda e terceira linha representam, respectivamente, as quantidades de produtos produzidos em cada lote e o estoque de cada produto (produção não demanda). Na Figura 1 mostra-se um exemplo de uma solução para dois períodos (t_1 e t_2) com três produtos, onde L_j é o lote do produto j. A sequência de lotes no primeiro período é (L_3, L_2, L_1) e no segundo período é (L_1, L_2, L_3). DL_{jt} e Est_{jt} representam, respectivamente, as dimensões do lote e o do estoque do produto i no período t.

	t_1			t_2		
Sequência de produção	L_3	L_2	L_1	L_1	L_2	L_3
Tamanho do Lote	DL_{31}	DL_{21}	DL_{11}	DL_{12}	DL_{22}	DL_{32}
Tamanho do Estoque	Est_{31}	Est_{21}	Est_{11}	Est_{12}	Est_{22}	Est_{32}

Figura 1: Representação de uma Solução

3.2. Construção da solução inicial

A construção de uma solução inicial é feita em duas fases. Na primeira fase, é construída a sequência de produção para cada período. Na segunda fase é determinado o dimensionamento de lotes para as sequências obtidas na primeira fase. A seguir descrevem-se cada uma das fases.

3.2.1. Construção das sequências de produção

Para construir as sequências de produção é adotada a heurística NEH (Nawaz, et al. 1983) de forma similar à ideia proposta por Mohammadi et al. (2010b). Neste trabalho, a sequência inicial π_I para a heurística NEH é determinada da seguinte forma. Para cada produto calcula-se o custo total de preparação, em seguida a sequência inicial é formada pelos produtos arranjados em ordem decrescente do custo total de preparação.

Seja π_I a sequência inicial dada por $\pi_I = (\pi_1, \pi_2, ..., \pi_N)$. Iniciando com $\pi_P = (\pi_1)$, uma sequência de produção é construída passo a passo inserindo-se o próximo produto de π_I na melhor posição possível de π_P. A heurística finaliza quando todos os produtos de π_I forem inseridos em π_P.

O procedimento heurístico é aplicado para todos os períodos. A sequências do período $t > 1$ deve iniciar com o último produto da sequência do período anterior $(t-1)$.

3.2.2. Inicialização do dimensionamento de lotes

Considerando a sequência de produção obtida na fase anterior, os tamanhos dos lotes são determinados do último período para o primeiro (dimensionamento regressivo) conforme os seguintes casos: i) se a capacidade das máquinas de um período t é suficiente para produzir todas as quantidades equivalentes à demanda de todos os produtos do mesmo período t, o tamanho do lote de cada produto é igual à demanda do mesmo, isto é a política lote por lote; ii) se a capacidade de um período t é insuficiente para produzir todos os produtos, são apenas produzidos os produtos para os quais a capacidade é suficiente, aplicando a política lote por lote. Os produtos não produzidos no período t serão produzidos no período imediatamente anterior com capacidade disponível. Este procedimento é similar ao utilizado por Shim et al.(2011). Em alguns casos pode-se obter uma solução inviável dado que as capacidades dos primeiros períodos podem ser insuficientes. Quando isto ocorre, é necessária a execução do modelo matemático proposta por Ramezanian et al.(2012) para determinar somente o dimensionamento de lotes utilizando as sequências de produção obtidas pela heurística NEH.

O Algoritmo 2 apresenta a inicialização do dimensionamento de lotes (*IDL*), o qual recebe como parâmetro só as sequências de produção para todos os períodos. Neste algoritmo as linhas 1, 2 e 17, mostram que o dimensionamento de lotes é realizado de maneira regressiva, iniciando com o último período T de planejamento e finalizando no primeiro. O dimensionamento inicia no primeiro produto $(j = 1)$ da sequência de produção e continua até determinar o tamanho do lote do último produto $(j = N)$ da sequência (linha 4). Na linha 5 do algoritmo testa-se se existe capacidade suficiente para produzir as quantidades equivalente à demanda do produto j no período t. Se existe tal capacidade, na linha 6 se define o tamanho do lote do produto j no período t (DL_{jt}) igual à demanda do mesmo produto j no mesmo período t (d_{jt}). Na linha 7 atualiza-se a capacidade. Note que após realizado um dimensionamento, a capacidade no período t é reduzida. Se o teste da linha 5 afirma que existe uma capacidade insuficiente para dimensionar o lote do produto j no período t, na linha 9 salva-se esse produto (i) e finaliza-se o ciclo referente ao dimensionamento do produto

j em seguida é feita a modificação das demandas (linhas 13-16). Dado que é obrigatório atender toda a demanda, a partir do produto i até o último produto da sequência de produção, os quais não foram produzidos no período t por capacidade insuficiente, deverão ser processados no período imediatamente anterior a t com capacidade disponível, para isto, a demanda de cada produto i no período t (d_{it}) é transferida para o período anterior $t-1$. Este procedimento de transferência de demanda para um período anterior esta representado nas linhas 13 até a 16. Finalmente na linha 19 retorna-se uma solução com os dimensionamentos determinados.

Algorithm 2 : $IDL(SOL)$;

1: $t := T$;
2: **while** $t \geq 1$ **do**
3: $i := N + 1$;
4: **for** $j := 1$ **to** N **do**
5: **if** $Capacidade\_Suficiente(j, t)$ **then**
6: $DL_{jt} = d_{jt}$; //SOL é atualizada neste passo.
7: $Atualizar\_Capacidade(t)$;
8: **else**
9: $i := j$;
10: $break$;
11: **end if**
12: **end for**
13: **while** $i \leq N$ **do**
14: $d_{i(t-1)} := d_{i(t-1)}$ + d_{it}
15: $i := i + 1$;
16: **end while**
17: $t := t - 1$;
18: **end while**
19: **return** SOL;

3.3. Procedimento de Destruição-Construção

Este procedimento consiste de duas operações: destruição e construção. A destruição é aplicada a cada sequência de produção π de N produtos. Esta operação remove aleatoriamente d produtos de π, obtendo duas subsequências, a primeira (π_D) com $N - d$ produtos e a segunda com os produtos removidos, a qual é definida como π_R. π_R contém os produtos que serão reinseridos em π_D para obter uma nova sequência de produção. A operação de construção inicia com a subsequência π_D e realiza d passos nos quais os produtos de π_R são reinseridos em π_D. Este processo inicia inserindo o primeiro produto de π_R em todas as $N - d + 1$ possível posições de π_D. A melhor posição para este produto é selecionada (melhor com relação ao custo de preparação (f_{cp})). Este processo é repetido até que a subsequência π_R esteja vazia. O procedimento de Destruição-Construção é aplicado à sequência de produção de cada período. No Algoritmo 3 apresenta-se este procedimento, o qual recebe como parâmetros uma solução SOL e o parâmetro de destruição d.

Algorithm 3 : Destruição_Construção(SOL, d);

for $t := 1$ **to** T **do**
  π := sequência de produção inicial (do período de t).
  $\pi_D := \pi$;
  $\pi_R := \emptyset$;
  for $i := 1$ **to** d **do**
    π_D := remova um produto aleatoriamente de π_D e insira este em π_R;
  end for
  //os tamanhos de π_D e π_R são $N - d$ e d respectivamente.
  for $i := 1$ **to** d **do**
    π_D := melhor sequência obtida após inserir o produto $\pi_R(i)$ em todas as posições possíveis de π_D;
  end for
  if $f_{cp}(\pi_D) < f_{cp}(\pi)$ **then**
    $\pi := \pi_D$;
  end if
end for
return π;

Algorithm 4 : Busca_Local$(SOL, Iter_{Bl})$

for $t := 1$ **to** T **do**
  for $l := 1$ **to** $Iter_{BL}$ **do**
    Loop := 1, Tipo_Viz := 1
    while Loop := 1 **do**
      $\phi := \pi$ //sequência de produção do período de t
      if Tipo_Viz := 1 **then**
        ϕ := Sequência obtida pela troca de dois produtos selecionados aleatoriamente;
      else
        ϕ := Melhor sequência obtida após inserir um produto aleatoriamente selecionado em todas as posições possíveis;
        Loop := 0;
      end if
      if $f_{cpm}(\phi) < f_{cpm}(\pi)$ **then**
        $\pi := \phi$; //SOL é atualizada neste passo.
        Loop := 1, Tipo_Viz := 1
      else
        Tipo_Viz := 0
      end if
    end while
  end for
end for
return SOL;

3.4. Busca local

A sequência retornada pelo procedimento de Destruição-Construção é melhorada pelo procedimento de busca local (BL). Este procedimento é baseado nos movimentos de troca simples e inserção. Esta BL é uma variante do método VNS (*Variable Neighborhood Search*) proposta por Tasgetiren et al. (2007) para resolver o problema de sequenciamento flow shop. O movimento de

troca seleciona aleatoriamente dois produtos na sequência e intercambia suas posições. O movimento de inserção remove aleatoriamente um produto da sua posição inicial e é inserido em todas as possíveis posições. A função a minimizar na BL é também o custo de preparação (f_{cp}). A BL é detalhada no Algoritmo 4, o qual recebe como parâmetros: uma solução SOL e o máximo número de iterações $(Iter_{BL})$ que é usado como critério de parada.

3.5. Melhoria do Dimensionamento de Lotes

A Melhoria do Dimensionamento de Lotes (*MDL*) consiste de dois métodos. O primeiro método adianta a produção (*MAP*) e o segundo método posterga a produção (*MPP*).

O *MAP* adianta a produção (lote inteiro) de um produto j de um período t para o período imediatamente anterior $t-1$, se neste último período o produto j esta sendo produzido e existe capacidade disponível na respectiva máquina. O produto j e o período t são aleatoriamente selecionados. Neste método, o adiantamento de lotes parciais não é considerado, desta maneira evitam-se custos de processamento, preparação e estoque no período t, incorridos pela produção do produto j.

Para ilustrar um movimento do *MAP*, na Figura 2, o produto 3 e o período t_2 foram ambos selecionados de maneira aleatória. Posteriormente no período t_1 verifica-se se existe capacidade disponível para produzir todo o lote do produto 3. Dado que existe capacidade, todo este lote é adiantado para o período anterior t_1, consequentemente, é gerado um estoque do produto 3 no período t_1. O produto 3 no período t_2 deve ser excluído já que as quantidades equivalentes à demanda deste produto estão sendo produzidas no período t_1.

No *MPP*, a produção excedente de um produto j de um período t é postergada para o período imediatamente posterior $t+1$, se o produto j esta sendo produzido neste período. A produção excedente a ser postergadas depende da capacidade disponível no período $t+1$, desta forma, se é possível todo o estoque do produto j será postergado. Se não for possível, será postergado parte do estoque que seja permitido pela capacidade. O principal propósito deste método é diminuir o custo de estoque, portanto, o produto selecionado aleatoriamente j devera ter produção excedente. Para uma melhor eficiência do *MPP*, se tem uma lista com os períodos nos quais existem produtos com estoques positivos, dentre esses, o período t é selecionado também de maneira aleatória.

A Figura 3 ilustra um movimento do *MPP*. No *MPP*, primeiramente é selecionado aleatoriamente um período com produção excedente de algum produto. No exemplo da Figura é selecionado o período t_1, este tem os produtos 1 e 3 com estoques. Dentre estes produtos o 1 é aleatoriamente selecionado. Em seguida, verifica-se se existe capacidade disponível no período t_2 para produzir todo o estoque do produto 1 do período t_1. Dado que a capacidade é suficiente, todo o estoque do produto 1 no período t_1 é postergado para o período t_2. Nota-se que o estoque do produto 1 no período t_1 foi reduzido a zero, isto leva a um aumento no tamanho do lote deste mesmo produto no período t_2.

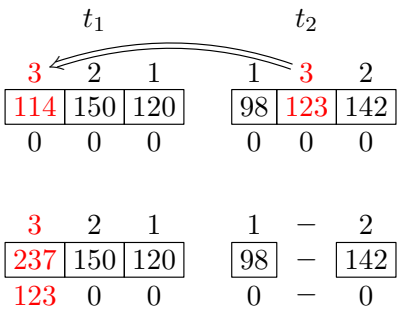

Figura 2: Método para Adiantar a Produção

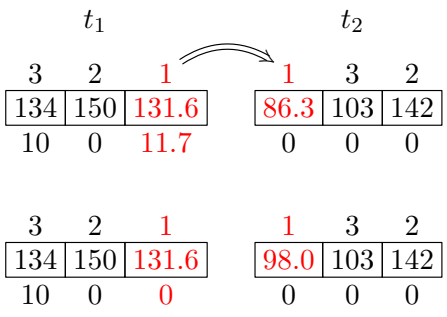

Figura 3: Método para Postergar a Produção

4. Experimentos Computacionais

Nesta seção apresentam-se os testes computacionais realizados para analisar o desempenho da heurística proposta. Todos os testes foram executados em um computador com processador *Intel(R) Xeon(R) CPU X5650 2.67GHz com 48GB de RAM*. Os resultados obtidos pela heurística *IG+MDL* são comparados com os resultados obtidos pelas heurísticas *A-Teams* e Horizonte Rolante (*HR*) propostos por Belo Filho et al. (2012) e Ramezanian et al. (2012), respectivamente. Estas heurísticas foram reimplementadas conforme os artigos originais. No *A-Teams* foi usado o modelo *MIP* proposto por Ramezanian et al. (2012). Todos os algoritmos foram programados em *C++*, usando classes e bibliotecas de *IBM ILOG CPLEX 12.4 e CPLEX Concert Technology*.

Para avaliar o desempenho da heurística proposta, foram geradas 101 classes de instâncias do problema. Cada classe é definida pelo número de produtos, máquinas e períodos: (N, M, T) que determina o tamanho da instância. O tamanho das instâncias vária de $(3, 3, 3)$ a $(60, 20, 20)$. Estas instâncias foram geradas conforme os critérios definidos por Mohammadi et al. (2010a). Para cada classe (N, M, T), cinco instâncias foram geradas, obtendo um total de 505 instâncias.

4.1. Calibração da heurística *IG+MDL*

Para esta heurística *IG+MDL* quatro parâmetros foram calibrados: parâmetro de destruição (d), número de iterações da busca local $(Iter_{BL})$, número de iterações do método de melhoria do dimensionamento de lotes $(Iter_{MDL})$ e número de iterações totais da heurística $(Iter_{IG})$. Para cada parâmetro, três níveis de valores foram testados: $d = 0.2N, 0.3N, 0.4N$; $Iter_{BL} = 3, 4, 5$; $Iter_{MDL} = 4, 5, 6$; $Iter_{IG+MDL} = 100, 150, 200$. Ao fazer a combinação de valores dos quatro parâmetros obtém-se 81 configurações do algoritmo *IG+MDL*.

Para determinar a melhor combinação dos parâmetros, foi realizado um projeto experimental onde cada parâmetro é um fator controlável. Todas as 81 configurações foram testadas através de um projeto fatorial completo (Montgomery et al. 2001). Para cada configuração de valores, o algoritmo foi executado dez vezes para cada instância do problema. Como variável resposta para o experimento é usado a porcentagem de desvio relativo (*RPD*), a qual é calculado da seguinte forma:

$$RPD\% = 100 \times \frac{f_{metodo} - f_{melhor}}{f_{melhor}} \quad (1)$$

Onde f_{metodo} é a média do valor da função objetivo (entre as 10 execuções) obtidas por uma dada configuração do algoritmo e f_{melhor} é o melhor valor da função objetivo encontrado entre todas as configurações do algoritmo.

Todos os resultados (não mostrados em detalhes por razões de espaço) foram analisadas por médio de uma analise de variância (ANOVA). Os melhores valores encontrados para os parâmetros da heurística *IG+MDL* são: $d = 0,2N$, $Iter_{BL} = 5$, $Iter_{MDL} = 4$, $Iter_{IG} = 200$.

4.2. Resultados e comparações

Os resultados obtidos pelas três heurísticas, *IG+MDL*, *HR*, *A-Teams*, são comprados usando o *RPD* (equação 1). A fim de validar os resultados obtidos pelas três heurísticas e verificar se existem ou não diferenças significativas entre elas, foi realizado um teste ANOVA, onde o *RPD* é considerado como um fator controlável. Nesta análise foram usadas as 505 instâncias. Desde que o *valor-p* encontrado através do ANOVA é menor que 0.05, conclui se que existe uma diferença estatística significativa entre as heurísticas com um nível de confiança do 95%.

O teste ANOVA não especifica entre quais heurísticas existem diferenças. Então, é necessário realizar um teste de múltiplas comparações para comparar cada par de médias com um nível de confiança de 95%. A Tabela 1 mostra os resultados deste teste. Nesta tabela, a coluna "diferença", mostra o resultado da média do primeiro algoritmo menos a média do segundo. A coluna "+/- limite" mostra o intervalo de incerteza para a diferença. Para o par de heurísticas, cuja diferença (em valor absoluto) exceda o limite, estas são estatisticamente diferentes com um nível

Tabela 1: Teste de comparações múltiplas para o RPD (qualidade)

Contraste	Significância	Diferença	+/- Limites
HR - A-Teams	*	0,1532	0,0748
HR - IG+MDL	*	0,5142	0,0748
A-Teams - IG+MDL	*	0,3611	0,0748

de confiança de 95%, esta diferença é indicada por um $(*)$ na coluna "significância". Nesta tabela observamos que os três pares de heurísticas apresentam diferenças significativas.

A mesma análise também pode ser observada na Figura 4. Esta figura mostra o gráfico de médias com os intervalos da diferença honestamente significantes de Tukey (HSD) com um nível de confiança de 95%. Desde que o intervalo do algoritmo *IG+MDL* não se sobrepõe com nenhum outro intervalo, a média do *IG+MDL* apresenta uma diferença estatística significativa com respeito às outras heurísticas.

4.3. Análise do tempo computacional

O experimento descrido na seção anterior foi realizado considerando a qualidade da solução com relação ao valor da função objetivo. Porém, é também importante considerar o desempenho das heurísticas considerando os tempos computacionais gastos. De igual forma, foi realizada um teste ANOVA onde a variável em estudo foi o tempo computacional. Para cada heurística calculam-se as medias do *RPD*. Desde que o valor-p encontrado pelo ANOVA é 0.0000, o qual é menor que 0.05, existe uma diferença estatística significativa entre todos os tempos computacionais das heurísticas com um nível de confiança de 95%. A Tabela 2 mostra o resultado do teste de comparações múltiplas. Nesta tabela pode-se observar que existe diferenças significativas entre todo os pares da heurística. Este resultado indica que, com um nível de confiança de 95%, a heurística *IG+MDL* gastou o menor tempo computacional. Na Figura 5 podemos observar também a diferença estatística significativa mostrada pelos intervalos da diferença honestamente significativos de Tukey (HSD) com um nível de confiança de 95%. Note que o *IG+MDL* possui os menores tempos computacionais.

Tabela 2: Teste de comparações múltiplas para o RPD (tempo)

Contraste	Significância	Diferença	+/- Limites
HR - A-Teams	*	1494	249,1
HR - IG+MDL	*	1683	249,1
A-Teams - IG+MDL	*	189	249,1

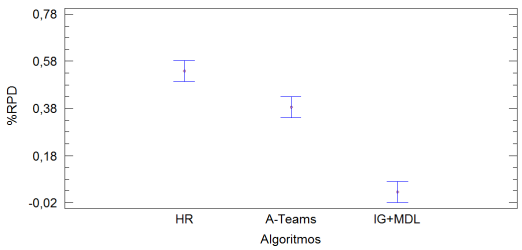

Figura 4: Gráfico de média e intervalos de Tukey (HSD) para avaliar a qualidade dos algoritmos

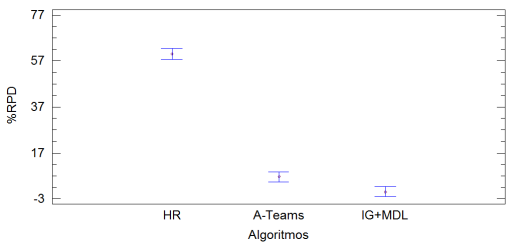

Figura 5: Gráfico de média e intervalos de Tukey (HSD) para avaluar o tempo computacional dos algoritmos

5. Conclusões

Neste artigo foi abordado o problema de dimensionamento e sequenciamento de lotes em um ambiente de produção flow shop. Para determinar as sequências de produção e as dimensões dos lotes de cada período do horizonte de planejamento, foi proposto uma heurística que combina a heurística Iterated Greedy (*IG*) com estratégias para a melhoria do dimensionamento de lotes (*MDL*). O desempenho da heurística proposta foi avaliado considerando instâncias com até 60 produtos e 20 períodos. As instâncias maiores utilizadas na literatura possuem somente 15 produtos e 15 períodos. Através de um projeto de experimentos, a melhor configuração de parâmetros para a heurística proposta foi determinada. Após dos testes computacionais e análises estatísticos realizados pode-se concluir que a heurística proposta obtém um excelente desempenho superando as duas heurísticas da literatura, *A-Teams*e *HR*.

Como pesquisas futuras sugere-se testar outros métodos para determinar o dimensionamento de lotes e também aplicar a heurística proposta para problemas de dimensionamento e sequenciamento de lotes com outros ambientes de produção.

Agradecimentos

Os autores agradecem ao CNPq, FAPEMIG e à CAPES pelo apoio financeiro ao desenvolvimento e publicação deste trabalho.

Referências

Almada-Lobo, B., Klabjan, D., Antónia M., e Oliveira, F. (2007). *Single machine multi-product capacitated lot sizing with sequence-dependent setups*, Taylor & Francis, International Journal of Production Research, 45(20), 4873–4894.

Almada-Lobo, B. e James, J. (2010). *Neighbourhood search meta-heuristics for capacitated lot-sizing with sequence-dependent setups*, Taylor & Francis, International Journal of Production Research, 45(3), 861–878.

Belo-Filho, M., Santos M. e Meneses, C. (2012). *Asynchronous teams for joint lot-sizing and scheduling problem in flow shops*, Taylor & Francis, International Journal of Production Research. 50(20), 5809–5822.

Barany, I., Van Roy, J. e Wolsey, A. (1984). *Strong formulations for multi-item capacitated lot sizing*, INFORMS, Management Science, 30(10), 1255–1261.

Drexl, A. e Kimms, A. (1997). *Lot sizing and scheduling survey and extensions*, Elsevier, European Journal of Operational Research, 99(2), 221–235.

Ik-Soo, S., Hyeok-Chol, K., Hyoung-Ho, D. e Dong-Ho L. (2011). *A two-stage heuristic for single machine capacitated lot-sizing and scheduling with sequence-dependent setup costs*, Computers & Industrial Engineering. 61(4), 920 - 929.

Karimi, B., Ghomi, F. e Wilson, J. (2003). *The capacitated lot sizing problem: a review of models and algorithms*, Elsevier, Omega, 31(5), 365–378.

Mohammadi, M., Ghomi, F., Karimi, B. e Torabi, S. (2010a). *Rolling-horizon and fix-and-relax heuristics for the multi-product multi-level capacitated lotsizing problem with sequence-dependent setups*, Springer, Journal of Intelligent Manufacturing, 21(4),501–510.

Mohammadi, M., Ghomi, F., Karimi, B. e Torabi, S. (2010b). *A new algorithmic approach for capacitated lot-sizing problem in flow shops with sequence-dependent setups*, Springer, The International Journal of Advanced Manufacturing Technology, 49(1-4), 201–211.

Mohammadi, M., Ghomi, F., Karimi, B. e Torabi, S. (2010c). *MIP-based heuristics for lotsizing in capacitated pure flow shop with sequence-dependent setups*, Taylor & Francis, International Journal of Production Research, 48(10),2957–2973.

Montgomery, D, *Design and Analysis of Experiments*, Wiley, New york, 2003.

Nawaz, M., Enscore, E. e Ham, I. (1983). *A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem*, Elsevier, Omega, 11(1), 91–95.

Pochet, Y. e Wolsey, A., *Production planning by mixed integer programming*, Springer, New york, 2006.

Ponnambalam, S. e Reddy, M. (2003). *A GA-SA multiobjective hybrid search algorithm for integrating lot sizing and sequencing in flow-line scheduling*, Springer, The International Journal of Advanced Manufacturing Technology, 21(2), 126–137.

Ramezanian, R., Saidi-Mehrabad, M. e Teimoury, E. (2012). *A mathematical model for integrating lot-sizing and scheduling problem in capacitated flow shop environments*, Springer, The International Journal of Advanced Manufacturing Technolog., 1–15.

Sikora, R. (1996). *A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line*, Elsevier, Computers & Industrial Engineering, 30(4), 969–981.

Sikora, R., Dilip, C. e Michael, J. (1996). *Integrating the lot-sizing and sequencing decisions for scheduling a capacitated flow line*, Elsevier, Computers & Industrial Engineering, 30(4), 659 - 679.

Smith-Daniels, V. e Ritzman, L. (1988). *A model for lot sizing and sequencing in process industries*, Taylor & Francis, The International Journal Of Production Research, 26(4), 647–674.

Tasgetiren, L., Yun-Chia, M. e Gencyilmaz, G. (2007). *A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem*, Elsevier, European Journal of Operational Research. 177(3), 1930–1947.

Zhu, X. e Wilhelm, E. (2006). *Scheduling and lot sizing with sequence-dependent setup: A literature review*, Taylor & Francis, IIE transactions, 38(11), 987–1007.