UM ALGORITMO ITERATED GREEDY PARA PROGRAMAÇÃO DA PRODUÇÃO EM AMBIENTES CONTROLADOS PELO SISTEMA PERIOD BATCH CONTROL

Fabio Molina da Silva (Dep-UFSCar)
fabio@dep.ufscar.br
Roberto F Tavares Neto (Dep-UFSCar)
tavares@dep.ufscar.br

SISTEMAS DE COORDENAÇÃO DE ORDENS DE PRODUÇÃO E COMPRA SÃO RESPONSÁVEIS PELA PROGRAMAÇÃO DAS ORDENS DE PRODUÇÃO E DE COMPRA. SUA UTILIZAÇÃO ADEQUADA INFLUENCIA DIRETAMENTE NO DESEMPENHO DOS SISTEMAS PRODUTIVOS. ESTE TRABALHO DESENVOLVE UM CONJUNTO ALGORITMOS PARA A PROGRAMAÇÃO DA PRODUÇÃO DE SISTEMAS PRODUTIVOS QUE OPERAM SOB O CONTROLE DO SISTEMA PBC. AS PROPOSTAS APRESENTADAS BUSCAM MINIMIZAR DOIS CRITÉRIOS: I) ATRASO TOTAL DOS PEDIDOS EM CARTEIRA E II) CAPACIDADE OCIOSA DO SISTEMA PRODUTIVO, RESPECTIVAMENTE, OS OBJETIVOS SÃO TRATADOS COMO PRIMÁRIO E SECUNDÁRIO. O TRABALHO RESULTOU EM UMA HEURÍSTICA CONSTRUTIVA ROBUSTA E UMA HEURÍSTICA BASEADA EM IG PARA RESOLVER O PROBLEMA QUANDO COMPARADAS COM A SOLUÇÃO PROPOSTA COM O MODELO EXATO.

Palavras-chave: Period Batch Control; Programação da Produção; Ambiente Make-to-Order; Programa Mestre de Produção;

1. Introdução

O Sistema de Controle de Lotes Periódicos (do inglês, *Period Batch Control*- PBC) é um sistema de programação e controle da produção que se baseia na definição da demanda de produtos acabados no nível de Programa Mestre da Produção (PMP) para realizar o cálculo das necessidades de componentes e de materiais para a programação da produção (BURBIDGE, 1996). No PBC, os períodos de produção dos estágios produtivos são projetados para terem a mesma duração.

O PBC possui uma política de controle única e simples. Esse sistema é considerado de ciclo único e de fase única, respectivamente, todos os componentes de um determinado período do PMP têm suas emissões/liberações de ordens em um único período, e de fase única, porque todos os itens produzidos são consumidos dentro de seu ciclo de produção (lead-time de produção). Essa política de controle do PBC é destacada em Benders e Riezebos (2002), Riezebos (2010, 2011), Severino et al. (2010) e Fernandes et al. (2012) como relevante para os sistemas produtivos contemporâneos, especialmente aqueles que visam os modernos paradigmas de manufatura, como por exemplo, manufatura enxuta, manufatura responsiva e manufatura ágil. Entretanto, como salientam Benders e Riezebos (2002) o PBC não tem uma divulgação ampla na academia.

Diante da utilização maciça dos sistemas corporativos que implementam o sistema MRP II (*Manufacturing Resource Planning*) pelas empresas manufatureiras, apesar do MRP e PBC serem sistemas diferentes em sua concepção, existe a possibilidade de ajustes no chão de fábrica e nos parâmetros do MRP II para funcionarem conforme o PBC obtendo melhores benefícios da automatização.

Este artigo apresenta um algoritmo baseado na meta-heurística *Iterated Greedy* (IG) para a elaboração de PMP em ambientes de produção sob encomenda. O algoritmo proposto tem como objetivos minimizar o atraso total (*total tardiness*) da programação dos pedidos firmes e maximizar a utilização dos estágios produtivos, respectivamente, critério primário e secundário.

Sumariamente, este trabalho consiste em responder a seguinte pergunta: Considerando o mínimo de atraso total dos pedidos de produção em um ambiente *make-to-order* qual a programação da produção que melhor utiliza os recursos produtivos?

A elaboração do algoritmo heurístico desenvolvido neste trabalho está fundamentada no modelo de programação matemática apresentado em Silva, Tavares Neto e Fernandes (2015) o qual serviu de base para compreensão e avaliação do problema. Para validação do algoritmo realizou uma análise com trezentos casos de teste comparando o desempenho do algoritmo heurístico ao modelo matemático desenvolvido em Silva, Tavares Neto e Fernandes (2015).

Em uma extensa revisão bibliográfica sobre o sistema de coordenação de ordens PBC realizada em Silva (2014) constatou que este trabalho é inédito e está alinhado com a tendência de recentes publicações no tema.

Este trabalho apresenta uma breve revisão bibliográfica sobre a meta-heurística IG na seção 2, na seção 3 é detalhado o problema, incluindo o modelo de programação matemática adotado, na seção 4 é apresentado o algoritmo IG, os testes computacionais estão apresentados na seção 5 e considerações finais na seção 6.

2. Meta-heurística IG

As meta-heurísticas se apresentam como uma alternativa de desenvolvimento de técnicas de resolução de problemas combinatórios. As meta-heurísticas agem como padrões de implementação, que são adaptadas para a resolução de diversos problemas. Como exemplos de meta-heurísticas, temos o Algoritmos Genético (*Genetic Algorithms* - GA - ver, por exemplo, Goldberg (1989)), a Otimização por Enxame de Partículas (*Particle Swarm Optimization* – PSO - ver, por exemplo Kennedy (2010)), Otimização por Colônia de Formigas (*Ant Colony Optimization* - ACO – ver, por exemplo Dorigo; Maniezzoe Colorni (1996)).

Nesse sentido, uma meta-heurística que vêm se mostrando eficaz, simples e rápida para a resolução de problemas combinatórios é o *Iterated Greedy* (IG). O IG vem sendo aplicado em diversos problemas como o problema de mochila múltipla (*Multiple Knapsacks* – Garcia-Martinez; Rodriguez e Lozano (2014)), minimização de atraso total e atraso total ponderado em máquina única com *setup* dependente de sequência Ying; Lin e Huang (2009), problemas de sequenciamento da produção em ambientes *flowshop* como visto nos trabalhos Ruiz e Stutzle (2007) e Ruiz e Stutzle (2008).

Essa técnica é composta de 3 fases principais: inicialização, destruição e reconstrução. O desempenho do Iterated Greedy está fortemente relacionado com a escolha dos procedimentos aplicados em cada uma dessas fases.

Na fase de inicialização, uma solução inicial é gerada usando alguma técnica específica para o problema (normalmente uma heurística construtiva de baixa complexidade computacional e que apresente soluções de qualidade). Por exemplo, Ruiz e Stutzle (2007) se utilizam do algoritmo NEH Nawaz; Enscore e Ham (1983) para obter a solução inicial do problema de minimização do *makespan* em um ambiente de *flowshop* permutacional.

A fase de destruição consiste da remoção de um sub-conjunto dos elementos presentes na solução atual. A literatura traz um conjunto de estratégias para a desconstrução da solução (como por exemplo, Rodriguez; Lozano e Garcia-Martinez (2013)). Porém, percebe-se que a estratégia de se remover de forma aleatória alguns elementos da solução é muito utilizada pela literatura (por exemplo, ver Ruiz e Stutzle (2007) e Ruiz e Stutzle (2008)). Nesse sentido, é interessante perceber que Rodriguez; Lozano e Garcia-Martinez (2013) relata que as heurísticas propostas para a destruição não conseguiram, em todos os casos de teste, superar a estratégia aleatória simples.

A fase de reconstrução é, em grande maioria, uma heurística construtiva simples. Por exemplo, Ruiz e Stutzle (2007), Ruiz e Stutzle (2008) e Pan; Wang e Zhao utilizaram parte da heurística NEH;

Adicionalmente, três etapas também são relatadas na literatura: a busca local, o critério de aceitação e o critério de parada. Embora seja sempre citado como um estágio opcional, todos os artigos analisados possuíam um estágio de busca local.

O critério de aceitação é responsável pela substituição da candidata a melhoria. A estratégia de implementação mais simples desse estágio é um algoritmo que substitui a solução apenas em caso de melhoria da função objetivo do problema. Essa estratégia, segundo Ruiz e Stutzle (2007), pode levar à estagnação prematura da solução, porém, alguns autores (por exemplo, Ruiz e Stutzle (2007), Ruiz e Stutzle (2008), Rodriguez; Lozano e Garcia-Martinez (2013) e Pan; Wang e Zhao) utilizam de uma abordagem onde uma função com componente aleatório permite a aceitação de soluções que não tragam melhoria na função objetivo.

O critério de parada indica quantas iterações serão realizadas pelo algoritmo. São critérios comuns na literatura: (i) a execução de um número pré-determinado de vezes do algoritmo, ou (ii) a execução do algoritmo até que uma quantidade pré-determinada de tempo computacional seja utilizado Ying; Lin e Huang (2009).

O algoritmo pode ser então descrito da seguinte forma:

Passo 1: Gere uma solução inicial usando, por exemplo, uma heurística construtiva e a estabeleça como solução atual.

Passo 2: Desconstrua a solução atual, criando uma solução parcial com um conjunto de elementos removidos.

Passo 3: Usando uma heurística construtiva, crie uma nova solução adicionando novos elementos na solução parcial. Essa será a nova solução.

Passo 4 (opcional): Aplique um algoritmo de busca local para melhorar a nova solução.

Passo 5: Respeitando o critério de aceitação, substitua a solução atual pela nova solução.

Passo 6: Respeitando o critério de parada, termine o algoritmo ou volte ao Passo 2.

3. Definição do problema e programação matemática

Este problema definido em Silva, Tavares Neto e Fernandes (2015) definiram que o problema consiste em determinar um PMP para *n* períodos em um ambiente *make-to-order* operando com o PBC. Esse PMP tem como principal objetivo minimizar o atraso total dos pedidos em carteira e o objetivo secundário de alocar a maior carga produtiva aos estágios produtivos. Há restrições de capacidade produtiva nos estágios produtivos controlados pelo PBC que devem ser respeitadas. Todo pedido possui tempo de processamento para cada estágio produtivo. Na modelagem matemática, Silva, Tavares Neto e Fernandes (2015) utilizaram de um período de programação virtual com capacidade de produção infinita o qual pedidos alocados a este período de programação virtual significa que de fato esses pedidos não serão programados, ou seja, representa os pedidos que continuarão em compondo a carteira de pedidos em aberto.

Durante apresentação dos modelos e heurísticas são utilizados índices, parâmetros e variáveis. Esta seção apresenta todos os símbolos que são usados neste artigo para definir o problema estudado:

Índices

c: número efetivo de ciclos da programação (não inclui o ciclo denominado de buffer de capacidade infinita)

P: número de pedidos;

N: número de estágios produtivos;

HP = N+c: tamanho do horizonte de programação em períodos;

M: Número muito grande;

j: índice dos períodos de programação, j = 1, 2, 3, ...,HP;

i: índice dos pedidos, i = 1, 2, ...,P;

w: índice dos estágios produtivos, w = 1,..., N;

Parâmetros

TP_{iw} : Tempo de processamento do pedido i no setor produtivo w;

CP_{wj} : Capacidade produtiva do setor w no período j;

d_i : Data de entrega do pedido i;

Variáveis Positivas

A_i : atraso do pedido i;

CO_{wj} : Capacidade Ociosa do setor w no período j;

COR_{wj} : Capacidade Ociosa Real do setor w no período j;

CR_{wj} : Crédito de tempo do banco de horas do setor w no período j;

CRR_{wj} :Crédito de tempo real do banco de horas;

Variáveis Binárias

$$y_j = \begin{cases} \text{y} = 1 \text{se houver pedido alocado ao período } j \\ \text{y} = 0 \text{caso contrário} \end{cases}$$

$$x_{ij} = \begin{cases} 1 \text{se o pedido } i \text{ for alocado ao PMP no período } j \\ 0 \text{caso contrário} \end{cases}$$

O modelo matemático a seguir minimiza a capacidade ociosa dos setores produtivos controlados pelo sistema PBC, após garantir o mínimo atraso total dos pedidos em carteira. Para isso são realizadas duas etapas: na primeira etapa, o modelo realiza a programação do PMP com a função objetivo de minimizar o atraso total dos pedidos. Na segunda etapa, três equações são adicionadas, sendo uma delas que o atraso total dos pedidos não seja superior ao valor encontrado na iteração anterior. O modelo é composto pelas seguintes equações:

$$Min \sum_i (A_i) = z1 \qquad (1)$$

Sujeito a:

$$\sum_{j}\left(x_{ij}\right)=1,\forall\, i; \quad (2)$$

$$\sum_{j\le N-1}\left(x_{ij}\right)=0,\forall\, i; \quad (3)$$

$$\sum_{i}\left(TP_{iw}.x_{ij}\right)+CO_{wj-(N-w)}=CP_{w,j-(N-w)},\forall\, wj \mid N \le j < HP; \quad (4)$$

$$A_i \ge j.x_{ij}-d_i,\forall\, ij \quad (5)$$

A equação 1 é a função objetivo da primeira etapa, a qual minimiza o total de atraso dos pedidos programados. A equação 2 garante que todo pedido será alocado a um e somente um período do PBC. A equação 3 garante que nenhum pedido será alocado em período menor do que a antecedência necessária para sua produção. A equação 4 garante a restrição de capacidade/carga de cada setor w em cada período j. Nessa equação, a variável de folga $CO_{w,j}$ armazena a capacidade ociosa do setor produtivo w no período j em unidades de tempo. A equação 5 determina o atraso de cada pedido i.

Na segunda etapa, a função objetivo da etapa 1, mostrada na equação 1, é substituída pela equação 6, a qual minimiza a somatória da capacidade ociosa real (COR). Essa variável foi incluída ao modelo para não penalizar a função objetivo quando a capacidade produtiva for superior a carteira de pedido. Nesse caso, a ociosidade não seria oriunda da qualidade da programação da produção, mas da falta de demanda. Ao modelo da primeira etapa, também são adicionadas as equações 7, 8 e 9. As equações 7 e 8 garantem que, se houver pedido alocado ao período (y_j), a variável correspondente a capacidade ociosa real (COR) será igual a capacidade ociosa (CO), caso contrário, a variável COR será zero. A equação 9 representa a inclusão do limitante superior à somatória de atraso dos pedidos. Desse modo, a segunda etapa otimiza a utilização da capacidade garantindo mínimo atraso total.

$$Min\sum_{i}\left(COR_{wj}\right)=z2 \quad (6)$$

$$y_j * M \ge \sum_{i} x_{ij},\forall\, j \quad (7)$$

$$COR_{wj-N-w} \ge CO_{wj-N-w}-M*\left(1-y_j\right),\forall\, wj \mid N \le j < HP \quad (8)$$

$$\sum_i (A_i) \leq z1 \qquad (9)$$

Em suma, a primeira etapa tem como função objetivo a equação 1, sujeito as restrições das equações 2, 3, 4 e 5. A segunda etapa tem como função objetivo a equação 6, sujeito as restrições 2, 3, 4, 5, 7, 8 e 9. Com isso, o modelo analítico indicará a melhor solução de alocação dos pedidos em carteira aos períodos do PMP que garante o mínimo atraso total e maximiza a utilização da capacidade.

Durante a resolução do modelo pelo solver comercial GAMS/CPLEX, constatou-se que, em problemas de maior escala, o modelo matemático demandou uma quantidade proibitiva de tempo computacional para a obtenção da solução ótima. Esse fato direcionou a presente pesquisa para o desenvolvimento da técnica heurística baseada no IG, que é detalhada a seguir.

4. Algoritmo IG

Como explicado na revisão bibliográfica, a meta-heurística IG é composta em 3 principais fases: i) inicialização, ii) destruição e iii) reconstrução. A fase inicialização fornece uma solução inicial ao problema, geralmente, baseado em um algoritmo construtivo específico para o problema. A fase de destruição consiste em destruir parte da solução inicial por meio da remoção de um subconjunto de decisões da solução atual. A fase de reconstrução corresponde à inclusão de um novo subconjunto de decisões.

Como heurística construtiva para resolver o problema mencionado na seção anterior, utilizou-se de duas regras clássicas de *scheduling* denominada EDD (*Earliest Due Date* – sequencia as ordens de serviço em ordem crescente de tempo de processamento) e LTP (*Longest Processing Time* – sequencia as ordens em ordem decrescente dos tempos de processamento) para iniciar a ordenação dos pedidos em carteira. Essa regra de ordenação foi denominada de EDD Modificado. Nessa regra, os pedidos são ordenados de forma crescente pela data de entrega e, no caso de empate, utiliza-se a regra LPT. A seguir, de forma simplificada, apresenta-se o algoritmo.

Ordene os Pedidos pela regra EDD Modificada
Para cada Pedido não alocado faça
    Período = Tempo de Ciclo
    Enquanto Pedido não alocado e Período menor que Horizonte de Programação

```
        Alocar pedido ao período
      Se pedido não alocado
        Período = Período + 1
    Fim do enquanto
  Fim do para
```

Nesse algoritmo, a instrução da linha de número 5 – Alocar pedido ao período – realiza a operação de verificar a disponibilidade de cada setor produtivo, considerando sua antecipação em relação ao período de programação, para acolher a carga de trabalho do pedido atual e, em caso positivo, o pedido é alocado ao período atualizando o nível de capacidade de cada setor produtivo. O conjunto de comandos contidos entre as linhas de número 2 e 9 do algoritmo acima será encapsulado em um procedimento denominado de Alocar Pedidos.

O algoritmo construtivo apresentado, EDD Modificado, é adotado na fase de inicialização dos algoritmos heurísticos baseados na meta-heurística IG como apresentado a seguir:

```
Inicialização: Solução Atual igual a alocação com o Algoritmo EDD modificado
Repita enquanto não alcançar a condição de parada
    Destruir parte da Solução Atual utilizando método indicado
    Reconstruir Nova Solução utilizando método indicado
    Alocar Pedidos
    Atualizar Soluções se houver melhorias
Fim do Repita
```

No presente trabalho adotou-se como condição de parada de execução do algoritmo o número de iterações. O critério de aceitação da nova solução é o menor atraso total ou apresentar um menor atraso total igual ao da solução anterior com melhor aproveitamento da capacidade produtiva. Para as fases de destruição foram utilizados os seguintes métodos:

- Destruição aleatória (D1): Do conjunto de Pedidos Programados, escolhe-se aleatoriamente, com chances iguais, um subconjunto de pedidos que será retirado da programação (ou seja, produzido a posteriori);

- Destruição por maior tempo de processamento (D2): Entre os Pedidos Programados, remove o subconjunto de pedidos que mais utilizam capacidade produtiva.

Para a fase de Reconstrução foram elaborados os seguintes métodos:

- Reconstrução aleatória (R1): Do conjunto de Pedidos Não Programados é escolhido, aleatoriamente com chances iguais de escolha, um pedido para ser programado ao período mais recente possível (*Batch First Fit*).

- Reconstrução pelo menor tempo processamento (R2): Do conjunto de Pedidos Não Programados é escolhido o pedido com o menor tempo de processamento total para ser programado ao período mais recente possível.

- Reconstrução usando a regra *Batch First Fit* (R3): Do conjunto de Pedidos Removidos, ordenados decrescente pelo tempo total de processamento, cada pedido é alocado ao período mais recente possível.

- Reconstrução usando a regra *Batch First Fit* modificada (R4): Para cada pedido do conjunto de Pedidos Removidos, ordenados decrescentemente pelo tempo total de processamento, é alocado ao melhor período possível para reduzir o atraso: primeiramente, tenta-se uma programação para trás, tentando alocar o pedido no período mais tarde possível de forma que não ocorra atraso. Em caso de insucesso da programação para trás do pedido, é realizada a programação para frente, alocando o pedido no momento em que ocorre o menor atraso.

- Reconstrução priorizando pedidos com maior carga no setor com maior capacidade remanescente (R5): Enquanto houver capacidade, busca-se entre os Pedidos Não Programados aquele que possui maior utilização de capacidade no setor de maior capacidade ociosa.

Com esses métodos elaborados, foram montados os seguintes procedimentos apresentadas no Quadro 1.

Quadro 1- Heurísticas Iterated Greedy para o problema com capacidade constante

Nome	Inicialização	Destruição	Reconstrução
H1A	EDD Modificada	D1	R5
H1B	EDD Modificada	D1	R1
H1C	EDD Modificada	D2	R5
H1D	EDD Modificada	D2	R2
H1E	EDD Modificada	D1	R3
H1F	EDD Modificada	D1	R4
H1G	EDD Modificada	D1	R2

D1= Aleatória; D2 = Maior tempo de processamento; R1= Aleatório; R2 = menor tempo de processamento; R3= Batch First Fit; R4 = Batch First Fit Modificada; R5 = Pedidos com maior carga no setor com maior capacidade remanescente.

5. Resultados computacionais

Infelizmente, a revisão bibliográfica realizada não revelou nenhum conjunto de casos de teste que pudesse ser utilizado neste trabalho. Por isso, para analisar o comportamento do modelo de programação matemática e das heurísticas propostas, foram gerados casos de teste com n = 30, 50, 100, 200 ou 300 pedidos em carteira e w = 3, 5 ou 7 (estágios produtivos). Os períodos de programação para os casos de teste foram 2 e 4 períodos a frente. Para cada combinação desses parâmetros gerou-se 10 casos de testes, totalizando um conjunto 300 casos de teste.

As 7 heurísticas foram implementadas utilizando a linguagem de programação C++. As heurísticas que possuem comportamentos estocásticos foram executadas 30 vezes, e as melhores soluções foram obtidas.

As heurísticas foram executadas em um microcomputador com processador i3 com 4 gigabytes de memória RAM utilizando o sistema operacional Windows 7. O tempo de execução das heurísticas não ultrapassou a um minuto.

Os casos de teste foram executados, e dois parâmetros foram obtidos: (i) o gap do atraso, definido como (atraso obtido pela heurística – atraso obtido pelo GAMS/CPLEX) / (atraso obtido pelo GAMS/CPLEX + 1); e (ii) o gap do total de capacidade remanescente de cada período, definido como (capacidade remanescente obtida pela heurística – capacidade remanescente obtida pelo GAMS/CPLEX) / (capacidade remanescente obtida pelo GAMS/CPLEX + 1). Os resultados apresentados na Tabela 1 referem-se a média do gap de atraso e do desvio padrão do gap em percentuais.

Conforme Figura 1 e Tabela 1, de todas as estratégias propostas a heurística H1F se mostrou mais eficiente tanto em termos de redução do atraso total quanto ao melhor aproveitamento da capacidade produtiva. Os dados apresentados refletem o desvio em percentuais do resultado obtido pelo modelo exato.

Tabela 1- Análise das heurísticas - Média e Desvio Padrão

	Atraso		Capacidade Ociosa	
	Média	Desvio Padrão	Média	Desvio Padrão
EDD	0,058	0,079	123,21	2022,02
H1A	0,058	0,079	123,18	2022,12
H1B	0,058	0,079	122,61	2017,71
H1C	0,055	0,076	114,69	1909,18
H1D	0,046	0,075	119,06	1972,08
H1E	0,055	0,077	94,53	1635,08
H1F	0,041	0,058	62,21	1042,89
H1G	0,058	0,078	120,40	2005,86

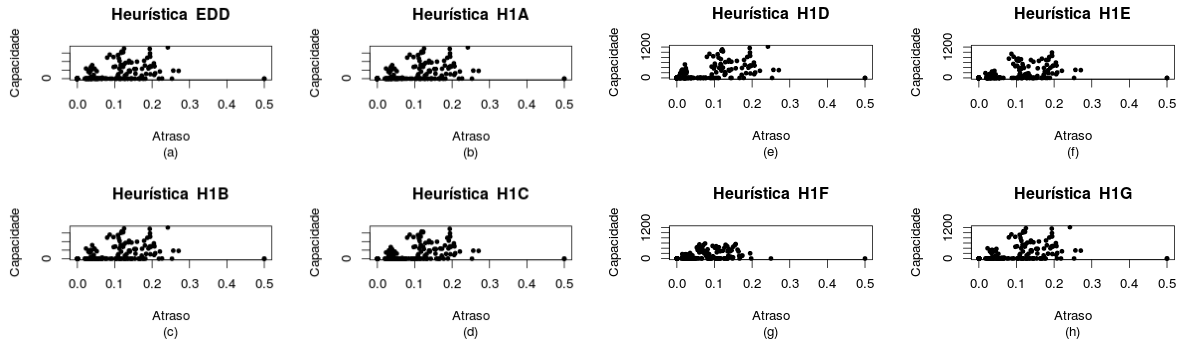

Figura 1- Análise da execução das heurísticas

As análises dos atrasos encontrados por cada heurística são mostradas na Figura 2. Em tal figura, percebe-se claramente que as heurísticas H1D e H1F possibilitaram uma maior redução médio de atraso, assim como desvios padrões menores.

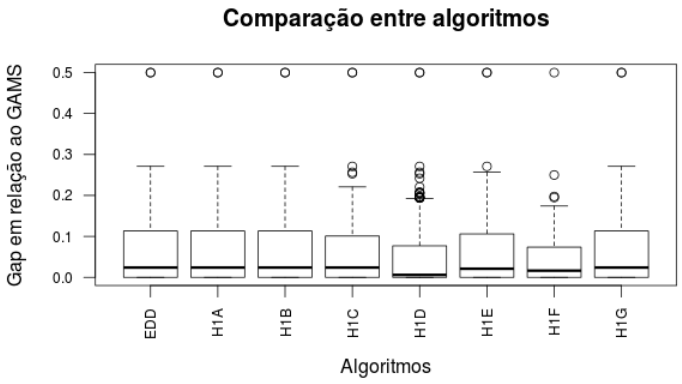

Figura 2- Comparação entre os algoritmos em termos de atraso total

Ao comparar a H1D com H1F percebe-se que a heurística H1D apresentou uma maior dificuldade com o uso da capacidade instalada quando comparado com a heurística H1F. Ao comparar a utilização da capacidade da heurística H1D com a heurística de inicialização

(EDD), evidencia-se que a melhora desta heurística foi significante em termos do atraso total (Figura 3)

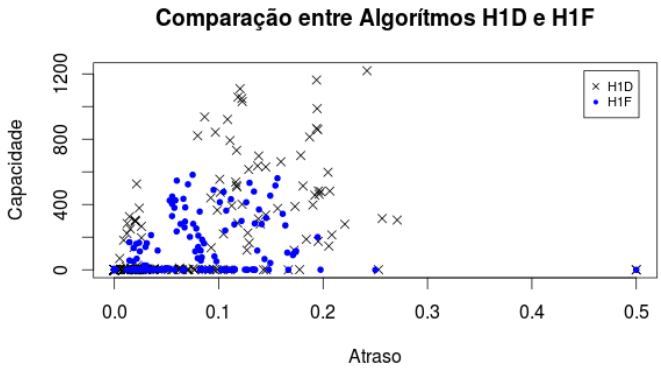

Figura 3- Comparação entre os Algorítmos H1D e H1F

6. CONSIDERAÇÕES FINAIS

O presente trabalho apresentou um modelo de programação linear mista e um conjunto de 8 heurísticas IG para resolver o problema de minimização de atraso e capacidade ociosa em um sistema produtivo controlado pelo PBC. Em testes computacionais a heurística H1F demonstrou desempenho superior às demais com o conjunto de casos de testes realizado. Cabe ressaltar que a heurística também apresentou resultado aceitável quando comparado com a solução ótima.

A heurística EDD modifica apresentou resultado robusto como método de iniciação para o algoritmo IG, pois poucas heurísticas melhoraram a solução inicial.

Como trabalho futuros, espera-se focar em variações desse cenário produtivo e na sua implementação como parte dos processos decisórios de empresas.

REFERÊNCIAS

BENDERS, J.; RIEZEBOS, J.; Period Batch control: classic, not outdated. Production Planning & Control: The Management of Operations, v. 13, n. 6, p. 497-506, 2002.

BURBIGDE, J. L. Period Batch Control, Oxford: Clarendon Press, 1996.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. The ant system: Optimization by a colony of cooperating agentes. IEEE Transactions on Systemas, Man, and Cybernetics-Part B, v. 26, p. 29-41, 1996.

FERNANDES, F. C. F.; GRACIA, E.; SILVA, F. M.; GODINHO FILHO, M. Proposta de um método para atingir a manufatura responsiva na indústria de calçados: implantação e avaliação por meio de uma pesquisa-açõa, Gestão & Produção, v. 19, n. 3, p. 509-529, 2012.

GARCIA-MARTINEZ, C.; RODRIGUEZ, F.; LOZANO, M. Tabu-enhanced iterated greedy algorithm: A case study in the quadratic multiple knapsack problem. European Journal of Operational Research, v. 232, n. 3, p. 454 – 463, 2014.

GOLDBERG, M. C. Genetic Algorithms in Search, Optimization & Machine Learning. Addison Wesley Longman Inc., 1989.

KENNEDY, J. Particle Swarm Optimization, Encyclopedia of Machine Learning, Springer US, p. 760-766, 2010.

NAWAZ, M.; ENSCORE, E.; HAM, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega, v. 11, p. 91–95, 1983.

PAN, Q.-K.; WANG, L.; ZHAO, B.-H. An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. International Journal of Advanced Manufacturing Technology, v. 38, n. 7-8, p. 778 – 786, 2008.

RIEZEBOS, J. Order release in synchronous manufacturing. Production Planning & Control: The Management of Operations. v. 21, n. 4, june 2010, p. 247-258, 2010.

RIEZEBOS, J. Order sequencing and capacity balancing in synchronous manufacturing. International Journal of Production Research. v. 49, n. 2, p. 531-552, 2011.

RODRIGUEZ, F. J.; LOZANO, M.; BLUM, C.; GARCíA-MARTíNEZ, C. An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem. Computers & Operations Research, v. 40, n. 7, p. 1829 – 1841, 2013.

RUIZ, R.; STUTZLE, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research, v. 177, n. 3, p. 2033 – 2049, 2007.

RUIZ, R.; STUTZLE, T. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. European Journal of Operational Research, v. 187, n. 3, p. 1143 – 1159, 2008.

SEVERINO, M. R.; LAGE JUNIOR, M.; CAMPANINI, L.; GUIMARÃES, A. A.; GODINHO FILHO, M.; AGUILERA, M. Proposta de utilização do sistema Period Batch Control para redução de lead time em uma empresa de bens de capital, Produção, v. 20, n. 4, out./dez., p. 612-625, 2010.

SILVA, F. M. Estratégias para a Programação da Produção em Ambientes com Capacidade Flexível Controladospelo Sistema PBC. Tese de doutoramento, CCET UFSCar, 2014.

STEELE, D. C.; MALHOTRA, M. K. Fators affecting performance of period batch control systems in cellular manufacturing, International Journal of Production Research, v. 35, n. 2, p. 421-446, 1997.

YING, K.-C.; LIN, S.-W.; HUANG, C.-Y. Sequencing single-machine tardiness problems with sequence dependent setup times using an iterated greedy heuristic. Expert Systems with Applications, v. 36, n. 3 PART 2, p. 7087 – 7092, 2009.

SILVA, F. M.; TAVARES NETO, R. F.; FERNANDES, F. C. F.: Uma abordagem baseada em programação inteira mista para a programação da produção em ambientes controlados pelo sistema de coordenação de ordens PBC in XXII SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO: Política Nacional de Inovação e Engenharia de Produção, Bauru, SP, Brasil, novembro de 2015.