PROGRAMAÇÃO DE ORDENS DE PRODUÇÃO COM DIFERENTES INSTANTES DE LIBERAÇÃO PARA UMA DATA ÚNICA DE ENTREGA

Claudio Fernando Furlan (USP) claudio.furlan@poli.usp.br
Débora P. Ronconi (USP) dronconi@usp.br

Resumo

Este trabalho apresenta uma heurística construtiva para o problema de programação de ordens de produção com data de entrega comum para todas as ordens que serão processadas em uma única máquina. Para evitar estoques antecipados e mesmo atrasos na entrega, são aplicadas penalidades tanto para adiantamento como para o atraso. O objetivo é minimizar a somatória das penalidades aplicadas as entregas adiantadas e as atrasadas. Para isto este trabalho vem propor uma heurística adaptada do algoritmo de Sridharan e Zhou (1996). O desempenho da heurística proposta é avaliado através de um estudo comparativo com resultados ótimos obtidos em problemas de pequena dimensão.

Palavras-Chave: *Programação de produção; Heurística e Adiantamentos e Atrasos Penalizados.*

1. INTRODUÇÃO

Na década de 70, os japoneses criaram uma filosofia de trabalho chamada *Just in time* (*JIT*), que significa que todas as atividades tinham que começar e terminar no tempo correto, nem antes e nem depois, sendo a fabricante de veículos automotores Toyota sua precursora, conforme escreve Womack e Jones (1992).

No ambiente de produção o *JIT* tem sido amplamente utilizado, conforme descreve Arnold (1999), "*JIT* é a eliminação de todo desperdício e a melhoria contínua da produtividade... que um dos objetivos da programação é fazer a melhor utilização dos recursos da produção e assim influenciar a produtividade".

Portanto, a programação de atividades tem um papel importante na melhoria dos resultados quando utilizada seguindo a premissa do *JIT*. Assim utilizando esta premissa de executar tarefas no tempo correto, a programação deve balancear a execução das atividades de forma que elas não terminem antes da necessidade, assim como não sejam finalizadas depois do instante requerido.

Ilustrando os problemas de se adiantar alguma atividade, Bagchi, Chang e Sullivan (1987) comentam que no caso de produtos finalizados antes do tempo incorre uma ocupação cada vez maior do espaço destinado ao estoque e conseqüente escassez de espaço, aperto no fluxo de caixa devido ao capital parado aguardando sua data de entrega, e que geralmente indica que a alocação e utilização dos recursos não estão balanceadas, isto é, indica ineficiência da cadeia.

Em relação a atrasar uma atividade, e neste caso mais precisamente a entrega de um produto, Davis e Kanet (1993), salientam as conseqüências que este cenário pode apresentar, como a perda da reputação em relação ao cliente, custos de oportunidade de perda de vendas e até penalidade financeira por contrato de venda.

Em resumo os custos envolvidos em ambas situações demonstram a ineficiência do sistema em relação à capacidade envolvida e a programação das atividades.

Este trabalho tem como característica principal apresentar uma heurística para a programação de ordens que estarão disponíveis em diferentes datas para serem processadas. Consideramos o ambiente de máquina única e penalidades iguais para todas as ordens aplicadas ao adiantamento e atraso em relação a data de entrega estabelecida, sendo as penalidades de atraso e adiantamento diferentes entre si.

Jeffrey Sidney (1977), nos diz o seguinte em seu trabalho: "A produção de produtos perecíveis é um outro exemplo no qual estas penalidades aparecem. Suponha, por exemplo, um produtor químico combina um item químico A, que deteriora rapidamente com um segundo item químico B para produzir o item C. Se A é produzido antes que B esteja pronto, ele se deteriora. Se A é produzido mais tarde, o atraso de C pode ser dispendioso." Em seu artigo Sidney exemplifica a particularidade de ter uma única data de entrega que neste exemplo seria a da produção do item C e fica nas entrelinhas a importância das penalidades para poder ajustar o processo das ordens o mais próximo possível da data de entrega. Outro exemplo prático é citado por Kanet (1981), para o caso de produção de componentes para a montagem final de um produto, onde a data de entrega comum se refere ao início da montagem do item final.

A figura 1 a seguir mostra uma situação onde um cliente faz um pedido de alguns itens, mas com a condição de que a entrega não pode ser parcial. Então a entrega terá data única e assim os itens deverão estar todos prontos até aquela referida data. As ordens de produção (OP) são liberadas conforme a disponibilidade de matéria prima e programadas para serem processadas em uma única máquina. Não é interessante que os itens deste pedido terminem nem muito tempo antes assim como não devem atrasar. Portanto, tudo dependerá de uma boa programação.

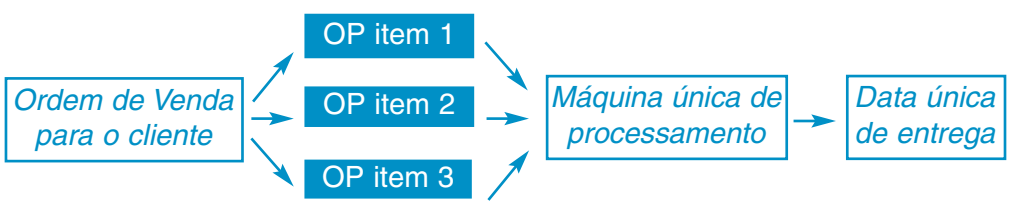

FIGURA 1 – Situação prática de entrega de vários itens para uma mesma data de entrega.

Estes exemplos refletem a importância deste trabalho, assim como as diferentes datas de liberação das ordens, pois no ambiente JIT matérias primas também devem chegar no seu tempo certo. Com isso cada ordem poderá ter uma diferente data de liberação para o processo.

Neste trabalho definiremos o problema em estudo assim como apresentando a heurística de Sridharan e Zhou (1996) modificada para atender as premissas definidas. O método de resolução é heurístico, pois de acordo com Valente e Alves (2003), este tipo de problema tem complexidade *NP-Hard*. Testes computacionais com problemas da literatura e seus resultados serão apresentados. Resultados de problemas de pequena dimensão foram comparados com uma programação ótima para mostrar a efetividade da implementação desta heurística.

A motivação para a realização deste trabalho é a ausência de artigos que abordem este problema coma as mesmas características anteriormente citadas. Principalmente quando tratamos a data de entrega d como restritiva, isto é, seu valor é menor que a soma dos tempos de processamento de todas as ordens.

2. DESCRIÇÃO DO PROBLEMA

O problema consiste de n ordens de produção com diferentes tempos de processamento p_i, sendo $i = 1...n$, que devem ser processadas por uma única máquina. Aqui podemos ressaltar que máquina única pode ser interpretada como uma célula de produção ou um centro de trabalho ou mesmo uma máquina gargalo da produção. As datas de chegada ou de liberação (r_i) das ordens para entrar na máquina são diferentes, ou seja, as ordens não estarão todas disponíveis no mesmo instante, portanto sua liberação será dinâmica ao longo do tempo. Consideraremos que os dados das ordens, tempo de processamento e data de liberação são determinísticos. Não será permitida interrupção *(preemption)* do processamento da ordem antes da sua conclusão. A programação será definida em função de minimizar os atrasos e os adiantamentos ponderados, isto é, tanto o adiantamento como o atraso terão penalidades.

O instante de término de cada ordem i será definido como C_i e o início denominado s_i, logo, $C_i = s_i + p_i$. As penalidades serão definidas como a para o adiantamento das ordens e b para o atraso e serão aplicadas sobre o tempo de adiantamento E_i e atraso T_i respectivamente. Sendo E_i=max $\{ d\text{-}C_i, 0 \}$ e T_i = max $\{C_i\text{-}d, 0\}$.

Este problema tem como particularidade data de entrega única. Neste caso podemos ter dois tipos de entrega única. O primeiro chamado de data não restritiva que significa que uma seqüência ótima pode ser construída sem que o valor da data de entrega seja considerado. O segundo caso chamado de data restrita onde a data de entrega se encontra na seguinte situação: $d < \Sigma\, p_i$. E este segundo será o caso do estudo deste trabalho.

O problema em estudo pode então ser ilustrado conforme a figura 2:

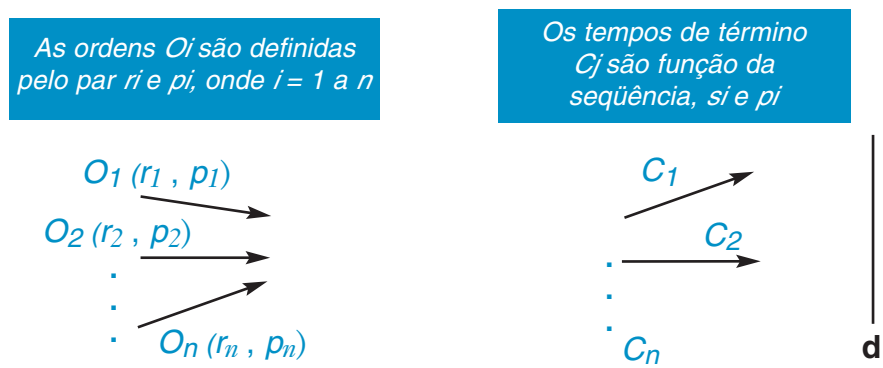

FIGURA 2 – Representação gráfica do problema.

Segue abaixo a formulação matemática do problema adaptada do trabalho de Biskup e Feldman (2001):

Função Objetivo:

$$Min \sum_{i=1}^{n} \alpha E_i + \beta T_i$$

$$T_i \geq s_i + p_i - d \qquad i = 1,..., n; \qquad (1)$$

$$E_i \geq d - s_i - p_i \qquad i = 1,..., n; \qquad (2)$$

$$s_i + p_i \leq s_k + R(1 - x_{ik}) \qquad i = 1,..., n-1;\ k = i+1,..., n; \qquad (3)$$

$$s_k + p_k \leq s_i + Rx_{ik} \qquad i = 1,..., n-1;\ k = i+1,..., n; \qquad (4)$$

$$T_i, E_i, s_i, r_i \geq 0 \qquad i = 1,..., n; \qquad (5)$$

$$x_{ik} \in \{0,1\} \qquad i = 1,..., n-1;\ k = i+1,..., n; \qquad (6)$$

$$s_i \geq r_i \qquad i = 1,..., n; \qquad (7)$$

Os tempos de adiantamento e atraso são calculados pelas restrições (1) e (2). As restrições (3) e (4) determinam os tempos de início das ordens. R é um valor grande o suficiente, definido previamente para não interferir no resultado das restrições. Se a ordem i precede a ordem k a restrição $s_i + p_i \leq s_k$ será verdadeira se $x_{ik} = 1$. Por causa da adição de R (4) não é restritiva com $xik = 1$. Por outro lado, para $x_{ik} = 0$ a restrição(4) será definido por $s_k + p_k \leq s_i$ e (3) não é restritiva. Importante notar que a variável binária é definida em (6).

A restrição (5) garante a não negatividade das variáveis e a restrição (7) define que uma ordem só pode ser iniciada se ela estiver disponível naquele instante.

Na literatura encontramos alguns artigos que levam em consideração a data comum e as penalidades por atraso e adiantamentos únicos a e b para todas as ordens, dos quais citamos alguns: Panwalkar, Smith and Seidmann (1982), Emmons (1987), Bagchi, Chang e Sullivan (1987) e Mondal e Sen (2001); mas em todos eles a data de liberação da ordem r_i é comum a todas as ordens, isto é, todas estão disponíveis para processamento no início da programação. Artigos que levam em consideração diferentes datas de liberação podemos citar Sridharan e Zhou (1996), Bank e Werner (2001), mas que utilizam como penalidade a_i e b_i para as ordens e Nandkeolyar, Ahmed e Sundararaghavan (1993) que penalizam o atraso e o adaintamento igualmente com $a = b$, apenas o trabalho de Cheng, Chen, Shakhlevich (2002) penaliza as ordens conforme o problema proposto acima, mas a data de entrega não é dada, é uma das variáveis do problema. Um trabalho recentemente publicado é o de Valente e Alves (2003) que trabalha com as mesmas premissas do problema em estudo, mas com uma variante que é a data de entrega não restrita. Portanto, até o momento não conhecemos na literatura disponível artigos com as mesmas premissas do problema tratado neste artigo.

3. HEURÍSTICA PROPOSTA

Devido à complexidade do problema é proposta uma heurística construtiva por ela ser de rápida solução. Esta metodologia em geral apresenta menor complexidade de implementação e obtêm-se respostas com grande rapidez o que é importante em aplicações práticas.

A heurística construtiva a ser aplicada é uma adaptação da heurística apresentada no artigo de Sridharan e Zhou (1996) denominada de DT-ET, que são as iniciais de *Decision Theory for Earliness and Tardiness problem.*

Esta heurística leva em consideração uma teoria de decisão na seleção da próxima ordem a ser processada avaliando as conseqüências de cada alternativa de acordo com um critério dado e escolhe

a melhor alternativa, conforme descrito no artigo de Kanet e Zhou (1993), assim como uma visão futura das ordens que estão para chegar.

A motivação para a escolha desta heurística foi baseada na estrutura proposta por Nandkeolyar, Ahmed e Sundararaghavan (1993) que realizaram um estudo sobre o problema E/T (*Earliness and Tardiness*) para máquina única com ordens chegando dinamicamente, ou seja, ordens com diferentes datas de liberação e considerando todas as penalidades iguais. Este artigo avalia diferentes heurísticas desenvolvidas de forma modular. Um diagrama em forma de fluxo de decisão é definido utilizando os módulos. Estes módulos são divididos em: Visão Futura (FE, iniciais de *front end*), Tema Principal (MT, iniciais de *main theme*) e Inclusão de Tempo Ocioso na programação (BR, iniciais de *balancing routine*).

Estes módulos se referem a: (FE) geração de um cenário futuro considerando ordens que estão para chegar após o instante de tomada de decisão de qual ordem será processada em seguida; (MT) definido por uma regra de despacho para gerar uma seqüência de ordens liberadas que será convertida em uma programação (regras como: *EDD*, *SPT*, *FCFS*, etc) e (BR) aceitação de inserção ou não de tempo ocioso ou parada de máquina na programação das ordens.

Após vários testes comparativos entre 12 heurísticas criadas através da combinação dos módulos Nandkeolyar, Ahmed e Sundararaghavan (1993) chegaram a conclusão de que a heurística utilizando a conjunção FE+BR tem a melhor performance quando a máquina está congestionada.

A heurística construtiva desenvolvida por Sridharan e Zhou (1996) tem em sua estrutura os dois módulos FE e BR. A idéia básica da heurística está resumida nestas duas perguntas: qual a próxima ordem a ser processada e quando começar seu processamento. Para responder a primeira questão ela considera ordens a serem liberadas e as atuais na fila, e para responder a segunda ela avalia a escolha de uma ordem em termos do impacto sobre todas as outras não escolhidas.

Para isto a heurística consiste de três procedimentos básicos:

A – Definir quais ordens serão analisadas.

B – Programar uma ordem como próxima a ser processada e estimar o instante de término das restantes. Refazer este cálculo para todas as ordens definidas em A. Calcular o custo total de cada programa. Escolher a ordem de menor custo.

C – Refazer os procedimentos A e B até que todas as ordens estejam programadas.

A seguir segue o algoritmo da heurística com as definições dos parâmetros e variáveis envolvidas.

Parâmetros

p_j – tempo de processamento da ordem j;

r_j – data de chegada da ordem j;

b - penalidade única por atraso;

a - penalidade única por adiantamento;

d – data de entrega comum;

Variáveis

C_j – instante de término da ordem j em estudo;

s_j – instante de início da ordem j em estudo;

t_0 – instante de decisão;

$\mathcal{J}$ – conjunto das ordens que estão na fila;

K – conjunto das ordens que irão chegar dentro do período: [t_0, $max\ (t_0+p_j, d_j)$];

$S(t_0)$ – conjunto das ordens candidatas no instante t_0, definido pela união dos conjuntos $\mathcal{J}$ e K ($\mathcal{J} \cup K$) ;

N – número de ordens em $S(t_0)$.

Antes de iniciar o passo 1, é necessário definir o conjunto $S(t_0)$, assim, somente as ordens contidas neste conjunto é que serão submetidas inicialmente a avaliação da heurística. Este conjunto será alterado a cada novo instante t_0. O instante t_0 será o instante de término C_j da ordem escolhida como próxima a ser processada. Para o inicio da programação será considerado como zero ou o menor r_j disponível caso nenhuma ordem esteja disponível no instante zero.

Para o passo a seguir uma ordem j qualquer será escolhida do conjunto $S(t_0)$ como a próxima a ser processada.

Passo 1: *Esta etapa define qual o melhor valor que pode assumir para a ordem j em estudo.*

$$C_j^* = \max\left\{t_0 + p_j, r_j + p_j, d\right\} \quad (8)$$

Passo 2: *Determine o instante médio de conclusão, das ordens remanescentes no conjunto $S(t_0)$:*

$$\overline{C} = \overline{a} + C_j^* + \frac{1}{2}\left(P - \overline{P}\right) + \overline{P} \quad (9)$$

$$P = \sum_{i \in S(t_0), i \neq j} p_i \quad (10)$$

$$\overline{a} = \frac{1}{N-1} \sum_{i \in S(t_0), i \neq j} \max\{r_i, t_0\} \quad (11)$$

$$\overline{P} = \frac{P}{N-1} \quad (12)$$

Passo 3: *Determine o valor a ser assumido por C_j e, por conseguinte, o melhor instante de início, s_j, para a ordem em estudo:*

Se. $\alpha < ß$ então.

$$C_j = \max\left\{C_j^* - \left(\overline{C} - d\right) r_j + p_j, t_0 + p_j\right\} \quad (13)$$

Caso a condição não seja atendida consideramos então:
$C_j = C_j^*$,
E a melhor data de início será: (14)
$s_j = C_j - p_j$

Passo 4: *Estime, agora, os instantes de início s_i e término C_i das ordens remanescente de $S(t_0)$:*

$$s_i = \max\left\{C_j + 0{,}5(P - p_i), r_i, d - p_i\right\} \quad (15)$$

$$C_i = s_i + p_i \quad (16)$$

$$\forall_{i \in S(t_0), i \neq j}$$

Passo 5: *Calcule para a ordem j em estudo qual seria seu custo total TC(j), como se ela fosse a próxima a ser processada.*

$$TC_{(j)} = \sum_{i \in S(t_0)} \alpha\left[d - C_i\right]^+ + \beta\left[C_i - d\right]^+ \quad (17)$$

Passo 6: *Retorne ao passo 1 utilizando agora uma nova ordem j contida em $S(t_0)$ e refaça todos os passos até o passo 5. Quando todas as ordens tiverem sido investigadas pule para o próximo passo.*

Passo 7: *Selecionar a ordem j com o menor TC calculado. Utilizar s_j e C_j calculados no passo 3. Programe esta ordem como a próxima da seqüência.*

Atualize t_0, volte a definir o conjunto $S(t_0)$ e inicie no passo 1 novamente.

Após a seqüência montada, a mesma é analisada para checar se há algum tempo ocioso entre ordens que se iniciam antes da data de entrega e que poderia ser removido adiantando a ordem ou as ordens anteriores. Como por exemplo, supondo que a primeira ordem da programação começa no instante zero e finaliza no instante 3. A segunda ordem só pode iniciar no instante 5 porque sua data de liberação impede que comece no instante anterior. Então a primeira ordem deverá ser adiantada a começar no instante 2 para que termine no instante 5, removendo o tempo ocioso que tinha entre as ordens e assim por diante.

4. RESULTADOS COMPUTACIONAIS

Os dados para a realização dos testes foram gerados utilizando o programa em Pascal desenvolvido por Biskup e Feldmann (2001). Neste programa os dados são gerados aleatoriamente dentro de uma faixa definida para cada parâmetro. Para manter a mesma idéia dos autores, foram mantidas a semente e a distribuição uniforme de valores de geração, logo os tempos de processamento, são os mesmos gerados por Biskup e Feldmann.

São gerados 70 problemas diferentes divididos em 7 diferentes tamanhos (10, 20, 50, 100, 200, 500 e 1000 ordens) e cada um com 10 diferentes replicações. Para o tempo de processamento a faixa da distribuição uniforme adotada foi [1-20], para penalidade por adiantamento [1-10] e para penalidade por atraso [1-15]. Para datas de liberação da ordem uma adaptação do programa original de Biskup e Feldmann (2001) foi realizada. A faixa de valores definida para a geração das datas de liberação foi baseada nos trabalhos de Akturk e Ozdemir (2001) e Chu (1992), que varia conforme a faixa de distribuição [0-k], e cujo limite superior k é definido por um fator multiplicador α sobre a somatória dos tempos de processamento.

$$k = \left\lceil \alpha \sum p_i \right\rceil, \text{ onde } a = 0{,}5 \qquad (18)$$

Conforme a equação (18), a distribuição uniforme de valores para gerar as datas de liberação é função da somatória dos tempos de processamento de cada replicação do problema. Logo, a faixa de geração da data de liberação será [0, k] para cada um dos problemas como mostra a tabela 1.

	Número de Ordens por Problema						
Réplica	10	20	50	100	200	500	1000
1	58	109	275	568	1065	2609	5306
2	65	119	256	641	1066	2718	5124
3	63	117	268	537	1001	2641	5234
4	51	115	239	523	1075	2742	5141
5	47	94	271	530	1086	2548	5343
6	44	104	274	526	1036	2569	5283
7	52	122	277	521	1030	2672	5283
8	40	101	319	593	1049	2675	5174
9	46	70	229	541	1038	2704	5176
10	64	108	253	532	1064	2638	5287

TABELA 1 – Limite Superior (k) para a geração das datas de liberação

A data de entrega *d* é gerada pela somatória dos tempos de processamento do problema e multiplicado pelo fator *h*.

$$d = \sum_{i=1}^{n} p_i * h \text{, sendo o fator } h = 0{,}2;\ 0{,}4;\ 0{,}6 \text{ e } 0{,}8 \qquad (19)$$

Os testes levaram em consideração 4 diferentes datas de entrega definidas pelo fator *h*, sendo portanto gerados 280 problemas. A heurística proposta foi implementada na linguagem Pascal e os testes realizados em um computador com processador de AMD K6-II de 500 MHz e 64 Mb de memória RAM.

Os problemas com 10 ordens foram também resolvidos de forma exata através do software LINDO (*Linear, Interactive and Discrete Optmizer*), com o intuito de avaliar a performance da heurística proposta. A tabela 2 fornece uma comparação entre os resultados obtidos para os 40 casos testados. Os valores gerados pelo LINDO estão definidos na tabela como FO*.

		Problemas										
h		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*Média*
0,2	*FO*	*1953*	*4638*	*5826*	*4666*	*1216*	*2730*	*4670*	*3568*	*531*	*2812*	
	*FO**	*1953*	*4494*	*5826*	*4666*	*1128*	*2730*	*4670*	*3568*	*521*	*2616*	
FO/FO* (%)		***0,0%***	***3,2%***	***0,0%***	***0,0%***	***7,8%***	***0,0%***	***0,0%***	***0,0%***	***1,9%***	***7,5%***	***2,0%***
0,4	*FO*	*1361*	*2908*	*3618*	*2898*	*1096*	*2226*	*2861*	*2418*	*372*	*1792*	
	*FO**	*1308*	*2908*	*3618*	*2786*	*968*	*1800*	*2861*	*2408*	*306*	*1562*	
FO/FO* (%)		***4,1%***	***0,0%***	***0,0%***	***4,0%***	***13,2%***	***23,6%***	***0,0%***	***0,42%***	***21,6%***	***14,7%***	***8,2%***
0,6	*FO*	*1273*	*2138*	*2634*	*2432*	*1096*	*2086*	*2172*	*2416*	*319*	*1152*	
	*FO**	*1082*	*2079*	*2592*	*2142*	*844*	*1587*	*1968*	*2042*	*290*	*874*	
FO/FO* (%)		***17,7%***	***2,8%***	***1,6%***	***13,5%***	***29,8%***	***31,4%***	***10,4%***	***18,3%***	***10,0%***	***31,8%***	***16,7%***
0,8	*FO*	*1389*	*2307*	*2376*	*2696*	*1096*	*2086*	*2561*	*3004*	*354*	*926*	
	*FO**	*1072*	*1726*	*2130*	*2068*	*844*	*1315*	*1829*	*1684*	*234*	*714*	
FO/FO* (%)		***29,6%***	***33,7%***	***11,5%***	***30,4%***	***29,8%***	***58,6%***	***40,0%***	***78,4%***	***51,3%***	***29,7%***	***39,3%***

TABELA 2 – Comparação da Heurística proposta com o software LINDO

A tabela 2 mostra que a heurística proposta tem uma perda de aderência cada vez maior em relação ao valor ótimo conforme o fator *h* é incrementado, isto é, para os problemas com valor de *h*=0,8 a média das diferenças entre o encontrado pela heurística e o valor ótimo foi de 39,3%, já com *h*=0,6 esta diferença média cai para 16,7% , para *h*=0,4 a média é 8,2% e por último *h*=0,2 a diferença média cai para 2,0%. Em resumo esta heurística conseguiu ter sua melhor performance e até atingir o ótimo (em 9 problemas, ou melhor, 22,5 % dos problemas) quando a data de entrega comum é altamente restritiva, isto é *h*=0,2. Isto indica que a heurística possui uma boa performance quando a maioria das ordens está em atraso e se quer minimizar este atraso. Para os outros casos apesar da média ser maior há algumas exceções que produzem bons resultados, como nos problemas 2, 3 e 7 com *h*=0,4 que atingiram o ótimo, os problemas 2 e 3 com *h*=0,6 e o problema 3 com *h*=0,8 cuja variação são respectivamente 2,8%; 1,6% e 11,6%; e que poderiam ser utilizados em um estudo futuro para melhoria da heurística.

5. CONCLUSÃO

Neste artigo foi apresentada uma heurística construtiva para minimizar a soma das penalidades de atraso e adiantamento das ordens de produção em relação à data de entrega comum e restrita. O método proposto foi submetido a testes computacionais para mostrar sua eficiência em comparação aos resultados ótimos obtidos para pequenos problemas.

Esta heurística, inicialmente concebida para resolver problemas com penalidades e data de entrega diferentes para cada ordem, ou seja, α_i, β_i e d_i e agora adaptada para resolver problemas de data única e penalidades do tipo α e β, tem potencial de solução já que a mesma incorpora uma visão futura das ordens que estão para chegar e analisa se não existe ordem em potencial para ser programada antes das que já estão na fila. Portanto, cabe ressaltar que esta heurística tem espaço para sofrer melhorias a ponto de reduzir, principalmente, a variação que existe nos resultados para datas de entrega cujo fator h está entre 0,6 e 0,8.

Novos testes estão em fase de desenvolvimento a fim de conseguirmos uma melhor performance da heurística proposta.

6. REFERÊNCIAS

ARNOLD, J. R. T. **Administração de Materiais.** São Paulo: Atlas, Cap. 15, 1999.

AKTURK, M. S.; OZDEMIR, D. **A New Dominance Rule to Minimize Total Weighted Tardiness with Unequal Release Dates.** European Journal of Operational Research, Vol 135, p.394-412, 2000.

BAGCHI, U.; SULLIVAN, R.; CHANG Y. (1987), **Minimizing Mean Square Deviation of Completion Times About a Common Due Date.** Management. Science, Vol. 33, p.894-906, 1999.

BANK, J.; WERNER, F. **Heuristic Algorithms for Unrelated Parallel Machine Scheduling with a Common Due Date, Release Dates, and Linear Earliness and Tardiness Penalties.** Mathematical and Computer Modelling, Vol. 33, p.363-383, 2001.

BISKUP, D.; FELDMANN, M. **Benchmarks for Scheduling on a Single Machine Against Restrictive and Unrestrictive Common Due Date.** Computers & Operations Research, Vol. 28, p.787-801, 2001.

CHENG, T. C. E.; CHEN, Z. L.; SHAKHLEVICH N. V. **Common Due Date Assignment and Scheduling with Ready Times.** Computers and Operations Research, Vol. 29, p.1957-1967, 2002.

CHU, C. **A Branch-and-Bound Algorithm to Minimize Total Tardiness with Different Release Dates.** Naval Research Logistics, Vol. 39, p.265-283, 1992.

DAVIS, J. S.; KANET, J. J. **Single-Machine Scheduling with Early and Tardy Completion Costs**. Naval Research Logistics, Vol. 40, p.85-101, 1993.

EMMONS, H. **Scheduling to a Common Due Date on Parallel Common Processors**. Naval Research Logistics Quartely, Vol. 34, p.803-810, 1987.

KANET, J. J. **Minimizing the Average Deviation of Job Completion Time About a Common Due Date**. Naval Research Logistics Quartely, Vol. 28, p.643-651, 1981.

KANET, J. J.; ZHOU, Z. **A Decision Theory Approach to Priority Dispatching for Job Shop Scheduling**. Production and Operations Management, Vol. 2, p.2-14, 1993.

MONDAL, S.; SEN, A. **Single Machine Weighted Earliness-Tardiness Penalty Problem with a Common Due Date.** Computers & Operations Research, Vol. 28, p.649-669, 2001.

NANDKEOLYAR, U.; AHMED, M. U.; SUNDARARAGHAVAN, P. S. **Dynamic Single-Machine-Weighted Absolute Deviation Problem:** Predictive Heuristics and Evaluation. International Journal of Production Research, Vol. 6, p.1453-1466, 1993.

PANWALKAR, S.; SMITH, M.; SEIDMANN, A. **Common Due Date Assigment to Minimize Total Penalty for the One Machine Scheduling Problem.** Operations Research, Vol. 30, p.391-399, 1982.

SIDNEY, J. B. **Single-Machine Scheduling with Earliness and Tardiness Penalties.** Operations Research, Vol.25, p.62-69, 1977.

SRIDHARAN, V.; ZHOU, Z. **A Decision Theory Based Scheduling Procedure for Single-Machine Weighted Earliness and Tardiness Problems.** European Journal of Operational Research, Vol.94. p.292-301, 1996.

VALENTE, J. M. S.; ALVES, R. A. F. S. **Efficient Polynomial Algorithms for Special Cases of Weighted Early/Tardy Scheduling with Release Dates and a Common Due Date**. Pesquisa Operacional, Vol.23, p.443-456, 2003.

WOMACK, J.; JONES, D. **A Máquina que Mudou o Mundo.** São Paulo: Editora Campus-São Paulo, 2 ed.