Verax consultoria

Função: **operações**
Segmento: **indústria / logística**
Tema: **planejamento de operações**
Metodologia: **análise quantitativa**

BEAM SEARCH FILTRADO COM INSERÇÃO DE OCIOSIDADE NA PROGRAMAÇÃO DE UMA MÁQUINA EM AMBIENTE DO TIPO JIT

Detalha um método para programar ordens de produção considerando custos de adiantamento e de atraso quando as ordens são concluídas fora da data exata devida. Idealmente, métodos desse tipo devem ser usados para se programar os recursos mais importantes do sistema produtivo, como por exemplo, a máquina-gargalo.

RESUMO

Este trabalho avalia a utilização do *beam search* filtrado (BSF) combinado com um algoritmo de inserção de ociosidade (AIO). O caso estudado é baseado em uma única máquina, com datas de entrega múltiplas e com penalidades distintas de adiantamento e de atraso para cada ordem. O objetivo a ser alcançado é a minimização do custo total. Para isso, o BSF é utilizado para gerar as seqüências, e o AIO, para definir os programas. Assume-se que a dificuldade de solução do problema é dependente de dois parâmetros: fator de atraso médio e amplitude relativa das datas de entrega. Testes empíricos comparativos são realizados através de simulação computacional, onde se mede o tempo de solução e o valor alcançado pela função-objetivo. Os resultados indicam que de uma forma geral, o procedimento proposto propicia uma diminuição no custo total e, além disso, que para a escolha de um procedimento apropriado, o ideal é se conhecer o valor dos parâmetros.

1 INTRODUÇÃO

Desde o final dos anos 80, vários trabalhos têm apontado para uma linha de pesquisa que utiliza um dos principais pontos da filosofia JIT – evitar que produtos e componentes sejam produzidos ou entregues antes da data correta – como parte da função-objetivo de um sistema que empurra a produção. Em outras palavras, seria a utilização de uma parte da filosofia JIT em um sistema que empurra a produção.

A adaptação dessa filosofia se resume basicamente na utilização da inserção de ociosidade. A inserção de ociosidade é relevante para ambientes onde exista um custo caso o produto seja finalizado em horário anterior à sua data de entrega. Podem-se citar exemplos da utilização dessa filosofia na produção de produtos perecíveis ou com custos de componentes comprados muito altos. Como esse custo é absolutamente eliminado com a inserção de ociosidade não há razão para aceitá-lo. O inconveniente dessa inserção é um possível "arrependimento" futuro no caso de se precisar do tempo que foi gasto como ociosidade.

Este trabalho avalia a utilização do *beam search* filtrado (BSF) juntamente com um algoritmo de inserção de ociosidade (AIO). Como será visto adiante, o BSF vai gerando seqüências de ordens e o AIO vai definindo os programas ótimos para as seqüências definidas pelo BSF. Os resultados computacionais indicam que, para uma

média dos valores alcançados nas replicações de tipos de instâncias, o procedimento proposto suplantou o BSF em todos os casos analisados, indicando que realmente propicia um custo total de programação menor. Há casos onde a inserção de ociosidade chega a diminuir o valor da função-objetivo em mais de 40%. Por outro lado, a inserção de ociosidade fez com que o tempo de solução computacional piorasse, embora ainda dentro de valores aceitáveis.

2 DEFINIÇÃO E REVISÃO BIBLIOGRÁFICA DO PROBLEMA

Definições e terminologias utilizadas neste trabalho acerca da programação e seqüenciação são baseadas principalmente no livro de MORTON & PENTICO (1993). Abaixo será feita uma breve revisão dos principais tópicos utilizados mais adiante.

Neste trabalho haverá uma diferença entre a programação e a seqüenciação. A seqüenciação será definida como uma ordenação dos pedidos – ordens – de produção. Para um conjunto de ordens $J=\{J_1,...,J_n\}$ e para $\sigma(j)$ a j-ésima ordem na seqüência σ, deve-se encontrar a seqüência $\sigma=<\sigma(1),\sigma(2),...,\sigma(n)>$ com $\sigma(j)\in\{1,2,...,n\}$ e $\sigma(j)\neq\sigma(i)$ sempre que $j\neq i$. Observar que uma seqüência representada por <...> não deve ser confundida com um conjunto representado por {...}, pois o mesmo não possui seus elementos ordenados. Cada ordem J_j (j=1,...,n), possui quatro valores associados $\{p_j,d_j,w_j,h_j\}$, onde p_j,d_j,w_j,h_j são o tempo de processamento, a data de entrega, a penalidade de atraso e a penalidade de adiantamento respectivamente.

A programação será definida como a determinação dos horários de início e de término das ordens de uma dada seqüência. Para uma dada ordem J_j (j=1,...,n), existe um intervalo de processamento $[e_j,e_j+p_j]$ onde e_j significa o horário efetivo de início do processamento da ordem J_j. No caso do problema de uma máquina, para duas ordens consecutivas $J_{\sigma(j)}$ e $J_{\sigma(j+1)}$, com intervalos de processamento definidos por $[e_{\sigma(j)},e_{\sigma(j)}+p_{\sigma(j)}]$ e $[e_{\sigma(j+1)},e_{\sigma(j+1)}+p_{\sigma(j+1)}]$, poderá haver intersecção entre os dois intervalos apenas nos pontos extremos dos intervalos. Seja $W_{\sigma(j)}$ o período de ociosidade antes da realização da ordem $J_{\sigma(j)}$. Supondo que as ordens estejam disponibilizadas no instante 0, o programa $\pi(\sigma)$ de um conjunto de ordens J, é a determinação do horário de início e de término de cada ordem, ou seja, $\pi(\sigma)=<W_{\sigma(1)},p_{\sigma(1)},W_{\sigma(2)},p_{\sigma(2)},...,W_{\sigma(n)},p_{\sigma(n)}>$.
Semanticamente pode-se dizer que para uma dada seqüência, um programa define o período de ociosidade antes do início de cada ordem $W_{\sigma(j)}$ e o período de processamento da mesma $p_{\sigma(j)}$, para j=1,2,...,n.

Para casos em que não haja a possibilidade da inserção de ociosidade, isto é, $W_{\sigma(j)}=0\forall j\in\{1,2,...,n\}$, a programação e a seqüenciação são equivalentes pois cada ordem possui apenas um tempo de processamento a ela atribuído.

2.1 DEFINIÇÃO DO PROBLEMA

Para um programa π, pode-se denominar o horário de término da ordem $J_{\sigma(j)}$ a função:

$$C_{\sigma(j)}(\pi)=\begin{cases}W_{\sigma(j)}(\pi)+p_{\sigma(j)} \text{ para } j=1\\ C_{\sigma(j-1)}(\pi)+W_{\sigma(j)}(\pi)+p_{\sigma(j)} \text{ para } j=2,...,n.\end{cases}$$

A maioria das outras funções de programação depende desta função. A função diferença representa a quantidade de tempo em que o término de uma ordem difere de sua data de entrega. Por uma questão de simplicidade notacional, quando não causar erro conceitual, será dada preferência para a notação que omite a seqüência σ. Exemplos desse caso são apresentados abaixo, onde os valores das funções para $J_{\sigma(j)}$ são equivalentes a valores para J_j. Define-se a função diferença como $L_j=C_j-d_j$. Costuma-se separar a função diferença quando ela é positiva e quando ela é negativa. Se ela é negativa, é chamada de adiantamento e é definida por $E_j=\max\{0,-L_j\}$, enquanto que se ela é positiva, é chamada de atraso e pode ser definida por $T_j=\max\{0,L_j\}$.

Será considerado o caso de uma única máquina que deverá processar uma ordem de cada vez de um conjunto de ordens $J=\{J_1,J_2,...,J_n\}$. A ordem $J_{\sigma(j)}$ deve ser finalizada antes que a ordem $J_{\sigma(j+1)}$ seja iniciada para $\sigma(j)$=1,2,...,n–1. As chegadas das ordens serão estáticas, ou seja, todas estarão disponíveis no instante inicial. O programa deverá ser realizado de maneira a cumprir as datas de entrega d_j, sabendo-se que cada ordem demanda um determinado tempo p_j para o seu processamento. Há possibilidade de inserção de tempo ocioso W_j antes da elaboração de uma determinada ordem J_j. A função-objetivo a ser analisada será do tipo min $\sum_{j=1}^{n}(h_jE_j+w_jT_j)$, sendo as penalidades h_j>0 e w_j>0.

2.2 PRINCIPAIS RESULTADOS CONHECIDOS

A primeira publicação que se tem notícia acerca do problema especificado acima é devida à FRY *et al.* (1987). Embora os autores façam todo o desenvolvimento

teórico com penalidades para cada ordem, h_j e w_j, os testes computacionais são apresentados com um único valor para todas as ordens, h e w. Não há comentários sobre a razão dessa simplificação. Com o procedimento proposto encontrou-se ótimos globais em diversas instâncias testadas e o comportamento médio em todo estudo foi menor do que 2% acima do ótimo global. A proposta de FRY *et al.* é uma heurística do tipo troca de pares adjacentes, com a possibilidade da inserção de ociosidade através da utilização da programação linear. Pelas características do problema, o procedimento de programação linear pôde ser resolvido em tempo polinomial, através de um procedimento de um passo.

ABDUL-RAZAQ & POTTS (1988) analisaram o problema sem a possibilidade da inserção de ociosidade, utilizando a programação dinâmica e o *branch-and-bound.* Inicialmente os autores apresentaram 3 maneiras de se definir limitantes inferiores através da utilização de uma relaxação de estado-espaço na programação dinâmica. Em seguida, formularam um procedimento ótimo utilizando o *branch-and-bound* com seqüenciação para frente, com um limitante superior sendo definido por uma heurística de um passo e com os limitantes inferiores conforme uma das 3 propostas. Não houve nenhuma indicação forte de que um dos métodos fosse superior aos outros em todos os sentidos. Os autores concluem que a programação dinâmica deve ser utilizada na resolução de problemas até 10 ordens, enquanto que para problemas de até 25 ordens, deve-se utilizar o *branch-and-bound* com os limitantes inferiores sendo definidos pela programação dinâmica.

Os trabalhos de OW & MORTON (1988 e 1989) apresentam comparações entre 4 procedimentos heurísticos que combinam regras de despacho e funções de prioridade com *beam search* (BS). O problema analisado foi equivalente ao tratado aqui, só que sem a possibilidade de inserção de ociosidade. O melhor dos quatro procedimentos foi o BSF que será explicado detalhadamente adiante. O desempenho de todos esses procedimentos foi medido através do custo obtido pela heurística em teste e pela melhor solução conhecida – para instâncias pequenas *branch-and-bound* e para instâncias maiores, limitante inferior, utilizando uma relaxação envolvendo a divisão da ordem em subordens e estas sendo resolvidas como um problema de programação linear. Para as instâncias de 15 e de 25 ordens, os procedimentos ficaram entre 1 e 7% acima do limitante inferior.

Posteriormente, YANO & KIM (1991) forneceram outro procedimento de inserção de ociosidade. Nesse procedimento, os autores utilizaram uma formulação com equações recursivas, assumindo que $w_j \geq h_j \geq 0$. Os autores utilizaram o *branch-and-bound*, já considerando a inserção de ociosidade no limitante inferior para verificar o desempenho de quatro regras de despacho e uma heurística de troca de pares adjacentes. Esse último procedimento é parecido com o utilizado por FRY *et al*. e utiliza como seqüência inicial àquela definida pela regra de despacho com melhor desempenho dentre as quatro testadas. Nos cinco casos, após a definição da seqüência final, utiliza-se o procedimento para a inserção de ociosidade e elaboração do programa. Os resultados indicaram que em média, a heurística ficou apenas a 0,1% do ótimo. Os autores finalizam o trabalho apontando que haveria uma necessidade de se criar limitantes inferiores melhores do que os que foram utilizados no estudo.

3 BEAM SEARCH (BS)

De uma maneira direta e simplificada, pode-se dizer que o BS é uma técnica de busca que utiliza certo número de soluções em paralelo em uma árvore de busca. O número de soluções em paralelo é chamado de largura da busca.

Para o desenvolvimento do trabalho serão utilizadas três operações com seqüências, definidas como segue:

Definição 1: se $\sigma=<\sigma(1),...,\sigma(n)>$ é uma seqüência com n elementos, define-se a operação de transformação da seqüência σ, no conjunto J, denotada por $conj(\sigma)=J$, como aquela que forma um conjunto – sem ordenação – com todos os elementos pertencentes à seqüência, ou seja, $J=\{\sigma(1),...,\sigma(n)\}$ e $|J|=n$.

Exemplo: para $\sigma=<A,B,C>$, $conj(\sigma)=\{A,B,C\}=\{B,A,C\}=\ldots=\{C,B,A\}$.

Definição 2: se $s_1=<s_1(1),...,s_1(j)>$ e $s_2=<s_2(1),...,s_2(i)>$ são duas seqüências parciais (isto é, não são formadas necessariamente por todos os elementos disponíveis), define-se a concatenação de s_1 e s_2 como a seqüência $<s_1(1),...,s_1(j),s_2(1),...,s_2(i)>$. Uma operação de concatenação será denotada por s_1+s_2.

Exemplo: para $s_1=<A,B,C>$ e $s_2=<D,E,F>$, $s_1+s_2=<A,B,C,D,E,F>$.

Definição 3: se $\sigma=<\sigma(1),...,\sigma(n)>$ é uma seqüência completa e $s_1=<s_1(1),...,s_1(j)>$ é uma seqüência parcial – onde todo elemento de $conj(s_1)$ está em $conj(\sigma)$ –, define-se a diferença entre σ e s_1 como a seqüência que contém os elementos ordenados com as mesmas relações de precedência que em σ, mas não contém os elementos de s_1. Nesse caso a operação diferença será denotada por $\sigma-s_1=s_2=<s_2(1),...,s_2(n-j)>$.

Exemplo: para σ=<A,B,C,D,E,F> e s_1=<B,D,E>, σ-s_1=<A,C,F>.

As seqüências utilizadas adiante serão formadas pelos índices dos elementos de *J*. Seja ν ($\nu \in \mathbf{N}$: $0 \le \nu \le n$) o nível da árvore de busca; *b* a largura da busca, isto é, o número de seqüências completas retidas para arborescência no nível subseqüente; *r* ($r \in \mathbf{N}$:$1 \le r \le b(n-\nu+1)$) um nó em um determinado nível ν. Seja σ_r^ν uma seqüência completa formada no nó *r* do nível ν e σ^0 uma seqüência inicial completa. Cada seqüência σ_r^ν será formada por uma seqüência parcial fixa s_r^ν (onde $|\text{conj}(s_r^\nu)| = \nu$) e uma seqüência parcial complementar $\bar{s}_r^\nu = \sigma^0 - s_r^\nu$.

Para o nó *r* do nível ν, uma seqüência completa será formada da seguinte forma:

$$\sigma_r^\nu = s_r^\nu + \bar{s}_r^\nu =< s_r(1), s_r(2), \ldots, s_r(\nu) > + < \sigma^0 - s_r(1), s_r(2), \ldots, s_r(\nu) >$$

Deseja-se encontrar uma seqüência completa

$$\sigma_r^n = s_r^n + \bar{s}_r^n =< s_r(1), s_r(2), \ldots, s_r(n) > + \varnothing,$$

onde s_r^n representa uma seqüência parcial fixa contendo todos os elementos de *J*.

Seja R^0=conj(σ^0)={1,...,*n*} o conjunto de índices das ordens disponíveis pertencentes à seqüência inicial. Seja *q* ($q \in \mathbf{N}$:$1 \le q \le b$) um dos nós retidos para arborescência no nível posterior; $R_q^\nu = R^0 \setminus \text{conj}(s_q^{\nu-1})$ o conjunto de índices de ordens disponíveis para serem processadas nos nós descendentes de *q* do nível ν–1; $A^\nu = \{\sigma_r^\nu\}$ ($r \in \mathbf{N}$:$1 \le r \le b(n-\nu+1)$) o conjunto de todas as seqüências completas criadas no nível ν; $B^\nu =< B_q^\nu >$ ($q \in \mathbf{N}$:$1 \le q \le b$) a seqüência de seqüências – retidas – que contém os *b* menores valores da função $g(\sigma_r^\nu) : \sigma_r^\nu \in A^\nu$; B_q^ν a seqüência de *q*-ésimo menor custo. Para ($u \in \mathbf{N}$:$1 \le u \le b$), pode-se dizer que

$$B_q^\nu = \begin{cases} \sigma_u^\nu : \min\limits_{\sigma_r^\nu \in A^\nu} g(\sigma_r^\nu) = g(\sigma_u^\nu) \text{ se } q = 1 \\ \sigma_u^\nu : \min\limits_{\sigma_r^\nu \in A^\nu \setminus \text{conj}< B_{q-1}^\nu, B_{q-2}^\nu, \ldots, B_1^\nu >} g(\sigma_r^\nu) = g(\sigma_u^\nu) \text{ se } q \ge 2, \end{cases}$$

ou, de uma forma mais compacta,

$$B_q^\nu = \begin{cases} \arg\min\limits_{\sigma_r^\nu \in A^\nu} g(\sigma_r^\nu) \text{ se } q = 1 \\ \arg\min\limits_{\sigma_r^\nu \in A^\nu \setminus \text{conj}< B_{q-1}^\nu, B_{q-2}^\nu, \ldots, B_1^\nu >} g(\sigma_r^\nu) \text{ se } q \ge 2. \end{cases}$$

O procedimento tem início com a definição de um limitante superior, cuja seqüência é representada por $\sigma^0 =< \sigma^0(1), \sigma^0(2), \ldots, \sigma^0(n) >$, onde $\sigma^0(j)$ significa o índice da ordem de *j*-ésima posição na seqüência σ^0.

Posteriormente gera-se o primeiro nível da árvore de busca com *n* seqüências completas

$$\sigma_j^1 = s_j^1 + \bar{s}_j^1 =< j > + < \sigma^0 - j > \ (j=1,2,\ldots,n).$$

O conjunto de todas essas seqüências formará o conjunto $A^1 = \{\sigma_j^1\}$ (j=1,2,...,n), onde para cada uma das seqüências, deve-se analisar o valor de sua função objetivo $g(\sigma_j^1)$. Os *b* menores valores formarão a seqüência $B^1 =< B_q^1 >$ ($q \in \mathbf{N}$:$1 \le q \le b$).

No segundo nível, para cada seqüência parcial fixa retida no nó anterior $s_q^1 \in B^1$, deve-se formar outras (n–1) seqüências parciais fixas. Para cada um dos *b* nós do primeiro nível, as seqüências parciais fixas serão definidas pelo primeiro termo do primeiro nível, uma enumeração completa das possíveis ordens no segundo nível e o restante da seqüência derivado de σ^0. Portanto, cada índice de ordem $j \in R_q^2$ (onde $R_q^2 = R^0 \setminus \text{conj}(s_q^1)$) pertencente ao segundo nível, formará seqüências completas do tipo

$$\sigma_j^2 = s_j^2 + \bar{s}_j^2 =< s_q^1, j > + < \sigma^0 - s_q^1, j >.$$

O conjunto de todas essas seqüências formará o conjunto $A^2 = \{\sigma_r^2\} (r \in R_q^2;\ 1 \le q \le b; 1 \le r \le b(n-1))$, onde para cada uma das seqüências, deve-se analisar o valor de sua função objetivo $g(\sigma_r^2)$. Os *b* menores valores formarão a seqüência $B^2 =< B_q^2 >$ ($q \in \mathbf{N}$:$1 \le q \le b$).

Neste ponto a busca é começada no terceiro nível, e o mesmo procedimento é utilizado até o último nível da busca. No último nível, a seqüência escolhida como sendo a melhor, será a que é definida como $\sigma_r^n = B_1^n$.

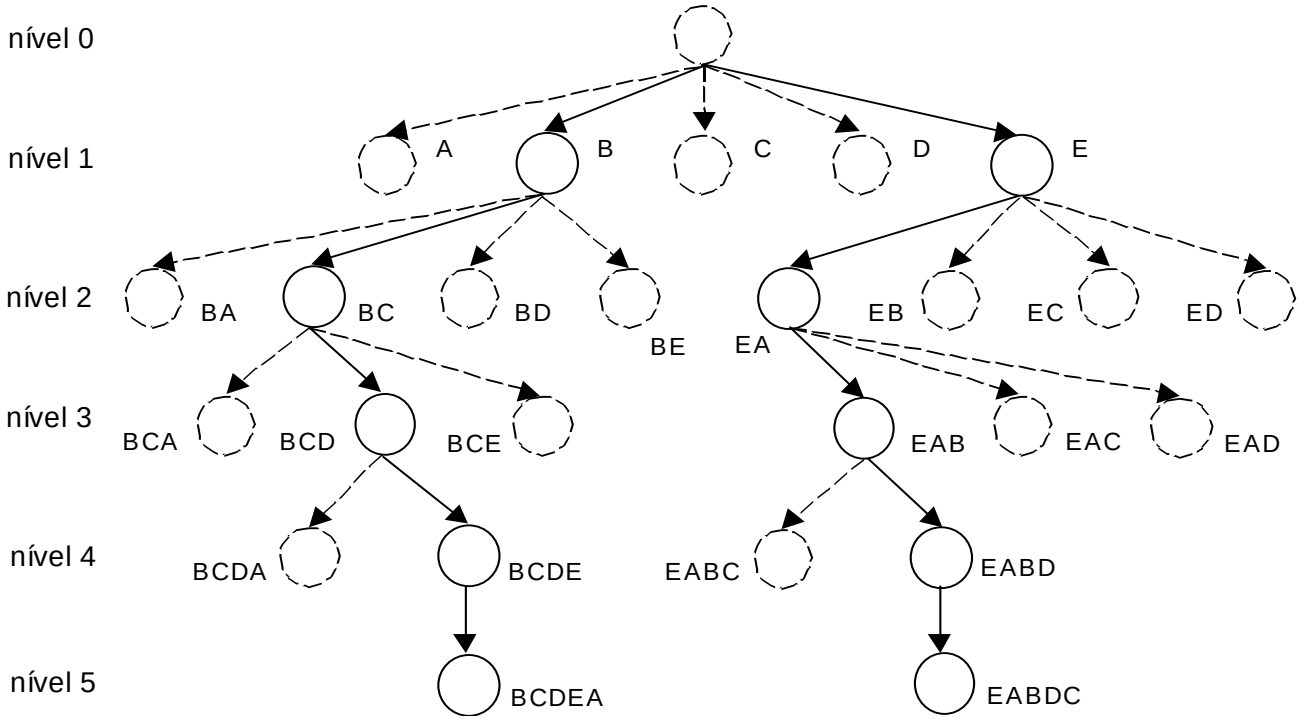

figura 1: árvore de busca utilizando o beam search

Como exemplo, pode-se citar o caso em que haja 5 ordens para serem seqüenciadas J_j (j=D,B,A,C,E). A largura de busca desejada é b=2. Um esquema gráfico do desenvolvimento da solução é mostrado na figura 1. Inicia-se o procedimento encontrando-se uma seqüência inicial de acordo com algum procedimento como uma regra de despacho por exemplo.

Utilizando a regra de despacho, a seqüência inicial é definida por σ^0= <A,B,C,D,E>. A seqüência inicial será considerada o limitante superior. Gera-se o primeiro nível da árvore com uma enumeração completa das ordens que possam estar na primeira posição da seqüência, mantendo os outros elementos de σ^0 e retirando-se o elemento que foi colocado na primeira posição da seqüência de σ^0, ou seja, σ^1_A = <A>+<σ^0–A>= <A,B,C,D,E>, σ^1_B =<B,A,C,D,E>,..., σ^1_E =<E,A,B,C,D>. Comparam-se os valores das funções-objetivo das seqüências obtidas $g(\sigma^1_r)$, e escolhe-se as b=2 melhores. Como resultado $A^1 = \{\sigma^1_A, \sigma^1_B, ..., \sigma^1_E\}$, $B^1_1 = \sigma^1_B$ pois $\min_{\sigma^1_r \in A^1} g(\sigma^1_r) = g(\sigma^1_B)$ e $B^1_2 = \sigma^1_E$ pois $\min_{\sigma^1_r \in A^1 \setminus B^1_1} g(\sigma^1_r) = g(\sigma^1_E)$.

Os nós criados no segundo nível serão apenas os filhos dos nós mantidos no primeiro nível. Para ν um certo nível da árvore de busca – diferente dos níveis 0 e l –, o número de nós criados é equivalente a $b(n-\nu+1)$. Portanto, no caso em análise, cada nó mantido no primeiro nível irá gerar outros quatro nós no segundo nível, totalizando 8 nós. Cada um dos nós será representado pelo primeiro termo do primeiro nível (B ou E), uma enumeração completa das possíveis ordens no segundo nível e o restante da seqüência derivado de σ^0. Para o nó B, σ^2_A =<B,A>+<σ^0–B,A> =<B,A,C,D,E>, σ^2_C =<B,C,A,D,E>, σ^2_D =<B,D,A,C,E>, σ^2_E =<B,E,A,C,D>. O mesmo procedimento é feito para o nó E, com o primeiro termo sendo σ^2_A =<E,A>+<σ^0–E,A> =<E,A,B,C,D>. O restante do procedimento é apresentado na figura 1. No segundo nível as melhores seqüências parciais são σ^2_C e σ^2_A, no terceiro nível σ^3_D e σ^3_B e no quarto nível σ^4_E e σ^4_D. No quinto e último nível, escolhe-se a seqüência de menor custo entre σ^5_A e σ^5_C.

4 BEAM SEARCH FILTRADO (BSF)

O filtro aplicado no BS teve sua origem nos trabalhos apresentados por OW & MORTON (1988 e 1989). O filtro pode ser definido como uma função de avaliação prévia para o BS, transformando o BS em um método de busca de dois estágios. Conforme os estudos de OW & MORTON, pode-se inferir que o BS consegue melhores resultados com a utilização de um filtro adequado.

Na programação e seqüenciação, o filtro pode ser representado por uma função de prioridade da ordem J_j no horário t, $\varphi_j(t)$. Dependendo da função-objetivo que se esteja trabalhando, essa função de prioridade também pode ser uma regra de despacho. Neste trabalho, será utilizada a função de prioridade denominada EXP-ET (*Exponential - Earliness Tardiness*) presente nos estudos de OW & MORTON e explicada detalhadamente no item 5.

Pode-se dizer que o procedimento é semelhante ao BS, com a diferença sendo a forma de escolha das seqüências $\sigma_j^\nu \in A^\nu$, a serem analisadas. Essa alteração de uma avaliação global de cada seqüência para uma avaliação local faz com que o método fique mais eficiente se o número de ordens é grande.

Deve-se formar um conjunto $F=\{F_{11},F_{21},...,F_{f1},F_{12},...,F_{f2},...,F_{1b},...,F_{fb}\}$ onde

$$F_{dq} = \arg \max_{j \in R_q^\nu \setminus \{F_{1q},F_{2q},...,F_{(d-1)q}\}} \varphi_j(t),$$

com $(d,q \in \mathbf{N}: 1 \le d \le f; 1 \le q \le b)$, $F_{0q}=\varnothing$ e lembrando que nesse caso o argumento é um $j \in R_q^\nu$ e não t. Para cada ordem $F_{dq} \in F$, deve-se elaborar e avaliar todas as seqüências completas $\sigma_{F_{dq}}^\nu$, mantendo-se as

$$\sigma_u^\nu = \arg \min_{F_{dq} \in F \setminus \{u-1,u-2,...,1\}} g(\sigma_{F_{dq}}^\nu),$$

para $(u \in \mathbf{N}: 1 \le u \le b)$, com $\sigma_0^\nu = \varnothing$.

As seqüências que não foram escolhidas para serem mantidas são eliminadas definitivamente. Portanto $A^\nu = \{\sigma_{F_{dq}}^\nu : F_{dq} \in F\}$. O restante do procedimento é equivalente ao BS.

Para uma maior clareza de apresentação, o procedimento BSF está disponível no apêndice como um pseudocódigo.

5 FUNÇÃO DE PRIORIDADE EXP-ET

Uma função de prioridade é um procedimento que utiliza informações das ordens para definir qual deve ser processada subseqüentemente e se a mesma deve ter ociosidade inserida, antes de iniciar o seu processamento. Especificamente, a função de prioridade EXP-ET ($\varphi_j(t)$) pode ser definida como sendo:

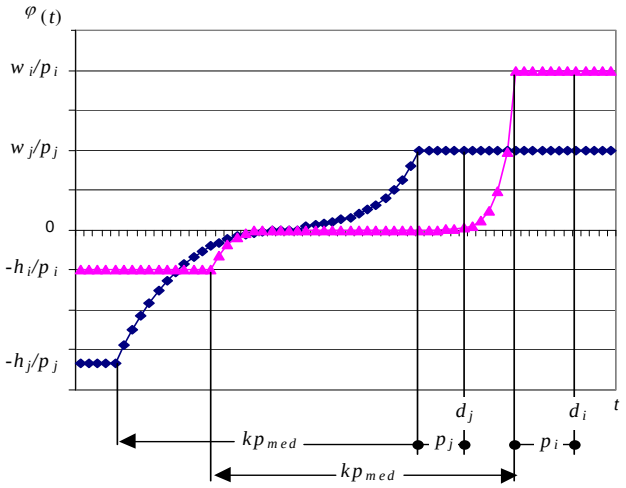

figura 2: significado gráfico da função de prioridade EXP-ET (k=5)

$$\varphi_j(t) = \begin{cases} \frac{w_j}{p_j} \exp\left(-\frac{(h_j+w_j)}{h_j}\frac{\max(0,d_j-t-p_j)}{p_{med}}\right) & \text{se } d_j - t - p_j \le \frac{w_j k p_{med}}{h_j+w_j} \\ \frac{h_j}{p_j}\left(\frac{w_j}{h_j} - \frac{(h_j+w_j)}{h_j}\frac{\min(kp_{med},d_j-t-p_j)}{kp_{med}}\right)^3 & \text{para outros casos,} \end{cases}$$

onde k é um parâmetro de ajuste que determina o horário em que a função de prioridade começa crescer e p_{med} é o tempo de processamento médio das ordens que ainda não foram seqüenciadas, $p_{med} = |D|^{-1} \sum_{j \in D} p_j$, sendo D o conjunto de ordens ainda não seqüenciadas.

A utilização da função pode ser no modo despacho, de maneira que, a ordem que apresentar o maior valor numérico dentre todas as disponíveis deve ser seqüenciada subseqüentemente. Neste trabalho a função será utilizada no modo despacho.

A figura 2 mostra, para 2 ordens, a função de prioridade variando no decorrer do tempo. Perceba que uma ordem J_j que no horário t era menos prioritária que J_i, pode se tornar mais prioritária do que J_i no horário $t+\Delta t$, onde Δt é um intervalo de tempo.

6 INSERÇÃO DE OCIOSIDADE

Problemas que levam em consideração o adiantamento, devem necessariamente utilizar a inserção de ociosidade. Caso não seja assim, uma das principais características desses problemas fica deixada de fora. Mesmo assim, vários autores não trabalharam com a inserção de ociosidade (OW & MORTON, 1988 e 1989, ABDUL-RAZAQ & POTTS, 1988), provavelmente pelo fato de não se conhecer um algoritmo para tal. Permitindo a inserção de ociosidade como uma variável contínua, um conjunto com n ordens possui infinitos programas viáveis.

O primeiro trabalho a apresentar um AIO pertence a FRY *et al.* (1987). De uma maneira bastante clara nesse trabalho, os autores formularam o problema de inserção de ociosidade no problema de atraso e adiantamento com penalidades individuais, como um problema de programação linear. Pela característica do problema, o mesmo pôde ser resolvido em tempo $O(n^2)$.

De uma maneira provavelmente independente, GAREY *et al.* (1988) também propuseram um AIO no problema de atraso e adiantamento sem penalidades, com tempo $O(n \log n)$. Ambos os trabalhos supõem que os algoritmos sejam utilizados em uma seqüência definida preliminarmente.

Algum tempo depois, YANO & KIM (1991) propuseram outro AIO baseado na programação dinâmica. A inserção de ociosidade também é feita após a seqüenciação e o tempo de solução no pior caso é $O(n^2 \log n)$ demonstrando que a implementação desse algoritmo proporciona uma inserção de ociosidade com tempo menos eficiente do que os dois casos comentados anteriormente.

Neste trabalho será utilizada uma versão adaptada do AIO de GAREY *et al.* (1988), disponível em COLIN (1997) e COLIN & SHIMIZU (1998), para a utilização em funções-objetivo do tipo min$\Sigma(h_jE_j+w_jT_j)$. De uma forma concisa, pode-se dizer que após uma seqüência completa ser gerada, o AIO insere a quantidade ótima de ociosidade na seqüência definida preliminarmente.

7 ESTUDO COMPUTACIONAL

O principal objetivo do estudo computacional é analisar o desempenho do BSF quando o mesmo é utilizado juntamente com o AIO. Comparações tentarão identificar casos onde a inserção de ociosidade poderia ser vantajosa levando em consideração características intrínsecas dos dados que estão sendo utilizados. Com a finalidade de se ampliar os estudos originais de OW & MORTON (1988 e 1989), as instâncias testadas serão de grandes dimensões. Devido a esse fator, torna-se inviável a utilização de um método ótimo para comparação.

Mesmo assim, uma hipótese a ser considerada é que a diferença entre a solução ótima e a solução gerada pelo BSF não se altera conforme o aumento do tamanho da instância. Assumindo que essa hipótese seja verdadeira – embora não haja uma garantia efetiva de que isso ocorra –, pode-se considerar as diferenças obtidas nos estudos de OW & MORTON equivalentes às diferenças obtidas neste estudo.

7.1 VARIÁVEIS CONTROLADAS

Desde o começo da década de 70, foram identificadas duas variáveis que parecem ter uma maior influência nos estudos que levam em consideração o atraso – considerando que o adiantamento é um atraso negativo. A primeira delas é o fator de atraso médio, τ (SRINIVASAN, 1971), que pode ser definida semanticamente como a proporção de ordens que atrasariam caso a seqüência fosse obtida aleatoriamente. Matematicamente é definida como

$$\tau = 1 - \frac{d_{med}}{np_{med}} = 1 - \frac{\sum d_j}{n\sum p_j}.$$

A segunda é a amplitude relativa das datas, R (WILKERSON & IRWIN, 1971, BAKER & MARTIN, 1974). Essa variável indica o quanto as datas de entrega estão distribuídas no intervalo total de processamento das ordens. Matematicamente

$$R = \frac{d_{\max} - d_{\min}}{np_{med}} = \frac{d_{\max} - d_{\min}}{\sum p_j}.$$

Para uma maior clareza desses conceitos, a figura 3 apresenta o significado gráfico dos mesmos. Note pela figura que as variáveis alteram a média das datas de entrega e o quanto essas datas estão distribuídas com relação ao tempo total de processamento.

7.2 EXPERIMENTO REALIZADO

O BSF e o AIO foram programados em PASCAL 6.0 e implementados em um microcomputador com processador PENTIUM-133Mhz e 8MB de memória RAM.

As instâncias foram geradas com as seguintes características:

- Número de ordens: n=100;
- Fator de atraso médio: $\tau \in \{0,1;0,2;0,4;0,6;0,8;0,9\}$;
- Amplitude relativa das datas de entrega: $R \in \{0,1;0,2;0,4;0,6;0,8;0,9\}$;
- Tempo de processamento: $p_j \sim U[1,100]$, onde $\sim U[\cdot]$ representa uma distribuição uniforme discreta;

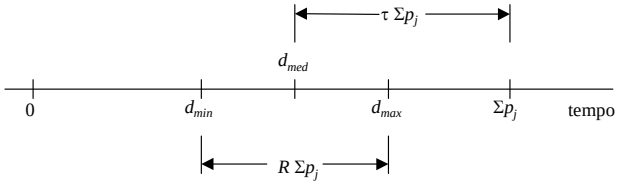

figura 3: variáveis testadas no estudo computacional

- Data de entrega: $d_j \sim U\left[P(1-\tau)-\frac{PR}{2}, P(1-\tau)+\frac{PR}{2}\right]$, onde $P=n\mathrm{E}(p)=(n^2+n)/2$ define a soma da esperança dos tempos de processamento, de acordo com a média da distribuição;
- Penalidade de adiantamento: $h_j \sim U[1,100]$;
- Penalidade de atraso: $w_j \sim U[1,100]$.

Combinando as instâncias com relação a τ e a R, obtem-se 36 tipos de instâncias. Para cada um desses 36 tipos foram geradas 20 instâncias.

Pelo fato de existirem problemas na geração de números aleatórios, uma instância só foi aceita de acordo com alguns critérios de aprovação. O critério é a proporção da diferença entre as variáveis reais e nominais, ou seja:

$$\mathrm{Prop}(\tau) = \frac{|\tau - \tau_{real}|}{\tau} \text{ e}$$

$$\mathrm{Prop}(R) = \frac{|R - R_{real}|}{R}.$$

As definições de τ_{real} e R_{real} para uma determinada instância inst (onde |inst| representa o número de ordens da instância), são as seguintes:

$$\tau_{real} = 1 - \frac{\sum_{j\in \mathrm{inst}} d_j}{|\mathrm{inst}| \sum_{j\in \mathrm{inst}} p_j} \text{ e}$$

$$R_{real} = \frac{\max_{i,k\in \mathrm{inst}} |d_i - d_k|}{\sum_{j\in \mathrm{inst}} p_j}.$$

Com a utilização do critério de aceitação não houve necessidade de se gerar as datas de entrega dependentes dos tempos de processamento, conforme normalmente ocorre na literatura. Note que os casos tratados na literatura podem ter problemas na geração dos tempos de processamento. Não há verificação se os mesmos foram gerados conforme o esperado. No caso proposto aqui, ambos, as datas de entrega e os tempos de processamento, têm seus valores avaliados após a geração.

Para uma dada proporção de desvio dos valores nominais, o programa aceita ou rejeita a instância em questão. Neste trabalho, foi utilizada uma proporção de 15%, tanto em τ como em R, para a aceitação das instâncias.

Para os pares $(\tau,R) \in \{(0{,}8;0{,}8),(0{,}8;0{,}9),(0{,}9;0{,}4),(0{,}9;0{,}6),(0{,}9;0{,}8),(0{,}9;0{,}9)\}$, foi permitida a geração de datas de entrega negativas. Os demais pares foram gerados de acordo com datas positivas. A razão para esse fato fica clara pela observação da figura 3. Se o τ e o R são grandes, a parte esquerda de R – de d_{min} até d_{med} – possui uma região que fica antes de 0.

As soluções das instâncias foram separadas em 2 grandes grupos. O primeiro, denominado grupo 1, é o das soluções no modo despacho, isto é, sem inserção de ociosidade. O grupo 2 é no modo programação, isto é, com a utilização do AIO e com a seqüência inicial sendo definida por EXP-ET no modo despacho.

No modo despacho, programa-se as ordens de acordo com a seqüência, sem a inserção de ociosidade. Esse modo é muito mais facilmente implementável além de ser amplamente utilizado em indústrias. No modo programação, uma seqüência é definida, insere-se a ociosidade nessa seqüência definida preliminarmente para só então elaborar-se o programa com os respectivos horários de início e de término de cada ordem.

7.3 INTERPRETAÇÃO DOS RESULTADOS

Apresentam-se abaixo as comparações entre os diversos casos com relação aos tempos de processamento computacional bem como aos valores das funções-objetivo.

COMPARAÇÃO COM RELAÇÃO AOS TEMPOS DE PROCESSAMENTO COMPUTACIONAL: Os resultados apresentados aqui serão os tempos de processamento computacional médios das 20 instâncias resolvidas com relação a um determinado par (τ,R).

A figura 4 apresenta os tempos de processamento computacional em centésimos de segundo (cs) nos experimentos do grupo 1. Nesse caso não houve variação dos tempos com relação aos parâmetros (τ,R).

No caso dos procedimentos com inserção de ociosidade, o BSF depende dos parâmetros (τ,R). A figura 5 apresenta os tempos de processamento computacional para as larguras de busca 1 e 3, ao passo que a figura 6 apresenta para as larguras 5 e 8.

Pelos gráficos, observa-se que enquanto o produto *bf* é pequeno com relação à escala de tempos, o tempo de solução é relativamente invariante com (τ,R). De uma maneira geral, o tempo diminui com o aumento de τ e de R. Portanto, os menores tempos são obtidos com os

maiores valores do par (τ,R). Isso acontece porque nesse caso, a inserção de ociosidade é muito pequena – quando existe – e portanto o AIO é pouco utilizado.

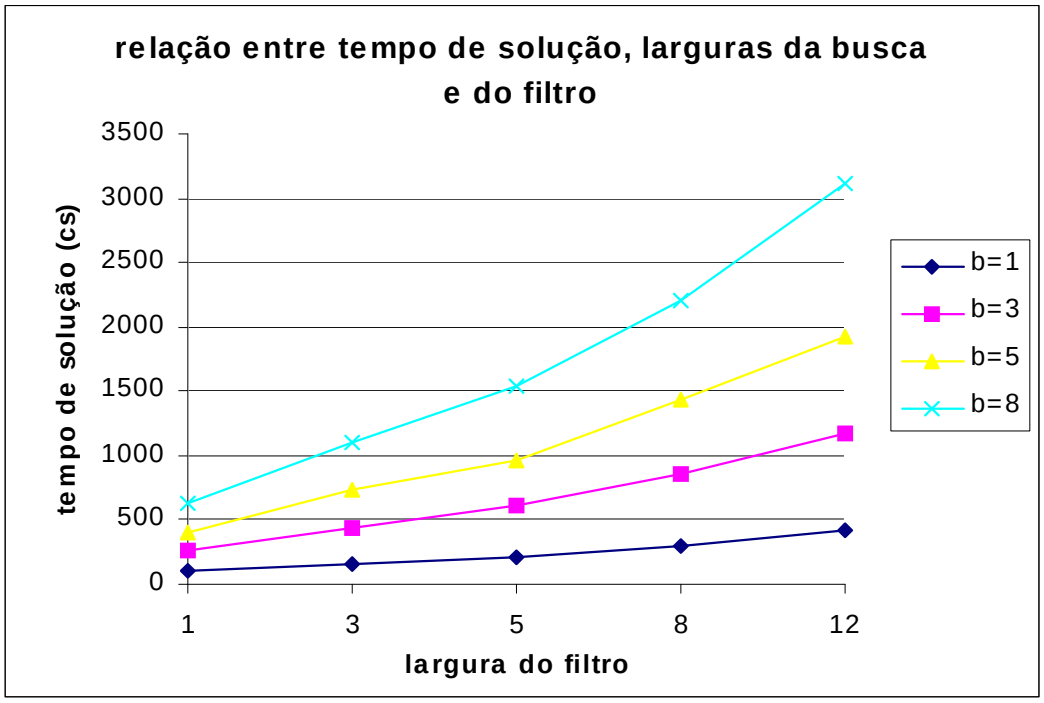

figura 4: relação entre tempo de solução, larguras da busca e do filtro no BSF sem inserção de ociosidade

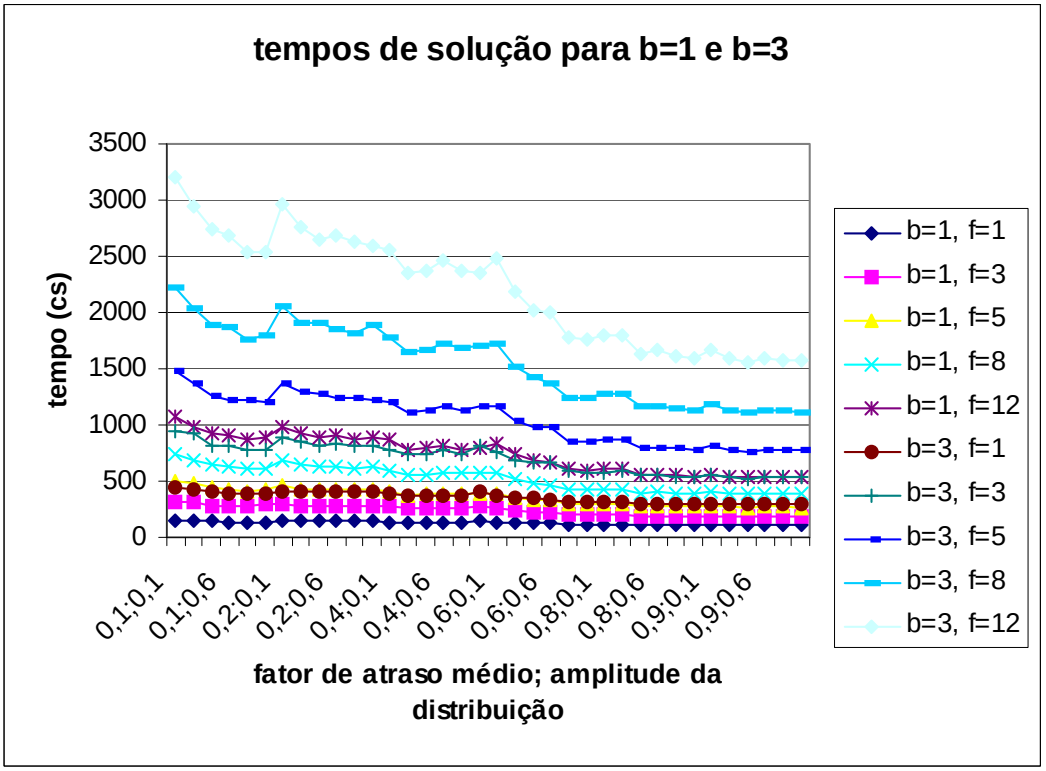

figura 5: tempos de solução do BSF com inserção de ociosidade para *b*=1 e *b*=3

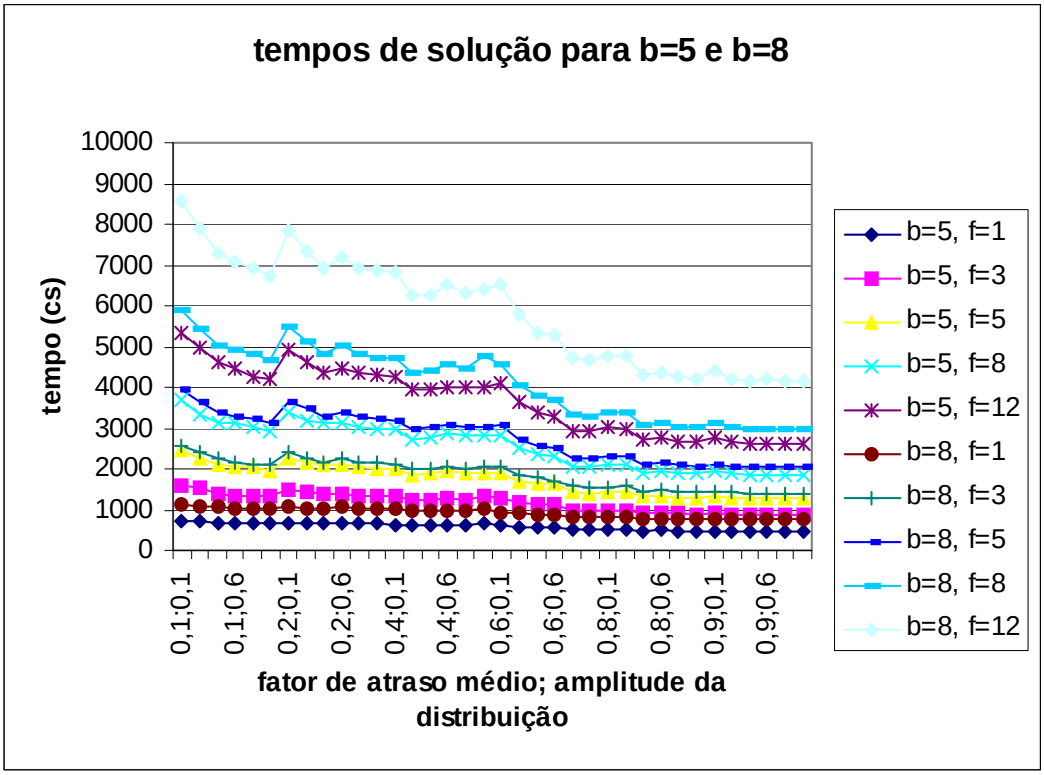

figura 6: tempos de solução do BSF com inserção de ociosidade para *b*=5 e *b*=8

tabela 1: resultados obtidos com relação às funções-objetivo

par	resultado grupo 1			resultado grupo 2			diminuição do custo
	g 1*	(*b* ;*f*) de *g* 1*	DP	*g* 2*	(*b* ;*f*) de *g* 2*	DP	1-*g* 2*/*g* 1*
0,1;0,1	4.562.972	5;5	3.083	4.560.666	8;3	3.662	0,1%
0,1;0,2	4.585.530	5;3	7.465	4.508.650	8;8	14.091	1,7%
0,1;0,4	4.747.974	5;3	12.029	4.163.312	8;1	28.910	12,3%
0,1;0,6	4.805.645	5;3	15.304	3.688.933	8;1	62.662	23,2%
0,1;0,8	5.119.174	5;3	16.743	3.374.460	8;1	65.530	34,1%
0,1;0,9	5.153.336	5;3	22.927	3.041.117	8;1	90.499	41,0%
0,2;0,1	3.422.170	5;3	2.478	3.422.170	8;3	2.478	0,0%
0,2;0,2	3.119.043	5;5	7.748	3.118.984	5;5	7.761	0,0%
0,2;0,4	3.050.337	5;3	12.926	2.981.846	5;5	18.429	2,2%
0,2;0,6	2.726.878	8;3	22.030	2.391.047	8;8	14.249	12,3%
0,2;0,8	3.064.066	8;3	26.677	2.106.824	8;1	66.343	31,2%
0,2;0,9	3.444.806	8;3	28.263	2.043.087	8;1	71.760	40,7%
0,4;0,1	2.400.427	8;5	3.555	2.400.427	8;5	3.555	0,0%
0,4;0,2	2.106.551	5;5	5.739	2.106.551	3;5	5.739	0,0%
0,4;0,4	1.553.698	8;5	8.224	1.553.698	8;5	8.224	0,0%
0,4;0,6	988.660	8;5	14.193	988.340	5;5	14.021	0,0%
0,4;0,8	719.671	8;3	24.706	678.796	8;3	21.899	5,7%
0,4;0,9	733.003	8;3	32.259	534.240	8;3	20.957	27,1%
0,6;0,1	2.636.199	8;5	8.072	2.636.199	8;5	8.072	0,0%
...	...	...	...	...	...	...	0,0%
0,9;0,9	5.423.469	8;3	2.604	5.423.469	8;3	2.604	0,0%

COMPARAÇÃO COM RELAÇÃO AOS VALORES DAS FUNÇÕES-OBJETIVO: A tabela 1 apresenta uma análise das funções-objetivo. Como mencionado anteriormente, toda a massa de dados analisada foi com relação à média das 20 instâncias para um determinado par (τ,R). A coluna "$g1$*" representa o melhor valor alcançado (isto é, melhor média de 20 instâncias), considerando-se todas as combinações de b e f. A coluna "(b;f) de $g1$*" mostra qual combinação de (b;f) propiciou o melhor valor alcançado. A coluna "DP" representa o desvio-padrão das médias dos valores de (b;f) para um determinado par (τ,R). A coluna "diminuição do custo" mostra o quanto o valor mínimo do grupo 2 é menor do que o menor valor do grupo 1. Observar que um desvio-padrão grande representa que a mudança nos valores de b e f alteram bastante os valores da função-objetivo.

Largura da busca e do filtro: Um fato curioso alcançado nos experimentos para o grupo 1 é que o f "ideal" para 25 ordens alcançada por OW & MORTON (1988 e 1989) se não equivalente, é muito próxima da alcançada aqui para o caso de 100 ordens. Outro ponto que pode ser citado é com relação aos parâmetros testados. Nos estudos de OW & MORTON o leitor fica inclinado a entender que a utilização do filtro é boa para qualquer combinação de (τ,R). Para o grupo 1, a utilização do filtro sempre foi vantajosa conforme pode ser observado na tabela 1, apesar de que em alguns casos, a utilização do filtro melhorou a busca em apenas frações de um ponto percentual. No grupo 2, isso nem sempre foi verdade. Os casos onde f=1 representam que o BS sem o filtro é melhor do que o BSF.

Na média geral, o f ideal para o grupo 1 está entre 2 e 4. Para o caso de b, desconsiderando-se o tempo de processamento computacional, quanto maior seu valor melhor. O filtro no grupo 2, na média geral também ficou entre 2 e 4. Um aumento da largura da busca também tende a melhorar os resultados.

Como resultado final e sugestão, pode-se dizer que o filtro utilizado com cautela em casos bem estudados, pode melhorar significativamente o BS com relação ao valor alcançado pela função-objetivo.

Inserção de ociosidade: Para casos específicos, a inserção de ociosidade torna-se absolutamente vital. Especialmente para τ pequeno e R grande, a inserção de ociosidade melhora bastante o valor da função-objetivo. A tabela 1 mostra casos cuja melhoria proporcionada pelo procedimento de inserção de ociosidade levou a função-objetivo diminuir em mais de 40%. Se for levado em consideração que nos casos práticos, onde os valores de τ tendem a ser pequenos e de R ser grandes (poucas ordens atrasarão e as datas de entrega são relativamente bem distribuídas ao longo do horizonte de programação), o procedimento proposto é ainda mais interessante.

Esses resultados indicam que uma boa escolha do tipo de procedimento, e dos parâmetros utilizados no procedimento como por exemplo b e f têm importância significativa para o bom desempenho do procedimento utilizado. Os resultados detalhados para todos os pares (τ;R) testados se encontram em COLIN (1997).

8 CONSIDERAÇÕES ADICIONAIS

Se forem levados em consideração casos reais, a importância da inserção de ociosidade provavelmente seria diminuída. Isso aconteceria devido às empresas sempre terem uma carteira de pedidos que se aproxima de sua capacidade produtiva. Mesmo assim, a inserção de ociosidade como no caso estudado aqui deve ser encarada como uma ferramenta adicional de programação. Seu principal intuito é diminuir os custos referentes ao adiantamento na utilização de recursos.

A inserção de ociosidade freqüente indica que alguma providência deve ser tomada nos outros níveis de decisão. Por exemplo a eliminação de turnos de trabalho, a melhor definição das datas de entrega, o remanejamento de pessoal ou a diminuição do número de horas-extras realizadas podem ser opções mais lucrativas sob o ponto de vista da empresa. A inserção de ociosidade na programação da produção é feita com o intuito de diminuição de custos, mas sem levar em consideração a hipótese de se tomar decisões pertencentes a outros níveis.

Pela complexidade do procedimento, da forma que foi apresentado, parece que o BS só se adequaria num ambiente real de programação a casos de uma máquina. Dessa forma, ele seria utilizado fazendo a programação da máquina gargalo e as outras máquinas deveriam ser programadas com procedimentos mais simples como regras de despacho. Casos desse tipo são bem discutidos em MORTON & PENTICO (1993).

Como sugestão de pesquisas futuras, parece ser viável o desenvolvimento de um procedimento "mais poderoso" com relação aos resultados alcançados pelo BSF com inserção de ociosidade. Em alguns casos isolados, como por exemplo para certos (τ;R) e (b;f), a inserção de ociosidade fez os resultados da busca piorarem. A idéia seria tentar de alguma forma elaborar um limitante superior que utilize ambos, os procedimentos com e sem inserção de ociosidade de modo que o procedimento alcançe o melhor dos dois. Por exemplo, YANO & KIM

(1991) utilizam o procedimento de inserção de ociosidade após a seqüência final ser definida. Por conseqüência, não há como o valor da função-objetivo no caso com inserção de ociosidade ser pior do que no caso sem.

9 REFERÊNCIAS BIBLIOGRÁFICAS

ABDUL-RAZAQ, T. S. & POTTS, C. N.: "Dynamic programming state-space relaxation for single machine scheduling", *Journal of Operational Research Society*, vol. 39, pp. 141-152, 1988.

BAKER, K. R. & MARTIN, J. B.: "An experimental comparison of solution algorithms for the single-machine tardiness problem", *Naval Research Logistics Quarterly*, vol. 21, pp. 187-199, 1974.

COLIN, E. C.: "*Beam search* e inserção de ociosidade no problema de programação de uma máquina em ambiente do tipo JIT", São Paulo, Escola Politécnica, Universidade de São Paulo, dissertação de mestrado, 106p., 1997.

COLIN, E. C. & SHIMIZU, T.: "Algoritmo de programação de máquinas individuais com penalidades distintas para adiantamento e para atraso", submetido à publicação, 1998.

FRY, T. D.; ARMSTRONG, R. D. & BLACKSTONE, J. H.: "Minimizing weighted absolute deviation in single machine scheduling", *IIE Transactions*, vol. 19, pp. 445-450, 1987.

GAREY, M. R.; TARJAN, R. E. & WILFONG, G. T.: "One-processor scheduling with symmetric earliness and tardiness penalties", *Mathematics of Operations Research*, vol. 13, pp. 330-348, 1988.

MORTON, T. E. & PENTICO, D. W.: *Heuristic scheduling systems: with applications to production systems and project management*, Wiley, New York, 1993.

OW, P. S. & MORTON, T. E.: "Filtered beam search in scheduling", *International Journal of Production Research*, vol. 26, pp. 35-62, 1988.

OW, P. S. & MORTON, T. E.: "The single machine early/tardy problem", *Management Science*, vol. 35, pp. 177-191, 1989.

SRINIVASAN, V.: "A hybrid algorithm for the one machine sequencing problem to minimize total tardiness", *Naval Research Logistics Quarterly*, vol. 18, pp. 317-327, 1971.

WILKERSON, L. J. & IRWIN, J. D.: "An improved method for scheduling independent tasks", *AIIE Transactions*, vol. 3, pp. 239-245, 1971.

YANO, C. A. & KIM, Y.-D.: "Algorithms for a class of single-machine weighted tardiness and earliness problems", *European Journal of Operational Research*, vol. 52, pp. 167-178, 1991.

10 APÊNDICE: PSEUDOCÓDIGO DO BSF

Para o entendimento do pseudocódigo, define-se adicionalmente: um ";" separam dois comandos; $C_{s_i^{\nu-1}}$ o horário de término de processamento da seqüência parcial s_i no nível ν–1.

procedimento *Beam Search* Filtrado (b,f,R^0,σ^0)
início
Calcule p_{med} de R^0; F←∅;
para i:=1 **até** n **faça**
  início
  Calcule $\varphi_i(0)$; Insira i de acordo com valores crescentes de $\varphi_i(0)$ em F;
  fim;
para i:=1 **até** max{b,f} **faça**
  início
  Retire de $\bar{s}_i^0$, a ordem de i-ésima posição em F e insira-a em s_i^1 ; Concatene s_i^1 e $\bar{s}_i^1$, gerando σ_i;
  Calcule $g(\sigma_i)$; Insira σ_i ordenada por função objetivo no conjunto A;
  fim;
para i:=1 **até** b **faça**
  Insira a i-ésima seqüência do conjunto A em B;
para ν:=2 **até** n–1 **faça**
  início
  para i:=1 **até** b **faça**
    Auxiliar;
  para j:=1 **até** b **faça**
    ($\sigma_j \in B$)←($\sigma_j \in A$);
  fim;
fim;

Figura 7: pseudocódigo BSF

procedimento Auxiliar
início
R^ν←R^0\conj($s_i^{\nu-1}$); Calcule p_{med} de R^ν; F←∅;
para j:=1 **até** |R^ν| **faça**
  início
  Calcule $\varphi_j(C_{s_i^{\nu-1}})$; Insira j de acordo com valores crescentes de $\varphi_j(C_{s_i^{\nu-1}})$ em F;
  fim;
para j:=1 **até** min{b,|R^ν|} **faça**
  início
  Concatene $s_i^{\nu-1}$ a ordem de posição j-ésima em F, gerando a seqüência parcial s_j^ν ;
  Concatene s_j^ν a $\bar{s}_j^\nu$, gerando σ_j; Calcule $g(\sigma_j)$; Analisados:=min{max{b,f},|R^ν|};
  se (|A|<Analisados) **então** Insira σ_j em A, ordenado de acordo com valores crescentes de $g(\cdot)$;
  caso contrário
    início
    PosInserção:=0;
    para m:=|A| **diminuindo até** 1 **faça**
      se ($g(\sigma_j)$<$g(\sigma_m)$) **então** PosInserção:=m;
    se (PosInserção>=1) **e** (PosInserção<|A|) **então**
      início
      Movimente todas as seqüências em A que ficam após a PosInserção-ésima posição para uma posição posterior, até a posição |A|–1;
      Insira σ_j na (PosInserção)-ésima posição de A;
      fim;
    fim;
  fim;
fim;

Figura 8: procedimento auxiliar do pseudocódigo BSF

Sobre a Verax Consultoria

A Verax é uma empresa de consultoria especializada em gestão. Temos uma ampla gama de experiências e competências como pode ser consultado em www.veraxc.com/areas.htm. Os líderes da empresa já proveram serviços de consultoria para mais de 60 organizações de diferentes segmentos e tamanhos, em mais de 150 projetos.

Informações adicionais

Para informações adicionais você pode nos contatar em contato@veraxc.com ou visite nosso sítio de internet em www.veraxc.com.

Autoria e publicação

Emerson Colin é o autor do documento e sócio da Verax Consultoria.

O documento foi publicado originalmente em Gestão e Produção.

Verax consultoria

Tel: +55-11-3266-7000

Rua Pamplona, 1018 – cj 51 – Jardim Paulista
01405-001 – São Paulo – SP, Brasil