Programação da produção com recursos restritos e disjunções

José Francisco Ferreira Ribeiro
(USP - Universidade de São Paulo, São Carlos, SP)
jffr@icmc.usp.br

Resumo
A programação da produção com recursos restritos e disjunções tem por objetivo determinar a seqüência e o calendário de fabricação em cada uma das máquinas disponíveis na fábrica, otimizando-se um critério. As peças são processadas de acordo com roteiros de fabricação previamente fixados e as durações das operações a executar são conhecidas. Neste artigo, o problema é estudado mediante duas abordagens: programação inteira e teoria dos grafos. O critério de otimização é o tempo total de fabricação. Um programa computacional baseado em teoria dos grafos foi escrito em linguagem C e testado. O programa permitiu a resolução eficiente de vários exemplos, apesar do caráter não-polinomial do problema estudado.
Palavras-Chave: Programação da produção, Restrições disjuntivas, Otimização.

1. Introdução

Em um problema de programação da produção com recursos restritos e disjunções, p peças devem ser fabricadas, e para tanto, dispõe-se de m máquinas. As máquinas executam diferentes tipos de operações e é fornecido um roteiro de fabricação para cada uma das peças. Uma peça, por exemplo, deverá passar sobre a máquina 2 antes da máquina 1, depois sobre a máquina 1 antes da máquina 3, etc. Um seqüenciamento w é uma m-upla (O_1, O_2, ..., O_m), onde O_q designa a ordem de passagem das peças sobre a máquina q. Busca-se determinar o seqüenciamento de tempo total mínimo.

2. Formulação do Problema

Entre os modelos de programação inteira desenvolvidos desde os anos 50, Muth e Thompson (1963) destacam: Wagner (1959), Bowman (1959) e Manne (1960), que usam o método dos planos de corte (Salkin, 1975) para resolução. Balas (1969) aplica o algoritmo de resolução implícita (Salkin, 1975) para resolver o modelo de Manne (1960). Nos anos 70, a resolução dos problemas programação com restrições disjuntivas sofreu um novo impulso com os trabalhos desenvolvidos na área da teoria dos grafos por Roy (1970) e continuada por Balas (1969), Gondran (1974), Gondran e Dostatni (1977), Gondran e Minoux (1985), Carlier (1975, 1978, 1984), Carlier e Chrétienne (1982), Carlier e Pinson (1989).

2.1 Programação Inteira

As variáveis binárias (zero ou um) utilizadas nos modelos de Wagner (1959), Bowman (1959) e Manne (1960), citados por Muth e Thompson (1963) são as seguintes:

y_{ijk} = 1 se a peça i é a j-ésima peça processada na máquina k
0 caso contrário (Wagner, 1959)

x_{At} = 1 se a peça x é processada pela máquina A na t-ésima unidade de tempo
0 caso contrário (Bowman, 1959)

x_{ijk} = 1 se a peça j é processada antes da peça k na máquina i
0 caso contrário (Manne, 1960)

Para a formulação de Manne (1960), os dados do problema são os seguintes:
a) m e p, para {i = 1, ..., m} e {j = 1, ..., p}
b) d_{ij} = tempo de processamento da peça j na máquina i
c) roteiro de fabricação j[1], j[2], ..., j[m] para cada peça j

Seja t_{ij} o tempo inicial de execução da operação j na máquina i. Uma vez que a (r+1)-ésima operação sobre a peça j não pode ser executada antes que a r-ésima operação tenha sido completada, tem-se que:

$$t_{j(r+1),j} \geq t_{j(r),j} + d_{j(r),j} \text{ para } r = 1, ..., m-1 \text{ e para todo } j. \quad (1)$$

A disjunção existente entre as operações j e k (j < k) sobre a máquina i é representada através de:

$$t_{ik} \geq t_{ij} + d_{ij} \text{ se } x_{ijk} = 1 \quad \underline{\text{ou}} \quad t_{ij} \geq t_{ik} + d_{ik} \text{ se } x_{ijk} = 0. \quad (2)$$

Considerando M um número suficientemente grande (M >> max{0, t_{ij}-t_{ik}+d_{ij}, t_{ik} - t_{ij} + d_{ik}}), transforma-se as restrições ou acima, em restrições do tipo e:

$$t_{ij} - t_{ik} \leq d_{ij} + M (1 - x_{ijk}) \quad \underline{\text{e}} \quad t_{ik} - t_{ij} \leq d_{ik} + M\, x_{ijk} \quad (3)$$

Assim, o modelo para o problema é dado por:

minimizar $f = \Sigma\, t_{j(m),j}$

sujeito a (1), (3), $t_{ij} \geq 0$ para todo i, j e $x_{ijk} = 0/1$ para todo i, j, k.

2.2 Teoria dos grafos

Um grafo conjuntivo (somente com restrições do tipo e) G = (X, U) é conjunto finito não vazio X e um conjunto U de pares de elementos de X (Berge, 1970), Gondran e Minoux (1985), Sakarovitch (1984). Um grafo disjuntivo (com restrições do tipo ou) é um grafo W = (G, D), onde G = (X, U, d) é o grafo conjuntivo constituído do conjunto de nós X e do conjunto de arcos U cujos comprimentos d_{ij} são iguais às durações d_i das operações i que estão na origem dos arcos, e D é o conjunto de restrições disjuntivas. Uma disjunção entre 2 operações i e j é uma restrição do tipo ou e deve ser arbitrada de maneira a eliminar-se as disjunções e transformar-se W em um grafo conjuntivo: ou i é executada antes de j e insere-se, então, no grafo W um arco com origem em i e destino em j de comprimento igual à duração da operação i, ou j é executada antes de i e insere-se, então, no grafo W um arco com origem em j e destino em i de comprimento igual à duração da operação j. Um seqüenciamento sobre o grafo disjuntivo W pode ser definido pelos potenciais T = {t[i] | i ∈ X} tal que: a) (i, j) ∈ U : t[i] - t[j] ≥ d_i; b) (i, j) ∈ D : t[j] - t[i] ≥ d_i ou t[i] - [tj] ≥ d_j.

Uma arbitragem é um conjunto A de arcos disjuntivos tal que se (i, j) ∈ A, então (j, i) ∉ A. Assim, o fato do arco (i, j) pertencer a A impõe que a operação i seja executada antes de j. À arbitragem A, associa-se o grafo conjuntivo G(A) = (X, U ∪ A), notado G_A. Uma arbitragem é completa se todas as disjunções foram arbitradas. Uma arbitragem é compatível se o grafo conjuntivo associado à seleção A não contém circuito absorvente. Uma solução para o problema é uma arbitragem completa e compatível. A duração de uma solução é o valor do caminho crítico do grafo conjuntivo associado.

O conjunto de operações executadas sobre uma máquina forma uma clique de disjunção : W = (X, U ∪ D) designa o grafo disjuntivo associado à fabricação e A_o é uma arbitragem completa e compatível. Assim, A_o é uma solução para o problema de seqüenciamento W.

3. Método de Resolução

As restrições disjuntivas são de natureza combinatória: se [i,j] é um par de restrições disjuntivas, tem-se que escolher entre efetuar i antes de j ou j antes de i. Se m é o numero de pares de restrições disjuntivas, o número de escolhas possíveis será da ordem de 2^m, o que torna toda enumeração impossível, caso m ultrapasse dois dígitos (Garey e Johnson, 1979; Gondran, 1974).

O método desenvolvido neste artigo para a resolução do problema de programação é dividido em duas partes: a 1ª parte consiste na aplicação de propriedades teóricas para as soluções de um problema de programação com restrições disjuntivas propostas por Carlier (1978, 1984). A 2ª parte utiliza o método de penalidades descrito em Gondran e Minoux (1985).

3.1 Ordem Total Associada a uma Arbitragem Compatível

Escreve-se iρj se existe em G_{A_0} um caminho de i a j, de valor maior ou igual d[i], ou se i = j.

Lema 1: A relação ρ induz uma ordem total.

Demonstração: 1) ρ é reflexiva por definição; 2) ρ é anti-simétrica: iρj, jρi e i≠j ocasionam a existência de circuito em G_{A_0}, de valor superior a d[i]+d[j], portanto absorvente, o que contradiz a compatibilidade de A_o; 3) ρ é transitiva: sejam i, j, k três nós do grafo G_{A_0}; iρj ⇒ $l_o(x_i,x_j) \geq d[i]$; jρk ⇒ $l_o(x_j,x_k) \geq d[j]$; por definição tem-se: $l_o(x_i,x_k) \geq l_o(x_i,x_j) + l_o(x_j,x_k) \geq d[i] + d[j] \Rightarrow l_o(x_i,x_k) \geq d[i] \Rightarrow$ iρk; portanto é transitiva. n

Conseqüência: A clique tem um elemento **ê**, e um elemento **ŝ** . Denomina-se **ê** e **ŝ** entrada e saída da clique *C*. Procura-se encontrar para cada clique **ê** e **ŝ**), prosseguindo desta forma até que a cardinalidade da clique seja igual a zero ou um.

Teorema de Existência

Existe em G_{A_O} um caminho, de 1 a n, de valor ≥ $l_o(1,\hat{e}) + D(C) + l_o(\hat{s},n) - d[\hat{s}]$.

Demonstração: Seja $C = \{x_1, x_2, \ldots, x_c\}$ o conjunto dos elementos de *C*, relacionados pela ordem ρ, com : $x_1 \equiv \hat{e}$ e $x_c \equiv \hat{s}$. Tem-se, pela definição de ρ, que : $l_o(x_i,x_{i+1}) \geq d[x_i]$; pode-se ter a desigualdade restrita, por exemplo, se existe em G_{A_0} um caminho que vai de x_i a x_{i+1} de valor superior a $d[x_i]$. Somando-se os caminhos: $l_o(x_1,x_2) \geq d[x_1]$, $l_o(x_2,x_3) \geq d[x_2]$, ... , $l_o(x_{c-1},x_c) \geq d[x_{c-1}]$, obtém-se: $l_o(x_1,x_2) + l_o(x_2,x_3) + \ldots + l_o(x_{c-1},x_c) \geq d[x_1] + d[x_2] + \ldots + d[x_{c-1}]$. Lembrando que:

$d[x_1] + d[x_2] + \ldots + d[x_c] = D(C) \Rightarrow$

$\Rightarrow d[x_1] + d[x_2] + \ldots + d[x_{c-1}] = D(C) - d[x_c]$ (onde $d[x_c] = d[\hat{s}]$) $\Rightarrow$

$$\Rightarrow \sum_{x_i \in C-\{x_c\}} d[x_i] = D(C) - d[\hat{s}]$$

$$l_o(x_1,x_2) + l_o(x_2,x_3) + \ldots + l_o(x_{c-1},x_c) = \sum_{x_i \in C-\{x_c\}} l_o(x_i, x_{i+1}) \geq \sum_{x_i \in C-\{x_c\}} d[x_i] = D(C) - d[\hat{s}].$$

De onde resulta imediatamente que:

$$l_o(1,\hat{e}) + \sum_{x_i \in C-\{x_c\}} l_o(x_i, x_{i+1}) + l_o(\hat{s},n) \geq l_o(1,\hat{e}) + D(C) + l_o(\hat{s},n) - d[\hat{s}].$$ n

O caminho associado é obtido na ligação do caminho de valor máximo de 1 a ê aos caminhos de valor máximo de x_i a x_{i+1} (para $1 \leq i \leq c-1$) e, em seguida, no caminho de valor máximo de $x_c = \hat{s}$ a n.

3.2 Propriedades das Soluções de um Problema de Programação

Seja *f* a duração da melhor solução conhecida para o problema estudado (obtida, por exemplo, heuristicamente), E_1 o conjunto dos elementos ê tais que existe uma solução de duração inferior a *f* tendo ê como entrada da clique *C* e S_1, o conjunto dos elementos ŝ tal que existe uma solução de duração inferior a *f* tendo ŝ como saída da clique *C*. Seja E um conjunto incluso em *C* contendo E_1; S um conjunto incluso em *C* contendo S_1; e sejam i e j elementos de *C*. Seja Ψ uma solução; ê ∈ *C* (resp. ŝ ∈ *C*) é chamado entrada (resp. saída) da clique *C* se ê (resp. ŝ) é seqüenciado em Ψ antes (resp. depois) de todas as operações de *C*.

Teorema para Avaliação da Duração de Programação

A expressão: $\min_{e \in E}(l(1,e)) + D(C) + \min_{e \in S}(l(s,n) - d[s])$ é uma avaliação por *default* da duração ótima.

Demonstração: Basta mostrar que este número é menor que : $l_o(1,\hat{e}) + D(C) + l_o(\hat{s},n) - d[\hat{s}]$.

1) $\hat{e} \in E_1$ e $E_1 \subset E \Rightarrow \hat{e} \in E$; assim, para $e \in E$, $l_o(1,\hat{e}) \geq l_o(1,e)$; ou $l_o(1,e) \geq l(1,e) \Rightarrow \min_{e \in E} l_o(1,e) \geq \min_{e \in E} l(1,e)$; por transitividade tem-se: $l_o(1,\hat{e}) \geq \min_{e \in E} l(1,e)$; (1)

2) $\hat{s} \in S_1$ e $S_1 \subset S \Rightarrow \hat{s} \in S$; assim $l_o(\hat{s},n) - d[\hat{s}] \geq \min_{s \in S}(l_o(s,n) - d[s])$

ou $(l_o(s,n) - d[s]) \geq (l(s,n) - d[s]) \Rightarrow \min_{s \in S}(l_o(s,n) - d[s]) \geq \min_{s \in S}(l(s,n) - d[s])$;

por transitividade tem-se: $l_o(\hat{s},n) - d[\hat{s}] \geq \min_{s \in S}(l(s,n) - d[s])$; (2)

efetuando-se (1) + (2) e acrescentando-se $D(C) \geq D(C)$ obtém-se:

$l_o(1,\hat{e}) + D(C) + l_o(\hat{s},n) - d[\hat{s}] \geq \min_{s \in S}(l(1,e)) + D(C) + \min_{s \in S}(l(s,n) - d[s])$. n

Determinação da Entrada e da Saída de uma Clique de Disjunção

Quanto mais próximos de 1 os cardinais de E e S, melhor é a avaliação. A proposição abaixo permite retirar de E e S alguns nós; inicialmente considera-se E = S = *C*.

Proposição 1: Sejam as seguintes condições:

(1) $l(1,i) + D(C) + \min_{s \in S-\{i\}}(l(s,n) - d[s]) > f$

(2) $\min_{s \in E-\{j\}}(l(1,e)) + D(C) + l(j,n) - d[j] > f$

Se (1) ocorre então $i \notin E_1$. Se (2) ocorre então $j \notin S_1$.

Prova: Basta supor que (1) (resp. (2)) é satisfeita e que $i \in E_1$ (resp. $j \in S_1$). O primeiro membro da desigualdade (1) (resp. (2)) é um limitante inferior da duração quando i (resp. j) é saída (resp. entrada) de *C*. Assim, esta duração é estritamente maior que *f* e chega-se a uma contradição.n

Seja Y um subconjunto de *C* ($Y \subseteq C$) e H(Y) definido por: $H(Y) = \min\{l(1,i) / i \in Y\} + \sum\{d[i] / i \in Y\} + \min\{(l(i,n) - d[i]) / i \in Y\}$. Seja $k \in C$. Tem-se: $H(C-\{k\}) + d[k] > f$ (3)

Lema 2: Se (3) for satisfeita, então, em qualquer solução Ψ, a operação k é seqüenciada ou antes de todas as operações de *C* ou depois delas.
Demonstração: Seja Ψ uma solução tal que k não seja entrada nem saída de *C*. Então, as operações de *C* são executadas em alguma ordem $i_1, i_2, ..., i_h$ com $i_1 \neq k$ e $i_h \neq k$. Assim, a duração desta solução é maior que : $l(1,i_1) + \sum_{i \in C} d[i] + l(i_h,n) - d[i_h]$. Ou seja :

$$l_0(1,i_1) + \sum_{i_k \in C-\{i_h\}} l_0(i_k,i_{k+1}) + l_0(i_h,n) > \quad l(1,i_1) + \sum_{i \in C} d[i] + l(i_h,n) - d[i_h] \geq$$
$$\min_{i \in C-\{k\}} (l(1,i)) + \sum_{i \in C-\{k\}} d[i] + d[k] + \min_{i \in C-\{k\}} (l(i,n) - d[i]) =$$
$$= H(C\text{-}\{k\}) + d[k] \Rightarrow f > H(C\text{-}\{k\}) + d[k], \text{ o que contradiz (3). n}$$

Proposição 2: Se as condições (1) (proposição 1) e (3) são satisfeitas, então k é a saída da clique *C* em qualquer solução Ψ.
Prova: Se ocorre (1) então $k \notin E_1$. Assim se (3) é satisfeita, k só poderá ser saída. n

Proposição 3: Se as condições (2) (proposição 1) e (3) são satisfeitas, então k é a entrada da clique *C* em qualquer solução Ψ.
Prova: Se ocorre (2) então $k \notin S_1$. Assim se (3) é satisfeita, k só pode ser entrada. n

Cálculo dos Elementos l(1,i) e l(j,n) da Matriz L (i e j ∈ G).
As duas proposições abaixo permitem calcular o valor de l(1,i) e l(j,n) após ter-se determinado o conjunto dos elementos de entrada e saída para cada clique. A proposição 4 é aplicada somente quando o cardinal de E ou de S é igual a 1.

Proposição 4: (1) Se e é a entrada da clique *C*, então: $l(e,n) \geq D(C) + \min_{s \in S}(l(s,n) - d[s])$; (2) Se s é saída da clique *C*, então: $l(1,s) \geq \min_{e \in E} l(1,e) + D(C) - d[s]$.

Proposição 5: (1) Se $i \notin E$ ($i \in C$), então $l(1,i) \geq \min_{e \in E}(l(1,e) + d[e])$; (2) Se $j \notin S$ ($j \in C$), então $l(j,n) \geq d[j] + \min_{s \in S} l(s,n)$.
Prova: No grafo G, i é um descendente da entrada e, j um antecedente da saída. n

Arbitragens Triviais
A proposição (6), abaixo, é aplicada somente quando o cardinal de E ou S é 1. Ela permite introduzir uma conjunção entre *e* (ou *s*) e todos os outros nós da clique *C*.

Proposição 6: (1) Se *e* é a entrada da clique *C*, então os arcos disjuntivos (*e*,k) ($k \in C\text{-}\{e\}$) serão selecionados em qualquer solução. (2) Se *s* é a saída da clique *C*, então os arcos disjuntivos (k,*s*) ($k \in C\text{-}\{s\}$) serão selecionados em qualquer solução. (3) Se $E_1 = \varnothing$ ou $S_1 = \varnothing$, o problema não tem solução de duração inferior a *f*.
Prova: Pela definição de E_1 e S_1n

Seleção Imediata de uma Restrição Disjuntiva
Quando *C* contém exatamente duas operações i e j, obtém-se:

Proposição 7: (1) Se l(1,i) + d[i] + l(j,n) > f e (2) l(1,j) + d[j] + l(1,n) ≤ f, então o arco disjuntivo [i,j] será selecionado no sentido (j,i).
Prova: A escolha de (j,i) é decorrente direto das condições (1) e (2). n

3.3 Regra de Separação e Penalidades

O método *branch-and-bound* percorre o conjunto das soluções em uma arborescência, sem que nenhum nó seja aberto (Gondran e Minoux, 1985; Sakarovitch, 1984). A avaliação por *default* permite a exploração implícita de ramos inteiros da arborescência sem o exame explícito de todos os nós. Considere-se um par disjuntivo [i,j] ainda não arbitrado. Suponha que a tarefa i seja executada antes de j. Se t[i] + d[i] > T[j] então o caminho mais longo do grafo aumentará pelo menos de r(i,j) = t[i] + d[i] - T[j]. Assim r(i,j) representa a penalidade de se escolher i antes de j. Da mesma forma, r(j,i) representa a penalidade de se escolher j antes de i. Seja H o grafo inicial (grafo conjuntivo com as disjunções a serem arbitradas). Procedendo-se à separação do nó, segundo a disjunção [i,j], tem-se : $H° = H \cap \{(i,j)\}$ e $H^1 = H \cap \{(j,i)\}$, ou seja, H° (resp. H^1) corresponde ao novo grafo disjuntivo no qual a disjunção [i,j] foi arbitrada no sentido (i,j) (resp. (j,i)). Pode-se entender por H° (resp. H^1) o espaço das soluções que possuem a disjunção arbitrada no sentido (i,j) (resp. (j,i)). Considere o grafo H°. A introdução da condição (i,j) (fazer i antes de j) permite definir, a partir da avaliação por falta de H, uma nova avaliação por falta de H°, av(H°) = av_1(H) + r(i,j), em que av_1(H) é o valor do caminho máximo no grafo conjuntivo de H. Da mesma maneira, tem-se: av(H^1) = av_1(H) + r(j,i). Como, de qualquer forma, é preciso escolher (i,j) ou (j,i), tem-se que av_1(H) + θ(i,j) é uma avaliação de H, em que θ(i,j) = min(r(i,j), r(j,i)). Sendo K_2 o subconjunto das disjunções não arbitradas, define-se uma nova avaliação por falta: $av_2(H) = av_1(H) + \max_{[i,j]\in K_2} \theta(i,j)$. Seja γ(i,j) = | r(i,j) - r(j,i) |. Esta quantidade γ(i,j) é a penalidade da função objetivo, decorrente de não se arbitrar [i,j] no sentido correspondente a θ(i,j) = min(r(i,j), r(j,i)); em outras palavras, se θ(i,j) = min(r(i,j), r(j,i)) = r(i,j) e escolhe-se arbitrar no sentido j para i, a duração da programação será aumentada de γ(i,j). Denomina-se γ(i,j) de penalidade associada à separação de [i,j]; ela representa de qualquer maneira o arrependimento de não se arbitrar [i,j] pelo min(r(i,j), r(j,i)). Assim, a regra heurística deduzida deste fato consiste em arbitrar a disjunção que maximiza γ(i,j) / [i,j] ∈ K_2, no sentido correspondente ao menor atraso. Introduzem-se, assim, quatro parâmetros que correspondem às penalidades:

1ª) r(i,j) (resp. r(j,i)) representa a penalidade sobre av_1(H) quando se fixa a arbitragem (i,j) (resp. (j,i));
2ª) θ(i,j) = min(r(i,j), r(j,i)) representa a penalidade sobre av_1(H) quando se acrescenta a arbitragem, sobre a disjunção correspondente [i,j];
3ª) max θ(i,j) ([i,j] ∈ K_2) representa a penalidade sobre av_1(H) quando se arbitra [i,j] ∈ K_2 correspondente ao min(r(i,j),r(j,i));
4ª) γ(i,j) representa a penalidade sobre av_1(H) quando não se arbitra [i,j] pelo min(r(i,j), r(j,i)).

Essas penalidades permitem avaliar os atrasos de fabricação devidos à arbitragem de uma disjunção [i,j]. O algoritmo 1, que utiliza tais penalidades é apresentado abaixo.

algoritmo 1
A_k (S_k) = conjunto das operações que antecedem (sucedem) a operação k
enquanto existir uma disjunção [i,j] não arbitrada **fazer**
para toda disjunção [i,j] não arbitrada **fazer**
r(i,j):= max(0, t[i] + d[i] - T[j]); r(j,i):= max(0, t[j] + d[j] - T[i])
θ(i,j):= min(r(i,j), r(j,i)); γ(i,j):= | r(i,j) - r(j,i) |
fim para
para toda disjunção [i,j] não arbitrada **fazer**
- escolher a disjunção [i,j] não arbitrada que tenha γ(i,j) máximo e, em caso de igualdade, escolher aquela que tiver θ(i,j) máximo
- arbitrar a disjunção [i,j] no sentido do menor atraso
fim para
para toda operação k **fazer** calcular t[k] e T[k] (Bellman, 1958) **fim para**
fim enquanto
fim algoritmo 1

3.4 Método Geral

O método de resolução do problema de programação com recursos restritos e disjunções proposto neste artigo, é composto de duas etapas:
1ª) proposições teóricas de Carlier (1978, 1984)
2ª) cálculo das penalidades de Gondran e Minoux (1985)

algoritmo 2
A_k (S_k) = conjunto das operações que antecedem (sucedem) a operação k
r = índice ou número da máquina; C_r = clique de disjunção da máquina r
E_r (S_k) = conjunto dos nós, pertencentes à C_r que podem ser entrada (saída)
L1[k] (Ln[k]) = comprimento do caminho máximo indo do nó 1 (k) ao nó k (n)
para toda C_r **fazer** $C_r = E_r = S_r$ **fim para**
enquanto existir uma disjunção não arbitrada **fazer**
para toda operação k **fazer** calcular L1[k] e Ln[k] (Bellman, 1958) **fim para**
para toda clique C_r **fazer**
calcular a avaliação utilizando o teorema para avaliação da duração de programação
fim para
para a clique que possui a maior avaliação **fazer** aplicar a proposição 1
se E_r e S_r não são vazios **então** aplicar as proposições 2, 3, 4, 5 e 6
fim para
para toda clique que possuir apenas dois elementos **fazer** aplicar a proposição 7
fim para
se foi possível arbitrar alguma disjunção **então** árbitro := *true*
senão árbitro := *false*
se árbitro = *false* **então** algoritmo 1 (Gondran e Minoux, 1985)
árbitro := *true*
fim se
fim enquanto
fim algoritmo 2

O algoritmo 2 resume os passos fundamentais do método geral. Na implementação computacional do algoritmo 2, efetuou-se o cálculo de dois vetores L1[i] e Ln[i], em que L1[i] = l(1,i) e Ln[i] = l(i,n), no lugar da matriz L, originalmente proposta por Carlier (1978, 1984). Isto foi realizado, pois a matriz L ocupa excessivamente a memória do computador e compromete a resolução de problemas com mais de uma centena de nós (Noronha e Ribeiro, 1996).

4. Comparação dos Algoritmos

Os resultados obtidos com os programas computacionais correspondentes aos algoritmos 1 (cálculo das penalidades) e 2 (cálculo das proposições e das penalidades) sobre exemplos da literatura, são apresentados na Tabela 1.

Um estudo comparativo de desempenho, entre o algoritmo 2 e um método exato, é fornecido na Tabela 2.

Os programas 1 e 2 foram escritos em linguagem C e estão implantados em um microcomputador IBM-PC compatível.

A notação utilizada nas Tabs. 1 e 2 é a seguinte: m = nº de máquinas, p = nº de peças, nbnós = nº total de nós, nbdis = nº de disjunções no grafo inicial, f = duração da programação, τ = tempo de cálculo (cpu) em segundos, em um microcomputador Pentium, 200 MHz, 64 MBytes, ***** = tempo de cálculo < 1 sec.

A Tabela 2 mostra a duração da programação e os respectivos tempos de cálculo (cpu), em sec., para o programa 2 e para o programa desenvolvido por Carlier e Pinson (1989), com base no método de separação e avaliação, para determinação da solução ótima.

Deve-se observar que Carlier e Pinson (1989) utilizaram um computador PRIME 2655 em seus experimentos.

Os resultados obtidos apresentam, em média, um erro de 5,4%. Para os exemplos 8 e 12 foi possível obter a solução ótima; para o exemplo 14, que é constituído de 950 disjunções, o erro foi de 13,5%, sendo este o maior de todos.

					Algoritmo 1		Algoritmo 2	
Exemplo	p	m	nbnós	nbdis	f	τ	f	τ
1	3	2	8	6	31	*	31	*
2	3	4	14	12	34	*	32	*
3	4	3	14	18	27	*	27	*
4	3	4	13	10	105	*	105	*
5	4	3	14	18	31	*	31	*
6	3	2	8	6	8	*	8	*
7	5	4	15	15	13	*	13	*
8	11	5	57	275	7534	*	7038	*
9	12	5	62	330	8848	*	7725	1
10	14	4	58	364	8736	*	8247	1
11	7	7	51	147	6829	*	6749	*
12	6	6	38	90	57	*	55	*
13	10	10	102	450	1040	*	1040	5
14	20	5	102	950	1323	4	1323	9

Tabela 1: Resultados Obtidos

Exemplo	Algoritmo 2		Solução Ótima	
	f	τ	*f*	τ
8	7038	*	7038	41
9	7725	1	7312	14
10	8247	1	8003	64
11	6749	*	6558	53
12	55	*	55	1
13	1040	5	930	17985
14	1323	9	1165	1448

Tabela 2: Estudo Comparativo

5. Conclusão e Comentários

A resolução do problema de programação da produção com recursos restritos e disjunções por meio da teoria dos grafos combinada com heurísticas, tal como foi feito nesse artigo, é uma alternativa de solução que se mostrou eficiente, e assim, pode ser utilizada para a implementação de pacotes computacionais com o objetivo de utilizá-los em tempo real nas indústrias. Uma das razões da pouca aplicação industrial dos estudos de seqüenciamento e programação da produção advém do fato de a formulação matemática realizada para o problema ignorar variáveis importantes dos ambientes reais. Outra razão é o alto tempo computacional exigido para a resolução, o que impede a sua utilização em tempo real.

Referências

Balas, E., "Discrete sequencing via disjunctive graphs: an implicit enumeration algoritm", *Oper. Res*., 26, pp. 111-120, 1969.

Bellman, R. E., "On a routing problem", *Quart. Appl. Math*., 16, pp. 87-90, 1958

Berge, C., *Graphes et Hypergraphes*, Gauthier-Villars, 1970.

Bowman, E. H., "The schedule sequencing problem", *Oper. Res*., 7, 5, pp. 621-624, 1959.

Carlier, J., "Disjonctions dans l'ordonnancement", *RAIRO*, 2, pp. 83-100, 1975.

Carlier, J., "Ordonnancement à contraintes disjonctives", *RAIRO*,12, pp. 333-351, 1978.

Carlier, J., *Problèmes d'ordonnancement à contraintes de ressources*, Tese de Doutorado Un. P. et M. Curie, Paris, França, 1984.

Carlier, J., Chrétienne, P., Les problèmes d'ordonnancement, *RAIRO*, 16,3, pp. 175-217, 1982.

Carlier, J., Pinson, E., "A branch-and-bound method for solving the job shop problem", *Manag. Sc*., 35, 2, pp. 164-176, 1989.

Garey, M. R., Johnson, D. S., *Computers and intractability: np-completeness,* Freeman, 1979.

Gondran, M., Minoux, M., *Graphes et algorithmes*, Eyrolles, 1985.

Manne, A. S., "The job-shop scheduling problem", *Oper. Res*., pp. 219-223, 1960.

Muth, J. F., Thompson, G. L., *Industrial Scheduling*, Prentice Hall, 1963.

Noronha, A. B., Ribeiro, J. F. F., "Programação de operações", *Revista Gestão e Produção*, 3, 2, pp. 204-219, 1996.

Roy, B., *Algèbre moderne et théorie des graphes*, Dunod, 1970.

Sakarovitch, M., *Optimisation combinatoire: programmation discrète*, Hermann, 1984.

Salkin, H. M., *Integer Programming*, Addison Wesley, 1975.

Wagner, H. M., "An integer linear programming model for machine scheduling", *Naval Res. Logist. Quart*., 6, 2, pp. 131-140, 1959.