v.3, n.2, p. 204-220, ago. 1996

PROGRAMAÇÃO DE OPERAÇÕES COM RESTRIÇÕES DISJUNTIVAS

Adriana Backx Noronha
UNICAMP-FEE-DCA - Caixa Postal 6101
Campinas (SP) - Brasil

José Francisco Ferreira Ribeiro
Cassilda Maria Ribeiro
USP- ICMSC-SCE - Caixa Postal 668
São Carlos (SP) - Brasil

Resumo

O problema de programação de operações em um sistema de produção consiste em determinar a seqüência e o calendário de operações a processar em cada uma das máquinas disponíveis na fábrica, de modo que a duração total de execução da programação seja mínima. As peças são processadas de acordo com roteiros de fabricação fixos e as durações operatórias são conhecidas. Neste artigo, o problema da programação de operações com restrições disjuntivas é estudado mediante duas abordagens: programação inteira e teoria dos grafos. Um programa computacional baseado na teoria dos grafos foi desenvolvido e testado. Esse programa permitiu a resolução eficiente de vários exemplos, apesar do caráter não-polinomial do problema estudado.

Palavras-chave: programação de operações, produção intermitente, programação inteira, teoria dos grafos

1. Introdução

A fabricação de um conjunto de produtos (ou peças ou lotes de peças) deve ser realizada segundo um plano previamente estabelecido e o processamento dessa fabricação deve ser constantemente controlado. A coordenação planejamento-controle pode assegurar uma melhor utilização do tempo disponível e o respeito aos prazos de entrega, permitir aumentos de produtividade e a utilização

racional dos recursos existentes (máquinas, mão-de-obra, matérias primas), garantir uma produção eficiente, a custo razoável, etc.

Um plano de fabricação é uma lista de operações (ou atividades, ou tarefas) com indicação da ordem em que serão realizadas e da data - determinada por meio de algum algoritmo - em que terá início a execução de cada uma delas. Uma operação é caracterizada pela sua duração e pela posição que ocupa no roteiro (ou processo) de fabricação do produto sobre o qual é executada.

Os dados de um problema de programação de operações são as tarefas (operações ou atividades) e suas características, as restrições, os recursos e a função econômica. Denota-se, em geral, por I = {1,2,...,n} o conjunto das operações e por d[i] a duração da operação i, se essa duração não depende dos recursos que lhe são alocados. As datas de início de execução mais cedo e mais tarde da tarefa i são denotadas respectivamente por t[i] e T[i]. As tarefas são freqüentemente ligadas entre si mediante relações de anterioridade. Caso contrário, diz-se que elas são independentes. A restrição de anterioridade mais geral entre duas tarefas i e j, chamada restrição potencial, escreve-se da forma t[j] - t[i] ≥ a(i,j). Tal restrição permite exprimir a sucessão simples entre as tarefas, quando a(i,j) = d[i].

Em certos casos, a execução de uma tarefa pode ser dividida em partes *(preemptive)*, ou seja, uma determinada operação de usinagem pode ser interrompida e reiniciada após a execução de outras operações. Nesse caso, as técnicas de Programação Linear são freqüentemente utilizadas para a resolução do problema.

O estabelecimento de um plano de fabricação deve levar em consideração as restrições associadas a cada problema específico. Estas restrições podem ser resumidas da seguinte maneira: 1ª) restrições potenciais: a execução de uma operação j só pode ser iniciada após o término da execução de uma operação i. Exemplo: i e j são duas operações consecutivas sobre um mesmo produto; 2ª) restrições disjuntivas: um conjunto de operações que não podem ser executadas simultaneamente. Exemplo: as operações que utilizam uma mesma máquina; 3ª) restrições cumulativas: os meios necessários para a execução de um conjunto de operações são limitados. Exemplo: o número de máquinas disponíveis para a execução de uma operação é igual a 2.

A ordem e o calendário de fabricação são definidos satisfazendo-se as restrições existentes e segundo um critério de otimização. Entre os diferentes critérios de otimização mais empregados, podemos citar: 1º) minimização da duração total de fabricação *(makespan)*; 2º) minimização dos atrasos de fabricação (do maior atraso, da soma dos atrasos, da soma ponderada dos atrasos, do número ponderado de operações em atraso), ou seja, das diferenças entre as datas planejadas de fim de fabricação e as datas previstas de entrega *(due dates)*; 3º) minimização de um custo.

Distinguem-se na literatura três categorias principais do problema de sequenciamento ou programação de operações, segundo as restrições impostas às tarefas: 1º) *Flow Shop:* neste caso, a ordem de passagem das peças sobre as máquinas é imposta e é a mesma para todas as peças. Trata-se de um caso particular, em que o roteiro de fabricação é único e as p peças utilizam as m máquinas na mesma ordem, por exemplo, 1,2,...,m. 2º) *Job Shop:* a ordem de passagem das peças sobre as máquinas também é imposta, mas pode diferir segundo as peças; isto é, neste caso o roteiro de fabricação não é necessariamente igual para todas as peças. Trata-se de um problema de difícil tratamento, que exige a resolução de um problema matemático bastante complexo. 3º) *Open Shop:* nenhuma ordem é imposta. Este é um modelo de fábrica menos restrito que o *job-shop* e o *flow-shop*, pois a ordem de execução das tarefas de um trabalho é livre. Como anteriormente, m máquinas estão disponíveis para executar n

trabalhos. Cada trabalho compreende, por exemplo, k tarefas. Uma tarefa do trabalho i deve ser tratada pela máquina j em uma duração d_{ij}. Se uma máquina não for utilizada para o trabalho i, considera-se $d_{ij} = 0$. Na prática, podemos encontrar o problema de *open shop* na organização de exames médicos em um hospital ou no planejamento de uma oficina de reparos de automóveis. Nestes dois exemplos, os trabalhos são respectivamente os pacientes e os carros; as tarefas são os exames médicos e as reparações a efetuar; as máquinas são as salas ou aparelhos de exame (raio-x, endoscopia, eletros, etc.) e os postos de trabalho (carburação, funilaria, lavagem, etc.). A complexidade do problema é considerável.

No problema de *job shop*, objeto de interesse deste trabalho, as p peças a fabricar são processadas em m máquinas adotando-se as seguintes hipóteses e definições: 1ª) uma máquina pode processar uma única peça de cada vez; 2ª) a passagem de uma peça por uma máquina é chamada operação; 3ª) uma operação não pode ser interrompida *(not preemptive)*; 4ª) o roteiro de fabricação de uma peça é formado por diferentes operações e constitui um dado do problema; 5ª) a ordem de execução das operações nas máquinas é desconhecida e a resolução do problema consiste em determinar essa ordem e o calendário de fabricação associado, procurando otimizar segundo um dado critério, por exemplo: duração da programação, atrasos de fabricação quando se dispõe de um prazo máximo de entrega, etc.

2. Revisão Bibliográfica

O problema de se atribuir um intervalo de tempo disponível em uma máquina às operações a executar em uma fábrica com produção do tipo *job shop* faz parte da classe de problemas combinatórios de difícil resolução: os problemas np-completos (GAREY & JOHNSON, 1979). Dispõe-se de algoritmo de resolução polinomial unicamente no caso de problemas de dimensão reduzida, por exemplo, número de máquinas ≤ 2, para a definição de uma ordem de fabricação por meio do algoritmo de JOHNSON (1954). Esses problemas de pequeno porte podem, entretanto, ser resolvidos de maneira ótima por qualquer método: a busca exaustiva, por exemplo. Mesmo apresentando um número menor de variáveis e restrições que os modelos de WAGNER (1959) e BOWMAN (1959), o modelo de MANNE (1960) requer a determinação de $m*C_{p,2}$ variáveis bivalentes (NEMHAUSER & WOLSEY, 1988), o que impossibilita a resolução de um problema de porte médio em um tempo computacional aceitável.

Assim, o procedimento de busca exaustiva não pode ser utilizado para a resolução dos problemas de programação de operações de médio ou grande porte e o emprego de algoritmos aproximados, nesses casos, é uma técnica bastante difundida. Exemplos: busca de um ótimo local ou de uma melhor solução na vizinhança de uma solução inicial, decomposição (espacial, temporal, etc.) do problema a resolver em subproblemas de menor dimensão, realização de uma parada antes da obtenção da solução ótima por métodos exatos, etc.

Em CARLIER & CHRÉTIENNE (1982) e CARLIER (1984) são realizados estudos sobre as diferentes versões existentes do problema de programação de operações: problema com uma máquina, problema com m máquinas, problemas com restrições unicamente relacionadas às gamas de fabricação, problemas com restrições disjuntivas, etc. Uma revisão dos métodos capazes de operar em tempo real é apresentada em YAMAMOTO & NOF (1985); sendo exemplos a citar neste caso os métodos propostos por ROUBELLAT &

THOMAS (1988) e FERREIRA RIBEIRO & PRADIN (1991).

A resolução do problema de programação pelo método de separação e avaliação *(branch-and-bound)*, tratando as disjunções em um grafo por meio de procedimentos heurísticos é proposta por BAKER (1974) e NORONHA & FERREIRA RIBEIRO (1994a, b, c). NASR & ELSAYED (1990) propõem dois algoritmos para a obtenção de uma solução aproximada por meio da decomposição do problema global em subproblemas mais fáceis de resolver. GONDRAN & DOSTATNI (1977) propõem uma regra heurística para a resolução dos conflitos de utilização das máquinas em um grafo disjuntivo. DUPONT (1986) utiliza a técnica dos recozimentos simulados. No caso de problemas de grande porte, PORTMANN (1988) e MEGUELATI (1988) propõem uma parada antecipada após o processamento de um certo número de iterações por métodos exatos. YAMAMOTO (1977), PROTH (1986), DIDRI & PROTH (1987), PORTMANN (1988), MEGUELATI (1988), FERREIRA RIBEIRO & PRADIN (1991), FERREIRA RIBEIRO & RIBEIRO (1993), FERREIRA RIBEIRO *et al*. (1993), OLIVEIRA *et al*. (1994) utilizam a decomposição espacial e/ou temporal do problema para sequenciar subproblemas de porte reduzido. NEPOMIASTCHY (1978) propõe um algoritmo de busca de ótimo local e definição de uma programação admissível.

A busca arborescente por separação e avaliação (associada ou não a heurísticas) é uma das técnicas mais utilizadas para resolver o problema de programação de operações. Entre os métodos que a empregam, podemos citar: ASHOUR & HIREMATH (1973), GONDRAN (1974), CARLIER (1975, 1978, 1984), YAMAMOTO (1977), YAMAMOTO & NOF (1985), HUTCHISON & CHANG (1990), NORONHA & FERREIRA RIBEIRO (1994a, b, c). A programação dinâmica é proposta por BAKER & SCHRAGE (1978). A geração de um conjunto de planos de fabricação admissíveis ao fim de ajuda à decisão é proposta por ERSCHLER *et al*. (1976), ERSCHLER *et al*. (1979) e ROUBELLAT & THOMAS (1988).

Os métodos propostos na literatura podem ser divididos em duas grandes classes, segundo o critério de otimização adotado: Na primeira dessas classes agrupam-se os métodos que minimizam a duração total de fabricação *(makespan)* e, na segunda, os métodos que minimizam os atrasos de fabricação. Como exemplos de métodos da primeira classe, podem ser citados o PERT, o CPM e o MPM (para problemas não-disjuntivos) e outros métodos, tais como ASHOUR & HIREMATH (1973), GONDRAN (1974), BAKER (1974), KAUFMANN & LABORDÈRE (1974), CARLIER (1975, 1978, 1984), YAMAMOTO (1977), GONDRAN & DOSTATNI (1977), BAKER & SCHRAGE (1978), YAMAMOTO & NOF (1985), DUPONT (1986), HUTCHISON & CHANG (1990), NORONHA & FERREIRA RIBEIRO (1994 a, b, c), OLIVEIRA *et al*. (1994), etc. Na segunda classe, podem-se enumerar os métodos de PROTH (1986), PORTMANN (1988), MEGUELATI (1988), bem como o algoritmo de DIDRI & PROTH (1985) para estabelecer a seqüência ótima de produção em caso de máquina única.

Como exceção às duas classes citadas acima, podem-se citar o método de NASR & ELSAYED (1990) que minimiza o tempo de programação médio para execução das operações, o método de NEPOMIASTCHY (1978) que minimiza um custo suplementar devido à violação das restrições, os métodos de FERREIRA RIBEIRO & PRADIN (1991), FERREIRA RIBEIRO & RIBEIRO (1993) e FERREIRA RIBEIRO *et al*. (1993), que permite a utilização dos dois critérios, e os métodos de ERSCHLER *et al*. (1976), ERSCHLER *et al*. (1979) e ROUBELLAT & THOMAS (1988), que não otimizam um critério, preferindo fornecer ao operador um conjunto de soluções

compatíveis com as restrições a satisfazer, deixando-lhe a tarefa de selecionar a solução que lhe pareça a melhor. Um prazo de conclusão ou data de fim de fabricação mais tarde para os produtos *(due date)* é considerado nos métodos que procuram minimizar os atrasos de fabricação e também pelos métodos de NEPOMIASTCHY (1978), ERSCHLER *et al.* (1976), ERSCHLER *et al.* (1979), ROUBELLAT & THOMAS (1988), FERREIRA RIBEIRO & PRADIN (1991), FERREIRA RIBEIRO & RIBEIRO (1993), e FERREIRA RIBEIRO *et al*. (1993) nos quais se procura respeitar os prazos de entrega impostos aos pedidos. A atribuição de uma operação a uma máquina não é previamente fixada nos trabalhos de ERSCHLER *et al.* (1976), NEPOMIASTCHY (1978), ERSCHLER *et al.* (1979), ROUBELLAT & THOMAS (1988), MEGUELATI (1988), NASR & ELSAYED (1990), FERREIRA RIBEIRO & PRADIN (1991), FERREIRA RIBEIRO & RIBEIRO (1993), FERREIRA RIBEIRO *et al*. (1993). Nestes métodos, conhece-se um conjunto de máquinas aptas a executar cada operação e utilizam-se as diferentes alternativas possíveis para estabelecer a programação.

3. Formulação do Problema

Desde o final dos anos 50 vêm sendo desenvolvidos estudos para realizar a programação de operações com restrições disjuntivas por meio de modelos e técnicas de resolução propostos pela programação inteira. Entre os modelos apresentados, MUTH & THOMPSON (1963) citam: WAGNER (1959), BOWMAN (1959) e MANNE (1960), que utilizam o método dos planos de corte (SALKIN, 1975) para resolução da formulação considerada. BALAS (1969) aplica o algoritmo de resolução implícita (SALKIN, 1975) para resolver o modelo proposto por MANNE (1960). Este modelo apresenta um número de variáveis e restrições menor que o dos outros dois modelos propostos anteriormente. No início dos anos 70, a resolução dos problemas de programação com restrições disjuntivas sofreu um novo impulso com os trabalhos desenvolvidos na área da teoria dos grafos por ROY (1970) e retomados por GONDRAN (1974), GONDRAN & DOSTATNI (1977), GONDRAN & MINOUX (1985), CARLIER (1975, 1978, 1984), CARLIER & CHRÉTIENNE (1982), CARLIER & PINSON (1989).

3.1 Programação Inteira

As variáveis binárias (zero ou um) utilizadas nos modelos de WAGNER (1959), BOWMAN (1959) e MANNE (1960), citados por MUTH & THOMPSON (1963) são:

- 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:

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

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

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

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

$t_{ij} - t_{ik} \leq d_{ij} + M (1 - x_{ijk})$ e
$t_{ik} - t_{ij} \leq d_{ik} + M\, x_{ijk}$ (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 $x_{ijk} \in \{0, 1\}$ para todo i, j, k.

3.2 Teoria dos Grafos

Um grafo conjuntivo (ou seja, somente com restrições do tipo e) ou simplesmente um grafo G = (X, U) é conjunto finito não vazio X e um conjunto U de pares de elementos de X (BERGE, 1970), GONDRAN & MINOUX (1985), SAKAROVITCH (1984). Um grafo disjuntivo (ou seja, com restrições do tipo ou) é um grafo Ω = (G, D), no qual 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 restrição disjuntiva entre duas operações i e j é uma restrição do tipo ou e deve ser arbitrada de maneira a eliminar as disjunções, transformando Ω em um grafo conjuntivo: ou i é executada antes de j e insere-se, então, no grafo Ω 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 Ω um arco com origem em j e destino em i, de comprimento igual à duração da operação j.

Uma programação sobre o grafo disjuntivo Ω pode ser definida como um sistema de potenciais T = {t[i] | i ∈ X} tal que:
a) ∀ (i,j) ∈ U: t[i] - t[j] ≥ d_i, e **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 da operação j. À arbitragem ***A***, associa-se o grafo conjuntivo G(***A***) = (X, U∪***A***), denotado por G***A***. Uma arbitragem é dita completa se todas as restrições disjuntivas foram arbitradas. Uma arbitragem é dita compatível se o grafo conjuntivo associado à eleção ***A*** não contém circuito absorvente. ma solução para o problema é uma arbitragem completa e compatível. A duração da solução obtida é o valor do caminho crítico do grafo conjuntivo associado.

O conjunto de operações executadas sobre uma máquina, forma uma clique de disjunção: Ω = (X, U∪D) designa o grafo disjuntivo associado à fabricação e $\boldsymbol{A_O}$ é uma arbitragem completa e compatível. Assim, $\boldsymbol{A_O}$ é uma solução para o problema de programação representado por Ω; $\boldsymbol{L_O}$ corresponde à matriz de elemento genérico **lo**(i,j), em que **lo**(i,j) é o caminho de valor máximo indo de i a j no grafo G$\boldsymbol{A_O}$ = (X, U ∪ $\boldsymbol{A_O}$). Em outras palavras, uma clique de disjunção *C*, é o conjunto de operações tal que duas quaisquer entre elas estejam em disjunção. A duração D(*C*) da clique *C* é a soma das durações das tarefas que compõem a clique.

Ao problema de programação de operações em um sistema de produção

associa-se um grafo disjuntivo, em que os nós são as operações, acrescidos de duas operações fictícias: operação inicial (nó 1) e final (nó n). As restrições conjuntivas resultam do fato de que a passagem de uma peça sobre as diferentes máquinas se faz em uma ordem imposta. As restrições disjuntivas advêm do fato de que duas peças não podem utilizar a mesma máquina ao mesmo tempo.

Para se construir uma programação, é necessário arbitrar as restrições disjuntivas, isto é, escolher para cada disjunção [i,j] se i passará antes ou depois de j sobre a máquina correspondente, de maneira a determinar a seqüência de produção em cada uma das máquinas.

4. Método de Resolução

Na maior parte dos problemas de programação de operações encontra-se presente um grande número de restrições disjuntivas. Os problemas ligados a essas restrições disjuntivas são de natureza combinatória: com efeito, 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, desde que m ultrapasse dois dígitos (GONDRAN, 1974). Assim, os critérios de escolha nessas explorações são fundamentais.

O método desenvolvido neste artigo para a resolução do problema de programação de operações é dividido em duas partes: a primeira parte consiste na aplicação de propriedades teóricas para as soluções de um problema de programação com restrições disjuntivas propostas por CARLIER (1978, 1984). A segunda parte utiliza o método de penalidades descrito em GONDRAN & MINOUX (1985).

4.1 Ordem Total Associada a uma Arbitragem Compatível

Em um problema de programação de operações com restrições disjuntivas, dispõe-se de m máquinas para fabricar p peças. 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 de passar sobre a máquina 1, depois sobre a máquina 1 antes de passar sobre a máquina 3, etc. Uma programação w é uma m-upla (O_1, O_2, ..., O_m), na qual O_q designa a ordem de passagem das peças sobre a máquina q. Busca-se determinar a programação de duração mínima.

Relação de Ordem ρ

Escreve-se iρj se existe em G_{A_0} um caminho de i a j, de valor superior ou igual a 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 e k três nós pertencentes ao grafo G_{A_0};

   $i\rho j \Rightarrow l_o(x_i,x_j) \geq d[i]$; $j\rho k \Rightarrow$

   $\Rightarrow l_o(x_j,x_k) \geq d[j]$;

   por definição tem-se:

   $l_o(x_i,x_k) \geq l_o(x_i,x_j) + l_o(x_j,x_k) \geq$

   $\geq d[i] + d[j] \Rightarrow l_o(x_i,x_k) \geq d[i] \Rightarrow i\rho k$;

   portanto é transitiva. ■

Conseqüência: A clique tem um elemento menor **ê**, e um elemento maior **$**. **ê** e **$** são denominados entrada e saída da clique *C*,

respectivamente. Assim, a idéia fundamental consiste em encontrar para cada clique o elemento entrada (**ê**) e o elemento saída (**$**), prosseguindo desta forma até que a cardinalidade da clique seja igual a zero ou um.

Teorema de Existência

Existe em G_{A_0} um caminho, de 1 a n, de valor superior ou igual a: $l_o(1,\hat{e}) + D(C) + l_o(\$,n) - d[\$]$.

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 $x_c \equiv \$$. 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:

$$d[x_1] + d[x_2] + \ldots + d[x_c] = D(C) \Rightarrow$$
$$\Rightarrow d[x_1] + d[x_2] + \ldots + d[x_{c-1}] =$$
$$= D(C) - d[x_c] \text{ (onde } d[x_c] = d[\$]) \Rightarrow$$
$$\Rightarrow \sum_{x_i \in C - \{x_c\}} d[x_i] = D(C) - d[\$]$$

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

De onde resulta imediatamente que:

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

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 $x_c = \$$ a n.

4.2 Propriedades das Soluções de um Problema de Programação Disjuntiva

O estudo das propriedades propostas por CARLIER (1978, 1984) comuns às soluções de duração inferior a *f*, sendo *f* a duração da melhor solução conhecida (obtida, por exemplo, heuristicamente) é realizado abaixo.

Seja $\mathcal{E}_1$, o conjunto dos elementos ê tais que existe uma solução de duração inferior a *f* tendo ê como entrada da clique *C*. Seja $\mathcal{S}_1$, o conjunto dos elementos $ tal que existe uma solução de duração inferior a *f* tendo $ como saída da clique *C*. Seja $\mathcal{E}$ um conjunto incluso em *C* contendo $\mathcal{E}_1$; $\mathcal{S}$ um conjunto incluso em *C* contendo $\mathcal{S}_1$; e sejam i e j elementos de *C*. Seja Ψ uma solução; $\hat{e} \in C$ (resp. $\$ \in C$) é chamado entrada (resp. saída) da clique *C* se ê (resp. $) é sequenciado em Ψ antes (resp. depois) de todas as operações de *C*.

Teorema para Avaliação da Duração de Programação

A expressão:

$$\min_{e \in \mathcal{E}} (l(1,e)) + D(C) + \min_{e \in \mathcal{S}} (l(s,n) - d[s])$$

é uma avaliação por falta *(default)* da duração ótima.

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

1) $\hat{e} \in \mathcal{E}_1$ e $\mathcal{E}_1 \subset \mathcal{E} \Rightarrow \hat{e} \in \mathcal{E}$; assim, para $e \in \mathcal{E}$, $l_o(1,\hat{e}) \geq l_o(1,e)$; ou $l_o(1,e) \geq l(1,e)$
$\Rightarrow \min_{e \in \mathcal{E}} l_o(1,e) \geq \min_{e \in \mathcal{E}} l(1,e)$; por transitividade tem-se: $l_o(1,\hat{e}) \geq \min_{e \in \mathcal{E}} l(1,e)$; (1)

2) $\$ \in \mathcal{S}_1$ e $\mathcal{S}_1 \subset \mathcal{S} \Rightarrow \$ \in \mathcal{S}$; assim $l_o(\$,n) - d[\$] \geq \min_{s \in \mathcal{S}} (l_o(s,n) - d[s])$

ou

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

por transitividade tem-se:

$$l_o(\$,n) - d[\$] \geq \min_{s \in \mathcal{S}} (l(s,n) - d[s]); \qquad (2)$$

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

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

Determinação da Entrada e da Saída de uma Clique de Disjunção

Quanto mais próximos de 1 estiverem os cardinais de $\mathcal{E}$ e $\mathcal{S}$, melhor será a avaliação do teorema. A proposição abaixo permite retirar de $\mathcal{E}$ e $\mathcal{S}$ alguns nós; inicialmente considera-se $\mathcal{E} = \mathcal{S} = C$.

Proposição 1: Sejam as seguintes condições:

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

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

Se (1) ocorre então $i \notin \mathcal{E}_1$. Se (2) ocorre então $j \notin \mathcal{S}_1$.

Prova: Basta supor que (1) (resp. (2)) é satisfeita e que $i \in \mathcal{E}_1$ (resp. $j \in \mathcal{S}_1$). 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. ■

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 $k \in C$. Introduz-se a condição: $H(C-\{k\}) + d[k] > f$ (3)

Lema 2: Se (3) for satisfeita, então, em qualquer solução Ψ, a operação k é sequenciada 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:

$$l_0(1,i_1) + \sum_{i_k \in C-\{i_h\}} l_0(i_k, i_{k+1}) + l_0(i_h,n) > l(1,i_1) + \sum_{i \in C} d[i] + l(i_h,n) - d[i_h] \geq \min_{i \in C-\{k\}} (l(1,i)) + \sum_{i \in C-\{k\}} d[i] + d[k] + \min_{i \in C-\{k\}} (l(i,n) - d[i]) =$$

$= H(C-\{k\}) + d[k] \Rightarrow f > H(C-\{k\}) + d[k]$, o que contradiz (3). ■

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 $k \notin \mathcal{E}_1$. Assim se (3) é satisfeita, k só poderá ser saída. ■

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 $k \notin \mathcal{S}_1$. Assim se (3) é satisfeita, k só poderá ser entrada. ■

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 dos elementos 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 $\mathcal{E}$ ou de $\mathcal{S}$ é igual a 1.

Proposição 4: (1) Se e é a entrada da clique *C*, então: $l(e,n) \geq D(C) + \min_{s \in \mathcal{S}} (l(s,n) -$

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

Proposição 5: (1) Se $i \notin \mathcal{E}$ ($i \in C$), então $l(1,i) \geq \min_{e \in \mathcal{E}} (l(1,e) + d[e])$; (2) Se $j \notin \mathcal{S}$ ($j \in C$), então $l(j,n) \geq d[j] + \min_{s \in \mathcal{S}} l(s,n)$.

Prova: No grafo G, i é um descendente da entrada e, j um antecedente da saída. ■

Arbitragens Triviais

A proposição (6), abaixo, é aplicada somente quando o cardinal de $\mathcal{E}$ ou $\mathcal{S}$ 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) ($k \in C-\{e\}$) serão selecionados em qualquer solução. (2) Se *s* é a saída da clique *C*, então os arcos disjuntivos (k,*s*) ($k \in C-\{s\}$) serão selecionados em qualquer solução. (3) Se $\mathcal{E}_1 = \varnothing$ ou $\mathcal{S}_1 = \varnothing$, o problema não tem solução de duração inferior a f.

Prova: Pela definição de $\mathcal{E}_1$ e $\mathcal{S}_1$. ■

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) $l(1,j) + d[j] + l(1,n) \leq f$, 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). ■

4.3 Regra de Separação e Penalidades

Este método consiste na descrição sistemática do conjunto das soluções na forma de uma arborescência, percorrida sem que nenhum nó seja aberto. A noção de avaliação por falta *(default)* permite a exploração implícita de ramos inteiros da arborescência sem que seja examinado explicitamente cada um dos nós. Tal método de exploração adapta-se aos dados de cada problema específico. Portanto, ainda que o pior caso possa requerer um tempo de cálculo que cresça exponencialmente, consegue-se melhorar notavelmente, em média, o comportamento do algoritmo (GONDRAN & MINOUX, 1985). Três noções são fundamentais na exploração: 1ª) a noção de separação, que é a base do método; 2ª) a noção de avaliação por falta *(default)* que permite afirmar que não existem soluções melhores nos subconjuntos com avaliação por falta maior que a conhecida; 3ª) o critério de separação (noção de penalidade) que permite definir sobre qual variável deverá ser feita a separação.

Considere-se o seguinte problema: Min z = f(x), com $x \in W$, em que f(x) é a duração da programação das operações com arbitragem x, e W é o conjunto das arbitragens completas e compatíveis (isto é, o conjunto das soluções factíveis). Mediante um procedimento por separação e avaliação pode-se obter: uma solução ótima, todas as soluções ótimas, uma solução ε-ótima, todas as soluções ε-ótimas ou uma boa solução.

O primeiro princípio de um procedimento de arborescência por separação e avaliação consiste em explorar o conjunto das soluções factíveis W em subconjuntos cada vez menores, isolando-se assim a solução ótima (ou ε-ótima) em um desses subconjuntos. Para representar esta exploração, constrói-se uma arborescência cujo nó inicial corresponde ao conjunto W das soluções factíveis e os outros nós correspondem a subconjuntos de W. Para definir mais precisamente o procedimento de exploração, deve-se definir uma regra de separação (que permitirá construir a arborescência de exploração), e uma regra para se deslocar na arborescência.

O segundo princípio de um procedimento de exploração por separação e avaliação consiste em se determinar para cada

subconjunto $V \subset W$ um minorante (o maior possível) de $f(x)$ sobre V. Este minorante pode ser considerado como uma avaliação por falta de $f(x)$ sobre V, denotado por av(V). Tem-se $av(V) \leq f(x)$, $\forall x \in V$. O interesse principal desta avaliação por falta é o seguinte: se é conhecida uma solução factível para o problema, ϖ ($\varpi \in W$) e se: $av(V) \geq f(\varpi)$ (resp. $av(V) \geq f(\varpi) - \varepsilon$) não existe, então, solução melhor (resp. ε-melhor) no subconjunto V. Isto é evidente, uma vez que $\forall x \in V$, tem-se: $f(x) \geq av(V) \geq f(\varpi)$ (resp. $f(x) \geq f(\varpi) - \varepsilon$). Na prática é preciso calcular rapidamente uma "boa" avaliação por falta de $f(x)$ sobre todos os subconjuntos de V e obter rapidamente uma "boa" solução factível de$f(x)$ sobre W.

Um subconjunto V pode ser separado em mais subconjuntos V_1, V_2, ..., V_p tais que $V=V_1 \cup V_2 \cup ... \cup V_p$. Separa-se, geralmente, o subconjunto V em dois subconjuntos disjuntos ($V = V_1 \cup V_2$, $V_1 \cap V_2 = \varnothing$). Representam-se estes subconjuntos pelos nós de uma arborescência. Nesta arborescência, V será o "pai" de V_1 e V_2. Tenta-se, então, separar V em dois subconjuntos: um com grande probabilidade de conter a solução ótima, e outro com baixa probabilidade de conter "boas" soluções. Esta separação é freqüentemente realizada em um arco disjuntivo [i,j].

Assim, considera-se um par disjuntivo [i,j] 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:
$H^\circ = H \cap \{(i,j)\}$ 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°, $av(H^\circ) = av_1(H) + r(i,j)$, 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 $av_1(H) + \theta(i,j)$ é uma avaliação de H, em que $\theta(i,j) = \min(r(i,j), r(j,i))$. 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 $\gamma(i,j) = | r(i,j) - r(j,i) |$. Esta quantidade $\gamma(i,j)$ é a penalidade da função objetivo, decorrente de não se arbitrar [i,j] no sentido correspondente a $\theta(i,j) = \min(r(i,j), r(j,i))$; em outras palavras, se $\theta(i,j) = \min(r(i,j), r(j,i)) = r(i,j)$ e escolhe-se arbitrar no sentido j para i, a duração da programação será aumentada de $\gamma(i,j)$. Denomina-se $\gamma(i,j)$ 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 $\gamma(i,j) / [i,j] \in K_2$, 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) $\theta(i,j) = \min(r(i,j), r(j,i))$ representa a penalidade sobre $av_1(H)$ quando se acrescenta a arbitragem, sobre a disjunção correspondente [i,j];

3ª) $\max \theta(i,j)$ ($[i,j] \in K_2$) representa a penalidade sobre $av_1(H)$ quando se

arbitra [i,j] $\in K_2$ correspondente ao min(r(i,j),r(j,i));

4ª) $\gamma(i,j)$ 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 programação I, que utiliza tais penalidades é apresentado abaixo.

algoritmo programação I
A_k = conjunto das operações que antecedem a operação k
S_k = conjunto das operações que 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])
$\theta(i,j)$:= min(r(i,j), r(j,i))
$\gamma(i,j)$:= | 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 $\gamma(i,j)$ máximo e, em caso de igualdade, escolher aquela que tiver $\theta(i,j)$ 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 programação I

4.4 Método Geral

O método geral de resolução do problema de programação de operações com restrições disjuntivas, proposto neste artigo, é composto de duas etapas: 1ª) utilizam-se, em primeira instância, as proposições teóricas de CARLIER (1978, 1984) para arbitrar o maior número possível de disjunções no grafo associado ao problema; 2ª) em seguida, implementa-se o cálculo das penalidades de GONDRAN & MINOUX (1985), descrito no algoritmo programação I, para resolução das disjunções que não puderam ser arbitradas por meio das proposições.

O algoritmo programação II, apresentado abaixo, resume os passos fundamentais do método geral. Na implementação computacional do algoritmo programação II, efetuamos 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.

algoritmo programação II
A_k = conjunto das operações que antecedem a operação k

S_k = conjunto das operações que sucedem a operação k
r = índice ou número da máquina
C_r = clique de disjunção da máquina r
$\mathcal{E}_r$ = conjunto dos nós, pertencentes à C_r que podem ser entrada
$\mathcal{S}_r$ = conjunto dos nós, pertencentes à C_r que podem ser saída
L1[k] = comprimento do caminho máximo indo do nó 1 ao nó k
Ln[k] = comprimento do caminho máximo indo do nó k ao nó n
para toda C_r **fazer**
$C_r = \mathcal{E}_r = \mathcal{S}_r$
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 $\mathcal{E}_r$ e $\mathcal{S}_r$ 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** aplica-se o algoritmo programação I (GONDRAN & MINOUX, 1985)
árbitro:= *true*
fim se
fim enquanto
fim algoritmo programação II

5. Comparação dos Algoritmos

Os resultados obtidos com os programas computacionais correspondentes aos algoritmos programação I (cálculo das penalidades) e II (cálculo das proposições e das penalidades) sobre exemplos da literatura, são apresentados na Tabela 1. Um estudo comparativo de desempenho, entre o algoritmo programação II e um método exato, é fornecido na Tabela 2. Os programas computacionais programação I e II foram escritos em linguagem Turbo-Pascal e estão implantados em um microcomputador IBM-PC compatível. A notação utilizada nas Tabelas 1 e 2 é a seguinte: m = número de máquinas,
p = número de peças, nbnós = número total de nós do grafo, nbdis = número de disjunções no grafo disjuntivo inicial, f = duração da programação, τ = tempo de cálculo (cpu) em segundos, em um microcomputador IBM-PC 486, 75 MHz, 8 MBRAM, * = tempo de cálculo inferior a um segundo.

A Tabela 2 mostra a duração da programação das operações e os respectivos

tempos de cálculo (cpu), em segundos, para o programa correspondente ao algoritmo programação II e para o programa desenvolvido por CARLIER & PINSON (1989), com base no método de separação e avaliação, para determinação da solução ótima. Deve-se observar que CARLIER & 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. Uma alternativa interessante para melhorar o resultado de exemplos como este, seria realizar uma busca na vizinhança da solução encontrada.

Tabela 1: Resultados Obtidos

					Algoritmo I		**Algoritmo II**	
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	6
8	11	5	57	275	7534	2	7038	6
9	12	5	62	330	8848	3	7725	9
10	14	4	58	364	8736	2	8247	10
11	7	7	51	147	6829	*	6749	2
12	6	6	38	90	57	*	55	1
13	10	10	102	450	1040	5	1040	19
14	20	5	102	950	1323	19	1323	62

Tabela 2: Estudo Comparativo

Exemplo	**Algoritmo II**		**Solução Ótima**	
	f	τ	f	τ
8	7038	6	7038	41
9	7725	9	7312	14
10	8247	10	8003	64
11	6749	2	6558	53
12	55	1	55	1
13	1040	19	930	17985
14	1323	62	1165	1448

6. Conclusão e Comentários

A despeito da atenção considerável dada em periódicos de pesquisa operacional aos modelos de programação inteira para o tratamento de problemas de sequenciamento, programação da produção e rotas, estes têm apresentado uma utilização bastante limitada. Uma razão para isso é que tais modelos freqüentemente ignoram algumas variáveis importantes associadas aos ambientes reais de programação de operações. A outra razão é devida ao tempo computacional demasiadamente alto exigido para a resolução destes modelos e apresentação de uma solução. O tratamento do problema de programação de operações com restrições disjuntivas via teoria dos grafos combinada com procedimentos heurísticos é uma alternativa de solução cuja engenhosidade tem demonstrado eficiência e, assim, pode constituir uma alternativa para a implementação de pacotes computacionais com o objetivo de utilizá-los em tempo real nas indústrias.

Outras técnicas podem contribuir para a continuidade e o desenvolvimento deste trabalho, notadamente a introdução da decomposição espacial e/ou temporal para o tratamento de problemas de porte enorme, na linha dos trabalhos descritos em YAMAMOTO (1977), PORTMANN (1988), MEGUELATI (1988) e FERREIRA RIBEIRO & PRADIN (1991), integração de um sistema especialista para a tomada de decisão quando da resolução de disjunções críticas, introdução da técnica de busca em vizinhança e a utilização da computação paralela, com o objetivo de diminuir o tempo de cálculo e aumentar o número de possibilidades de soluções a serem exploradas, no caso dos problemas reais da indústria. O trabalho de OLIVEIRA *et al.* (1994) é pioneiro nesta linha de pesquisa e os testes realizados, bem como os resultados obtidos nessa pesquisa constituem uma perspectiva animadora.

Referências Bibliográficas:

ASHOUR, S. & HIREMATH, S.R.: "A branch-and-bound approach to the job shop scheduling problem", *Int. J. Prod. Res*., 11, 1, pp. 47-58, 1973.

BAKER, K.R.: *Introduction to sequencing and scheduling*, John Wiley, 1974.

BAKER, K.R. & SCHRAGE, L.E.: "Finding an optimal sequence by dynamic programming: an extension to precedence-related tasks", *Oper. Res*., 26, 1, pp. 111-120, 1978.

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.: "Un domaine très ouvert: 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.

DIDRI, N. & PROTH, J.M.: "Ordonnancement des tâches: une méthode basée sur la TG", *II CISP*, Paris, França, pp. 61-74, 1987.

DUPONT, L.: *Algorithmes et Ordonnancements*, Tese de Doutorado Un. J. Fourier, Grenoble, França, 1986.

ERSCHLER, J.; ROUBELLAT, F.; VERNHES, J.P.: "A decision making process for the real time control of a production unit", *Int. J. of Prod. Res*., 14, 2, pp. 275-284, 1976.

ERSCHLER, J.; FONTAN, G.; ROUBELLAT, F.: "Potentiels sur un graphe non conjonctif et analyse d'un problème d'ordonnancement à moyens limités", *RAIRO*, 13, 4, pp. 363-378, 1979.

FERREIRA RIBEIRO, J.F. & PRADIN, B.: "TEGO: un logiciel pour l'ordonnancement d'ateliers de production", *GSI'3*, Tours, França, pp. 888-896, 1991.

FERREIRA RIBEIRO, J.F. & RIBEIRO, C.M.: "Um método para sequenciamento baseado na decomposição dos sistemas de manufatura", *XXV SBPO*, Campinas, SP, pp. 346-350, 1993.

FERREIRA RIBEIRO, J.F.; RIBEIRO, C.M.; PRADIN, B.: "Un algorithme basé sur les îlots de fabrication pour l'ordonnancement", *GSI'4*, Marseille, França, pp. 219-224, 1993.

GAREY, M.R. & JOHNSON, D.S.: *Computers and intractability: a guide to the theory of np-completeness*, Freeman, 1979.

GONDRAN, M.: *Programmation combinatoire*, Bulletin EDF, série C, Mathématiques, Informatique, 2, pp. 45-66, França, 1974.

GONDRAN, M. & DOSTATNI, M.: *Le traitement des exclusives dans Plannec*, Bulletin EDF, série C, Mathématiques, Informatique, 1, pp. 69-78, 1977.

GONDRAN, M. & MINOUX, M.: *Graphes et algorithmes*, Eyrolles, 1985.

HUTCHISON, J. & CHANG, Y.L.: "Optimal nondelay job shop schedules", *Int. J. Prod. Res*., 28, 2, pp. 245-257, 1990.

JOHNSON, S.M.: "Optimal two and three stage production schedules with set up times included", *Naval Res. Logist Quart*., 1, pp. 61-68, 1954.

KAUFFMANN, A. & LABORDÈRE, A.H.: *Méthodes et modèles de la RO*, Dunod, 1974.

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

MEGUELATI, S.: *Méthodes de Classification pour la constitution d'îlots e l'ordonnancement*, Tese de Doutorado INSA, Toulouse, França, 1988.

MUTH, J.F. & THOMPSON, G.L.: *Industrial Scheduling*, Prentice Hall, 1963.

NASR, N. & ELSAYED, E.A.: "Job Shop scheduling with alternative machines", *Int. J. Prod. Res.*, 28, 9, pp.1595 - 1609, 1990.

NEMHAUSER, G.L. & WOLSEY, L.A.: *Integer and Combinatorial Optimization*, John Wiley, 1988.

NEPOMIASTCHY, P.: "Résolution d'un problème d'ordonnancement à ressources variables", *RAIRO*, 12, 3, p. 249-261, 1978.

NORONHA, A.B. & FERREIRA RIBEIRO, J.F.: "Um algoritmo branch-and-bound para o sequenciamento da produção", *XIV ENEGEP*, João Pessoa, PB, pp. 672-677, 1994a.

NORONHA, A.B. & FERREIRA RIBEIRO, J.F.: "Um método e um programa para o sequenciamento da produção", *XXVI SBPO*, Florianópolis, SC, pp. 06-11, 1994b.

NORONHA, A.B. & FERREIRA RIBEIRO, J.F.: "Um método para o sequenciamento de operações baseado em busca arborescente e proposições", *XV CILAMCE*, Belo Horizonte, NG, pp. 1452-1459, 1994c.

OLIVEIRA, M.M.B.; FERREIRA RIBEIRO, J.F.; RIBEIRO, C.M.: "Sequenciamento de operações baseado na decomposição de sistemas de manufatura e uso de processamento paralelo", *X CBA / VI CLACA*, Rio de Janeiro, RJ, pp. 133-138, 1994.

PORTMANN, M.C.: "Méthodes de décomposition spatiales et temporelles en ordonnancement de la production", *RAIRO*, 22, pp. 439-451, 1988.

PROTH, J.M.: "GT in production management: a tool to simplify some scheduling problems", *IEEE Int. Conf. on Robot. & Aut*., pp. 1641-1644, 1986.

ROUBELLAT, F. & THOMAS, V.: "Une méthode et un logiciel pour l'ordonnancement en temps réel d'ateliers", *RAIRO*, 22, pp. 419-438, 1988.

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.

YAMAMOTO, M.: "An approximative solution of machine scheduling problems by decomposition method", *Int. J. Prod. Res*., 15, 6, pp. 599-608, 1977.

YAMAMOTO, M. & NOF, S.Y.: "Scheduling/ rescheduling in the manufacturing operating system environment", *Int. J. Prod. Res*., 23, 4, pp. 705-722, 1985.

SCHEDULING WITH DISJUNCTIVE CONSTRAINTS

Abstract

The job shop scheduling problem consists of determining a sequence of jobs to be processed on each of the available machines such that the schedule time is minimized. The parts are processed in accordance with a prespecified technological ordering and the required processing times of the operations pertaining to each job are known. In this paper we study the job shop scheduling problem in two ways, namely, integer programming and graph theory. A computational program based on graph theory was developed and tested. This program allows us to solve efficiently several examples, despite the non-polynomial nature of the problem studied.

Key-words: scheduling, job shop, integer programming, graph theory.