v.3, n.3, p. 228-248, dez. 1996

UM SISTEMA DE CONTROLE DA PRODUÇÃO PARA A MANUFATURA CELULAR. PARTE II: UM SISTEMA DE EMISSÃO DE ORDENS E PROGRAMAÇÃO DE OPERAÇÕES

Flávio César F. Fernandes
Prof. Doutor do Departamento de Engenharia de Produção,
Universidade Federal de São Carlos (UFSCar)
Fax: 016-274.8240.

Ricardo Ferrari Pacheco
Bacharel em Ciências da Computação e Mestre em
Engenharia de Produção pela UFSCar
Doutorando em Engenharia de Produção pela EP-USP

Resumo

Este artigo e o da Parte I (Sistema de Apoio à Decisão para a Elaboração do Programa Mestre de Produção) fazem parte de um trabalho que visa integrar todas as atividades de programação da produção dentro do contexto de um Sistema de Controle da Produção, concebido para operar na manufatura celular semi-repetitiva. Na Parte I explicamos o que entendemos por manufatura celular semi-repetitiva. O sistema proposto foi concebido e implementado no computador em 3 módulos (nível de produto final, de componentes e de operações). O primeiro foi tratado no artigo da Parte I, enquanto que os outros dois módulos são tratados neste artigo (Parte II). Em cada um deles existem contribuições relevantes para a literatura especializada.

Palavras-chave: controle da produção, programação da produção, emissão de ordens, programação de operações, manufatura celular.

1. Introdução

No artigo da Parte I (FERNANDES & TAHARA, 1996) mostramos a diferença entre Controle da Produção e Planejamento da Produção e explicamos o que entendemos por manufatura celular semi-repetitiva (que é o ambiente ao qual se destina o Sistema de Controle da Produção (SCP) que propomos). Para tanto, apresentamos nossa classificação de sistemas de produção, bem como da

manufatura celular, e apresentamos o nível 1 de decisão do SCP, ou seja, o Programa Mestre de Produção (PMP), que trata dos produtos acabados que são entregues aos clientes. É importante lembrarmos que, de forma resumida, definimos a manufatura celular semi-repetitiva como aquela em que: há uma porcentagem significativa de produtos repetitivos e de produtos não repetitivos, a produção é sob encomenda de produtos semi-padronizados e o padrão de fluxo nas células é *job-shop*.

Neste artigo são tratados os níveis 2 e 3 de decisão do SCP proposto, ou seja, o nível de componentes (emissão de ordens) e o nível de operações (programação de operações).

Na seção 2 apresentamos uma classificação dos sistemas de emissão de ordens (SEOs) e na seção 3 apresentamos nossa proposta de um SEO, o qual denominamos PBC Modificado. Na seção 4, apresentamos sucintamente o sistema de programação de operações proposto. Ele foi denominado de PROCEL. Na seção 5, temos o SEMPRO (sistema de emissão de ordens e programação de operações) que é a implementação em computador do PBC Modificado e do PROCEL. Na última seção apresentamos as conclusões deste artigo, bem como as considerações finais relativas ao sistema de controle da produção como um todo.

2. Emissão de Ordens e uma Classificação dos Sistemas de Emissão de Ordens

O processo de emissão de ordens tem como objetivo principal converter as necessidades colocadas no PMP em termos de produtos finais, para a forma de necessidades em termos de itens componentes (produzidos ou comprados). Vale ressaltar que é muito importante a escolha do sistema de emissão de ordens de produção (SEOP) empregado. Ela deve levar em conta principalmente o tipo de sistema de produção em que o SEOP vai operar. Em um sistema de produção repetitivo, que possui uma variedade cibernética pequena, um sistema simples como o Kanban é, em geral, altamente apropriado. Para um sistema de produção não-repetitivo não há atualmente uma alternativa melhor do que o MRP. Na situação intermediária, sistema de produção semi-repetitivo, é que a questão é crítica, e ainda não há uma resposta definitiva. O Kanban apresenta difícil aplicabilidade em ambientes semi-repetitivos, enquanto que o MRP apresenta uma complexidade além da exigida num ambiente de manufatura celular. Em uma pesquisa que orientamos (FUSSE, 1993), compreendendo 5 metalúrgicas do estado de São Paulo em que estavam implantados o MRP e células de manufatura, a principal conclusão foi que tais empresas estavam satisfeitas com as células de manufatura mas não com o desempenho do MRP nesse ambiente.

BURBIDGE (1983) classifica os Sistemas de Emissão de Ordens (SEOs) em três grupos:

(i) sistemas para fazer de acordo com o pedido;

(ii) sistemas de estoque controlado. São sistemas em que o nível de estoque de cada item é controlado de forma independente dos demais e de forma independente do PMP (plano de produção, na terminologia de Burbidge). O nível de estoque deve ser tal que, no momento em que a montagem precisar de um item particular, ele deverá estar disponível;

(iii) sistemas de fluxo controlado. São sistemas que determinam as necessidades de itens (em termos de tempo e quantidade) diretamente a partir do PMP.

Alterando essa classificação, propomos uma taxonomia dos SEOPs com cinco categorias, que se por um lado envolve muitas categorias para o número de SEOPs existentes, por outro lado ajuda a entender melhor tais sistemas e nos permite enquadrar o sistema Kanban, o que não é possível com a classificação acima. As categorias são:

(I) Sistema de Pedido Controlado. É usado nos sistemas de produção grande projeto e, eventualmente, no não repetitivo. Compreende o: (1) sistema da programação por contrato; e o (2) sistema da alocação de carga por encomenda;

(II) Sistema de Estoque Controlado que Puxa a Produção. Compõe-se do: (3) sistema de estoque mínimo;

(III) Sistema de Estoque Controlado que Empurra a Produção. Pertence a esta categoria o: (4) sistema de estoque-base;

(IV) Sistema de Fluxo Controlado que Empurra a Produção. Comporta os seguintes sistemas: (5) PBC (*Period Batch Control*= sistema do período padrão); (6) MRP; (7) SERVE (sistema de emissão de ordens do sistema de controle da produção OPT); (8) sistema dos lotes componentes; (9) sistema do lote padrão; (10) sistema do controle maxmin;

(V) Sistema de Fluxo Controlado que Puxa a Produção. Constitui-se do sistema: (11) Kanban. É um sistema híbrido, já que, com base no PMP, são requisitadas as peças que são usadas na montagem; assim, é um sistema de fluxo controlado; por outro lado, a solicitação de produção (ou fornecimento) de peças é feita com base no estoque controlado com produção puxada.

A grande maioria dos sistemas (1) a (11) está descrita em BURBIDGE (1983) e em ZACCARELLI (1987) sendo que os sistemas (4), (8) e (9) estão apresentados de maneira mais clara em ZACCARELLI (1987).

Nos sistemas que puxam a produção, a ordem de fabricação dos produtos segue a ordem contrária do seu processamento pelas máquinas, ou seja, a necessidade de se fabricar determinado produto irá gerar a solicitação de fabricação de seus subcomponentes ao centro de trabalho precedente. Para uma compreensão mais detalhada da diferença entre puxar e empurrar a produção sugerimos a leitura de LEE (1989).

3. O PBC Modificado

Vale a pena mencionar que para ZACCARELLI (1990) existem poucos sistemas cuja aplicação é cogitável atualmente, e entre eles estão os sistemas periódicos, um dos quais é obviamente o PBC.

Existem vários estudos que visam comparar o PBC e o MRP. Por exemplo, Zelenovic & Tesic (1988, *apud* YANG & JACOBS, 1992). O problema dessas comparações é que elas não dão destaque ao contexto em que as comparações foram feitas, nem ao fato de que as conclusões vão diferir em função do ambiente de manufatura estudado.

O PBC, criado pelo consultor inglês R.J. Gigli, é um sistema de emissão de ordens de fluxo controlado que empurra a produção. Gigli adaptou para a manufatura repetitiva sistemas semelhantes já existentes e usados

na produção em massa. O PBC é muito utilizado na Inglaterra. Em termos de planejamento ele puxa as várias etapas do processo. Seu esquema básico é o seguinte:

Etapa 0: recebe-se o Programa Mestre de Produção (PMP) definido para vários ciclos de igual tamanho;

Etapa 1: é feita a "explosão" do PMP para definir a quantidade de cada item que deve ser produzida para o ciclo em questão;

Etapa 2: atribuem-se tempos para, por exemplo, a

A) emissão das ordens mais a produção ou entrega de matérias-primas usadas no processamento;
B) processamento ou entrega de componentes;
C) montagem;
D) vendas.

Essa atribuição de tempos é repetida para cada um dos ciclos, obtendo-se um programa padrão (figura 1). Desta forma, tem-se um sistema de ciclo único (todos os itens têm emissão de ordens no mesmo ciclo) e de fase também única (visto que todos os itens são emitidos na mesma série de dias e com a mesma série de datas devidas para todos os itens). Além disso, é mais fácil planejar a seqüência de operações quando empregamos o PBC, do que quando empregamos qualquer outro SEOP.

No exemplo da figura 1 o ciclo é de duas semanas e as vendas futuras devem ser previstas com uma antecipação máxima de oito semanas (duração do programa padrão).

Quanto menor o horizonte da previsão de vendas, naturalmente mais confiável se torna a previsão. Outra importante vantagem de trabalhar com ciclos curtos é que o sistema se torna "mais flexível e pode rapidamente seguir as alterações na demanda do mercado, com um mínimo investimento em estoque" (BURBIDGE, 1975).

Outra observação importante é que quado definimos o ciclo como sendo de duas semanas, por exemplo, estamos admitindo que é possível executar as montagens em duas semanas, produzir e receber os itens em duas semanas, produzir e receber as matérias-primas em duas semanas.

Assim sendo, é claro que o ciclo não pode ser menor que o *lead-time* de nenhum dos componentes envolvidos. Portanto, caso o *lead-time* de entrega de alguns itens seja longo, para não ser necessário ampliar os prazos de todo o sistema, deve-se controlar esses itens por outro sistema que não o PBC, como por exemplo um contrato de fornecimento programado. KAKU & KRAJEWSKI (1995) propõem um modelo para examinar a escolha de tamanho do ciclo. A diminuição do ciclo no sistema PBC é desejável, caso seja possível, pois encurta os prazos de entrega e diminui o estoque em processo. O problema dessa diminuição do ciclo é que isso aumenta a proporção do tempo de preparação, o que leva à diminuição da capacidade produtiva total.

	Semana					
Vendas da semana	1-2	3-4	5-6	7-8	9-10	...
7-8	A	B	C	D		
9-10		A	B	C	D	
11-22			A	B	C	D
...				...	...	...

Figura 1: Programa padrão

Defendemos a utilização do sistema de emissão de ordens denominado PBC Modificado como a alternativa mais adequada ao ambiente da manufatura celular semi-repetitiva.

O módulo de Emissão de Ordens do sistema de controle de produção proposto, utiliza o método do PBC (*Period Batch Control*) Modificado, com período duplo de fabricação (figura 2) e um esquema de atribuição de prioridade às peças. A duração de cada período pode ser definida pelo usuário: uma semana, uma quinzena, etc.

período 1	período 2	período 3	período 4	período 5	período 6	período 7
obtenção Mat. PMP 1	Fabricação de Peças PMP 1	Fabricação de Peças PMP 1	Montagem			
	obtenção Mat. PMP 2	Fabricação de Peças PMP 1 ou 2	Fabricação de Peças PMP 1 ou 2	Montagem		
		obtenção Mat. PMP 3	Fabricação de Peças PMP 1 ou 2 ou 3	Fabricação de Peças PMP 1 ou 2 ou 3	Montagem	
			obtenção Mat. PMP 4	Fabricação de Peças PMP 2 ou 3 ou 4	Fabricação de Peças PMP 2 ou 3 ou 4	Montagem

Figura 2: O PBC Modificado

Admitindo que 1 período equivale a uma semana, notemos que na figura 2 a duração do programa padrão é de 4 semanas (1+2+1). Se estivéssemos usando o PBC, a duração seria de 6 semanas (2+2+2). É obvio que quanto menor a duração do programa padrão menores são os estoques em processo e mais rapidamente os clientes serão atendidos.

De forma resumida, podemos dizer que o objetivo do módulo de emissão de ordens é gerar o relatório de itens a serem produzidos em cada célula em determinado período, divididos em três categorias, a saber:

1) **Lista Expressa (E)**: lista dos itens que deviam ter sido fabricados no ciclo anterior, mas que por algum imprevisto qualquer, não foram concluídos. Entre estes motivos podem estar quebras de máquinas, greves, absenteismo, atrasos no fornecimento de matérias-prima, etc.
2) **Lista Gargalo (G)**: lista dos itens que devem ser fabricados neste ciclo de produção que se inicia e cujo *lead-time* de produção é maior que um período simples. Dessa forma, sua fabricação deve começar agora para terminar dentro do ciclo.
3) **Lista Normal (N)**: lista que contém as demais peças, isto é, as que deverão ser fabricadas no ciclo que se inicia, mas cujo *lead-time* não é maior que um período.

Embora não seja óbvio, devemos notar que sem esse esquema de atribuição de prioridades às peças, a proposta de trabalhar com período duplo de fabricação não funcionaria. Num mesmo período podemos

estar fabricando peças de até 3 PMPs (geralmente dois) e isso é acomodado exatamente pelo esquema de atribuição de prioridades proposto.

Os itens a serem produzidos no período duplo de fabricação foram obtidos do PMP aprovado para o ciclo. Depois de fazer a "explosão" da programação destes produtos na de seus itens componentes, totalizam-se as quantidades de itens componentes a produzir e estes são divididos nas listas G e N descritas acima, após serem incluídos na lista E os itens que não foram produzidos no ciclo anterior, devido a algum tipo de imprevisto. Obviamente os imprevistos são mais comuns na manufatura semi-repetitiva do que na repetitiva e, assim, o PBC Modificado deve fornecer, e de fato fornece, mecanismos para acomodar os imprevistos.

4. O PROCEL

PROCEL foi a denominação dada ao algoritmo aqui proposto para a programação de operações na manufatura semi-repetitiva.

Iniciando-se o módulo de programação de operações, devemos observar que as listas E, G, e N geradas pelo módulo de emissão de ordens para o ciclo que se inicia, serão alocadas às máquinas respeitando-se as categorias a que pertencem, ou seja, primeiramente serão alocadas as operações para os itens pertencentes à lista E, depois as correspondentes aos itens pertencentes à lista G, e depois as referentes aos itens da lista N.

Desta forma, obtém-se um mecanismo de *feedback* quanto ao andamento da produção, e novas emissões de ordens no período posterior são capazes de dar prioridade a lotes em atraso.

Dentro de cada lista, a seguinte seqüência de passos, sete ao todo, faz com que todas as operações sejam alocadas às máquinas:

Procel passo1)

O programa principal do módulo Procel consulta seqüencialmente os arquivos LISTAE, LISTAG e LISTAN, produzidos pelo PBC Modificado e, para cada uma dessas listas, faz a chamada dos passos seguintes do PROCEL.

Ao final da execução dos demais passos para as listas E, G e N, é acionada a rotina que apresenta a programação de operações que foi gerada.

Procel passo2)

Executa a comparação dos roteiros de fabricação (arquivo ROTFAB) para todas as pecas existentes em LISTAX (X podendo ser E, G ou N), com o objetivo de encontrar coincidências absolutas quanto às seqüências de máquinas. Peças com roteiros de fabricação coincidentes são anotadas como pertencentes à mesma subfamília.

É importante observar-se que, no caso de uma máquina eventualmente fazer parte de duas células ao mesmo tempo (ou seja, fazer parte de uma célula normal e de outra "virtual", responsável pela fabricação de um *black-sheep*), essa máquina deverá ser cadastrada no sistema duas vezes, uma para cada célula, e com códigos diferentes. As máquinas devem ser cadastradas indicando no seu código a que célula pertencem.

Procel passo3)

São carregadas no arquivo OP as peças de maior tempo de *setup* absoluto de cada subfamília em cada máquina. Cada uma delas será a primeira peça da subfamília a ser processada na máquina que está sendo considerada. Tempo de *setup* absoluto de

uma peça é o tempo de preparação da máquina para fabricar essa peça, dado que a máquina está em "vazio".

Definindo-se a peça de maior tempo de *setup* como sendo a primeira peça da subfamília a ser processada para cada máquina, os tempos ociosos que eventualmente surgirem na programação poderão ser usados para a preparação da máquina com vistas a receber a peça que requer mais tempo de preparação.

Procel passo4)

Neste passo, para cada máquina buscamos minimizar o tempo total de preparação ou, em outras palavras, a soma dos *setups* relativos entre as peças pertencentes àquela subfamília que serão executadas naquela máquina.

Como já definimos que a operação de maior tempo de *setup* absoluto é a primeira da seqüência, basta aplicarmos apenas uma vez por subfamília e por máquina o algoritmo de WHITE & WILSON (1979), que utiliza o método *branch-and-bound*. O algoritmo recai no conhecido problema do caixeiro viajante, sendo que neste caso a primeira peça já é a pré-definida pelo passo três. Traduzimos o código do caixeiro viajante para o Foxpro for Windows ® a partir do código em Pascal contido em SYSLO *et al.* (1983).

A seqüência obtida é anotada no arquivo OSUBFMAQ. Um esboço gráfico deste passo é dado pela figura 3.

É importante observar-se a diferença existente entre o que denominamos de *setup* absoluto e *setup* relativo. No passo 4 do Procel, são utilizadas as tabelas de *setup* relativo entre as peças de cada família (DIFSETUP), fabricadas em uma determinada máquina. Esse *setup*, denominado tempo de *setup* relativo, representa o tempo médio de preparação da máquina que, tendo acabado de produzir um lote de peças x, irá agora produzir um lote de y. Esse tempo não é o mesmo que o de *setup* absoluto, que denominamos em nosso trabalho simplesmente como *setup*.

O sistema possui uma tabela para o preenchimento dos *setups* relativos entre as peças pertencentes a cada subfamília, em cada máquina (tabela DIFSETUP). Num ambiente real, esses tempos deverão ser cronometrados, ou, caso isso não seja possível, devido à grande quantidade de combinações de peças existente numa subfamília, deverão ser estimados com base em algumas medições e levando em consideração a similaridade da preparação da máquina.

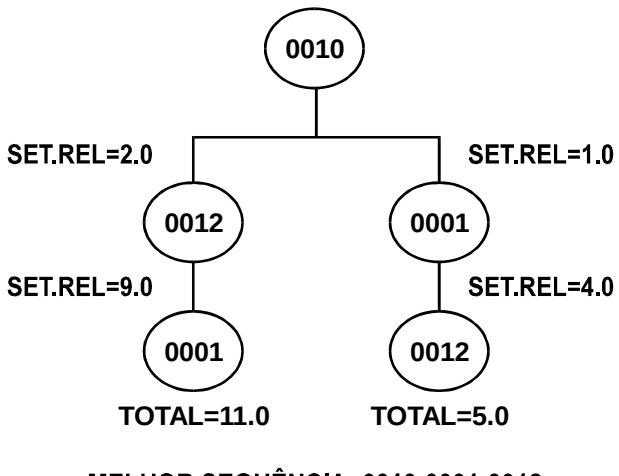

Figura 3: Encontrando a Melhor Seqüência de Operações das Peças Pertencentes a uma Subfamília, em Cada Máquina

Procel passo5)

No quinto passo do PROCEL, é calculado, para cada subfamília, o tempo total de processamento por máquina. Assim sendo, são somados o tempo de *setup* absoluto da primeira peça (a de maior *setup*) mais os tempos de *setup* relativos das demais peças, na ordem obtida pelo passo 4, mais os tempos de processamento unitários multiplicados pelas quantidades a serem produzidas de cada peça da subfamília.

Procel passo6)

Neste passo foi implementada uma solução heurística para a programação das tarefas às máquinas. Como já foi dito, toda a seqüência das operações pertencentes a uma mesma subfamília para uma determinada máquina converteu-se numa tarefa pelo passo 5. Em outras palavras, olhando apenas as operações que compõem cada uma das tarefas em cada máquina em que essas operações serão executadas, poderíamos afirmar que foi implementado um modelo *flow-shop* (visto que todas as tarefas têm a mesma seqüência de operações na máquina). Já entre as tarefas, temos obviamente um modelo *job-shop* que é composto das células, máquinas e tarefas.

Para obtermos a programação das tarefas no primeiro estágio do sistema (existe o segundo estágio, ou seja o passo 7, no qual se busca uma solução melhor do que a obtida no primeiro estágio), propomos uma adaptação do algoritmo heurístico geral para o problema do *job-shop* existente em BAKER (1974). A principal diferença entre o algoritmo original de Baker e a nossa proposta é que, no nosso caso, várias tarefas podem ser programadas em cada iteração, ao contrário do algoritmo original, na qual é programada uma tarefa por vez. Temos assim, um algoritmo mais rápido, que programa em cada iteração, tarefas para todas as máquinas.

Para o detalhamento do algoritmo proposto, criamos a idéia de listas de tarefas. Foram criadas as listas: Lista1, Coc, Anterior, Solução e Conflito.

Inicialmente alocam-se à Lista1 todas as tarefas a serem programadas. As primeiras operações (campos seqüência=1) de cada tarefa constituem-se inicialmente nas operações candidatas a serem programadas. Retiram-se estas operações de Lista1 para inclui-las em Coc (conjunto das operações candidatas). Para cada uma das operações candidatas, calcula-se o seu Ev, que é o maior valor entre a data do fim da operação precedente e a data da liberação da máquina a ser utilizada pela operação (no caso das primeiras operações, todas têm Ev=0). A operação com menor Ev para cada máquina será programada. Caso ocorram coincidências no valor do Ev para uma mesma máquina (chamaremos essas coincidências daqui por diante de conflitos), estas são armazenadas na lista de conflitos, para utilização pelo passo sete, e o desempate aqui é feito escolhendo-se a operação de maior *MWKR* (*Most Work Remaining*), ou seja, a operação com mais tempo restante para ser terminada. As operações programadas a cada interação são excluídas da lista de operações candidatas, e incluídas nas listas de Solução e de Anterior, que indicam quais foram as operações programadas no passo anterior. Novamente, iniciando mais uma iteração, observa-se quais as novas operações candidatas (são as operações subseqüentes, caso existam, às encontradas na lista de Anteriores, i.e., as que foram programadas no passo anterior) e incluem-se estas operações em Coc, juntamente com as

ainda não programadas que lá permaneceram. Novamente é calculado o Ev para essas operações e os menores são escolhidos, as operações são alocadas, as listas e os conflitos atualizados. Isso é feito até que todas as operações tenham sido programadas (ou seja, até que as listas Lista1 e Coc estejam vazias).

A lista Solução obtida é armazenada no arquivo OS e a lista de conflito é armazenada no arquivo Confli. A solução obtida é o Limitante Superior para o algoritmo do passo 7.

A figura 4 apresenta o relacionamento entre as listas de tarefas do Passo 6.

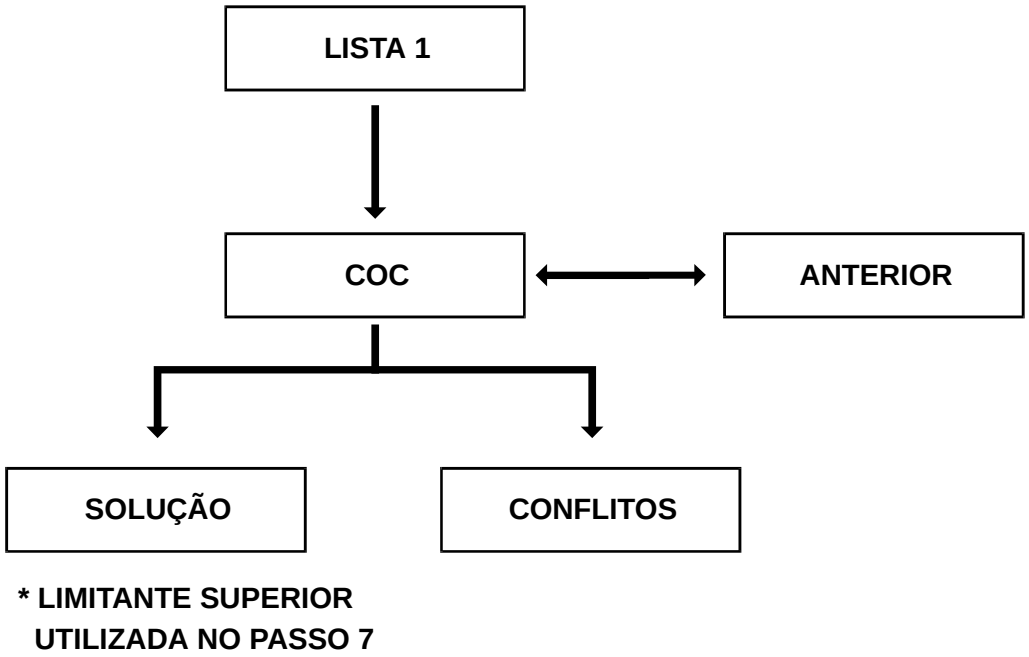

Figura 4: O Relacionamento entre as Listas de Tarefas do Passo 6

Como mostramos na figura 5, para a máquina MQ1, a tarefa de menor Ev é a OP23. Já para a máquina MQ3, ocorreu um empate no valor de Ev. Se imaginarmos que as tarefas OP33 e OP43 são as últimas tarefas da subfamília, os valores de MWKR corresponderão ao próprio tempo das tarefas (suponhamos que sejam 3 e 1 respectivamente). Se imaginarmos, em outro exemplo, a existência da tarefa OP44, com tempo de tarefa, digamos, igual a 2.5, então obteremos um valor diferente de MWKR para a tarefa OP43 (1+2.5= 3.5). Neste caso, a tarefa selecionada seria a OP43 e não a operação OP33.

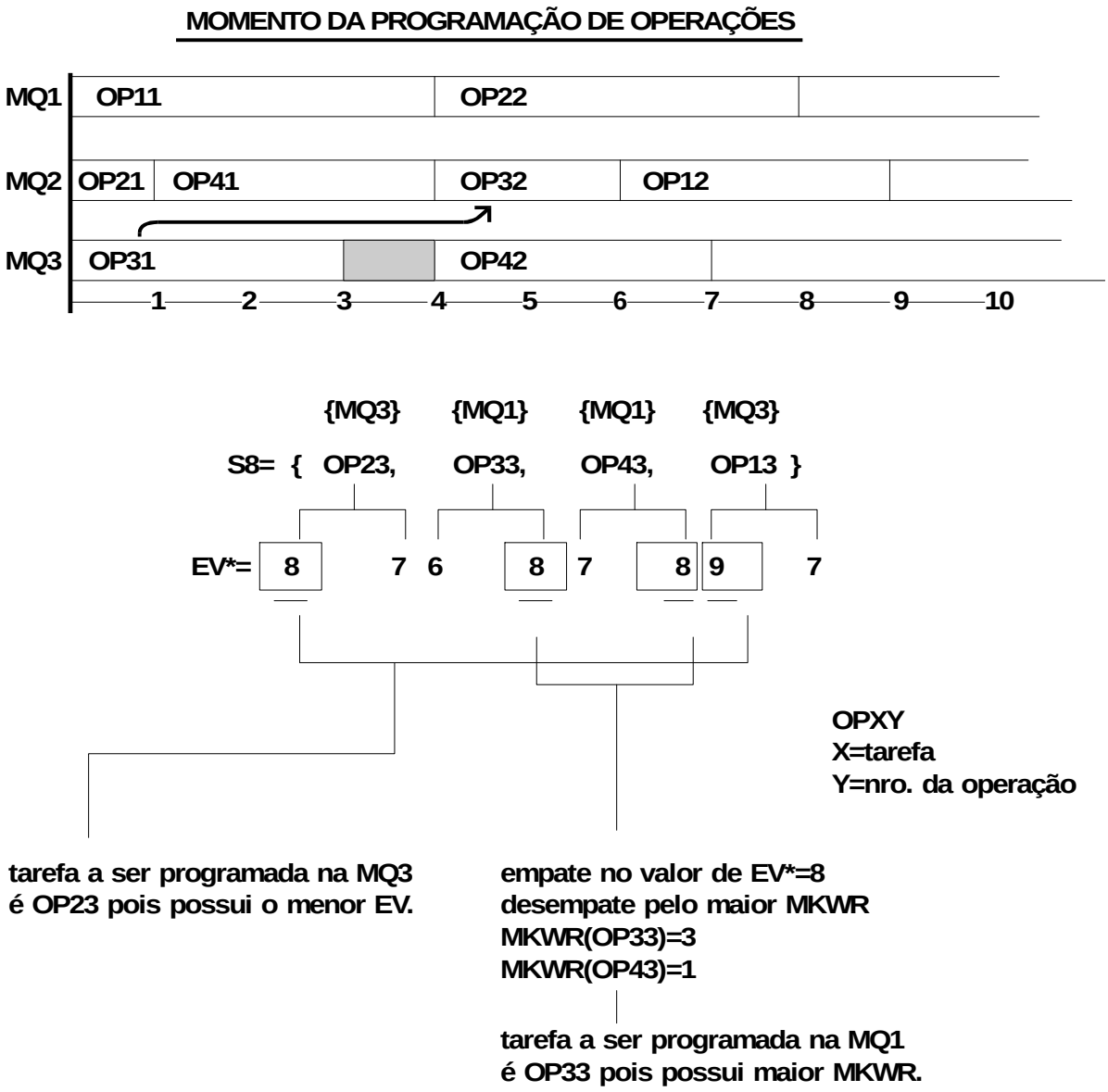

Figura 5: A Programação de Operações no Passo 6

Procel passo7)

A função do passo sete é tentar encontrar uma solução melhor que a do passo 6. Para tanto, elaboramos um algoritmo baseado na técnica de *branch-and-bound*, que toma a solução do passo 6 como solução inicial, ou *Upper-Bound* (UB).

Os conflitos são analisados de baixo para cima na árvore, enquanto ainda houver tempo disponível para a análise das possibilidades. Este tempo máximo de busca de uma solução melhor é fornecido pelo usuário.

Para cada nó-conflito, as operações subseqüentes são reprogramadas, supondo-se agora que esta operação do conflito, que não foi escolhida pelo passo 6, é agora a operação que foi selecionada para ser executada primeiramente no conflito, seguindo-se então as demais operações.

Desta forma, a operação que havia sido anteriormente escolhida como a primeira do conflito no passo seis, pelo desempate por meio do maior *MWKR*, passa a ser programada depois.

Esta reprogramação é feita sempre que o *lower-bound* deste nó-conflito é calculado como sendo menor que o UB. Neste caso, esta reprogramação passa a ter o valor do novo UB.

É importante observar ainda que o usuário, de acordo com a situação da fábrica, tem duas possíveis escolhas de critérios de desempenho:

1- minimizar o tempo médio de permanência das tarefas no sistema (F),
2- minimizar o tempo total de conclusão de todas as tarefas no sistema (*Makespan*).

A primeira opção pode ser usada pelo gerente da produção quando o volume da produção não está alto. Desta forma, é possível ter todos os lotes terminados com folga antes do final do período. O resto do período (em que a planta ficaria inativa) pode ser usado para a fabricação de componentes de reserva (caso haja interesse), ou para a manutenção dos equipamentos, por exemplo.

O segundo caso irá fazer com que todos os lotes terminem no menor tempo possível. Desta forma, quando se prevê a possibilidade de que um volume alto na produção possa vir a gerar algum atraso, o gerente pode ordenar que todos os lotes fiquem prontos o mais rápido possível.

A escolha por parte do gerente, pela abordagem 1 ou 2, fará com que o cálculo do *lower-bound* no passo sete seja feito de forma diferente. A descrição dos algoritmos do limitante inferior (*lower-bound*) para o caso de minimizar o tempo médio de permanência encontra-se em FERNANDES (1991; apêndice III) e para o caso de minimizar o *makespan*, a descrição está apresentada em BAKER (1974; páginas 193 a 195).

Em nossa proposta inicial, cada nó da árvore representava uma seqüência composta de todas as tarefas programadas entre dois conflitos. Em nossa implementação, cada tarefa corresponde exatamente a um nó da árvore. Isso se deve ao fato de que na proposta original há uma complexidade maior em termos de implementação computacional (sem nenhum ganho, a não ser o didático) ao se implementar uma estrutura de dados em que cada nó, ao invés de uma tarefa, contém uma lista de tarefas de tamanho variável e desconhecido. Por outro lado, os nós da proposta original teriam que ser decompostos no momento da reprogramação, pois todas as tarefas nele incluídas teriam que ter seus tempos de início e fim recalculados. Por este motivo, optamos por utilizar a mesma estrutura de listas (Lista1, Anterior, Coc e Solução) desenvolvida para o arquivo Nodes, que representa a árvore de operações e conflitos obtida no passo 6, e que será explorada no passo sete. Desta forma, os conflitos anotados no passo seis são possibilidades que devem ser desenvolvidas no passo sete, por meio do algoritmo do tipo *branch-and-bound* desenvolvido com essa finalidade.

Foi criado o campo ARQUIVO

Para o correto encadeamento dos nós da árvore, foi criado o campo PAI, que contém o número do registro pai de cada determinado nó. O campo IRMÃO contém o número do nó irmão, no caso de se tratar de um nó conflito. Foi criado também o campo NI (número do irmão), que contém a numeração de qual é o irmão do conflito.

A figura 6 mostra a estrutura do arquivo Nodes, que contém a programação obtida no passo seis, juntamente com os conflitos, a serem explorados no passo sete.

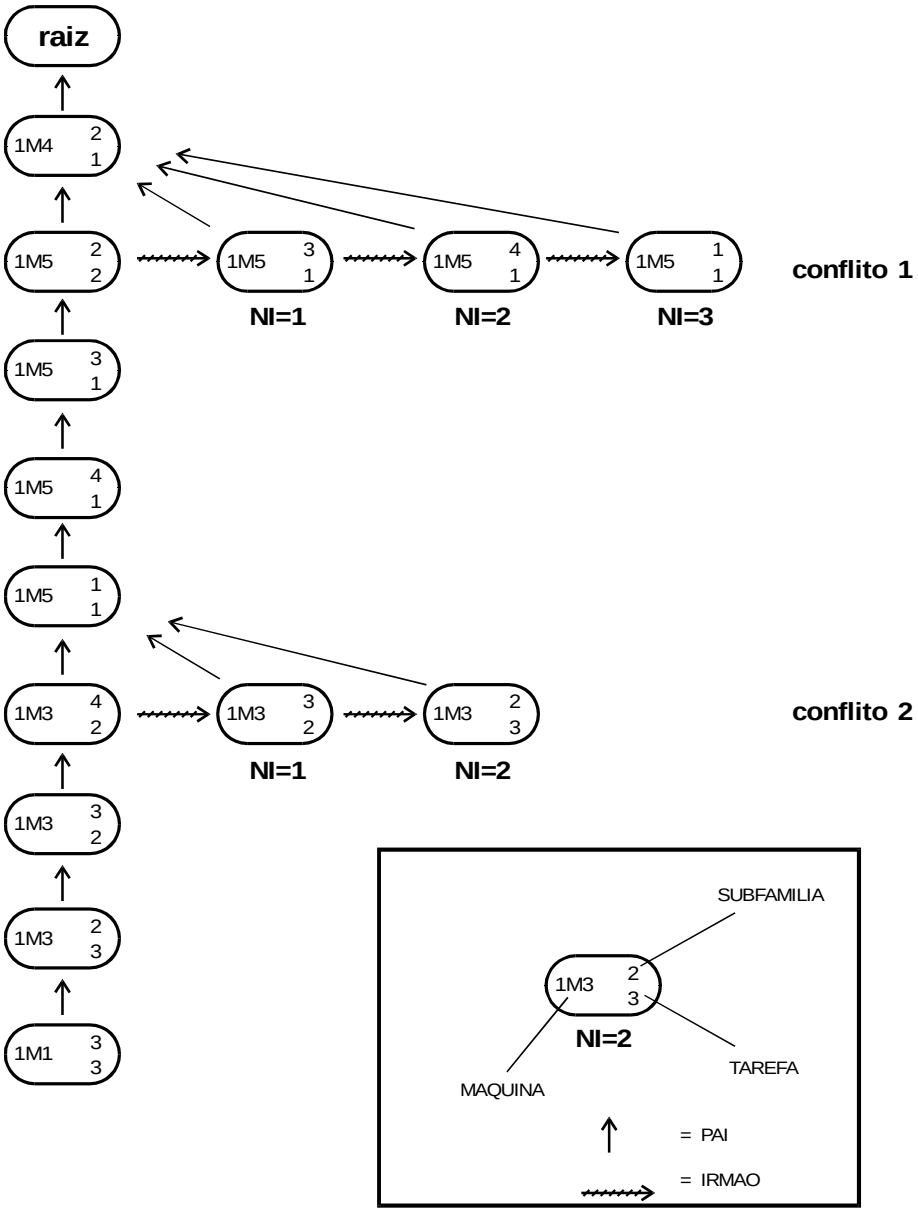

Figura 6: Estrutura do Arquivo Nodes

5. O SEMPRO

Nesta seção são apresentados alguns detalhes a respeito da implementação computacional do SEMPRO.

5.1 A Implementação do PBC Modificado

Para utilizar o sistema, deverão estar corretamente cadastrados nele:

1- O PMP escolhido (calculado pelo módulo SADEPMEP: artigo Parte I);
2- O período a ser considerado;
3- O tempo total alocado para produção nesse período;
4- O arquivo de produtos;

5- O arquivo de estrutura de produtos;
6- O arquivo de peças (e respectivos tempos de produção e de *setup*).

Os arquivos deste módulo são definidos pelos dados acima e pelas listas definidas anteriormente (as listas G e N e a lista E que deve ser preenchida (opção de Cadastros) em separado pelo operador do sistema, informando quais as peças que não foram completadas no período anterior e em que quantidades).

5.2 A implementação do PROCEL

Na figura 7 apresentamos a tela de chamada do módulo de programação de operações (PROCEL)

A seguir apresentamos alguns dos arquivos definidos no módulo:

ARQUIVO	FINALIDADE
ROTFAB	É o arquivo que contém o roteiro de fabricação de cada peça cadastrada. No roteiro constam as máquinas, a ordem das operações, os tempos de *setup* absoluto para cada operação e os tempos de produção unitários por máquina, além da célula produtora.
OP	Arquivo que guarda provisoriamente a primeira peça da subfamília que será processada. É o resultado do passo 3.
OSUBFMAQ	Este arquivo guarda a seqüência cíclica das operações de cada peça pertencente a cada subfamília, em cada máquina do seu roteiro de fabricação. Portanto, armazena o resultado do passo 4 do Procel.
TSUBFMAQ	Este arquivo guarda o resultado do passo 5 do Procel, ou seja, para cada subfamília e máquina, armazena o tempo total de ocupação da máquina, somando-se todos os tempos dos integrantes da subfamília, na seqüência que leva ao tempo minimizado.
DIFSETUP	Tabela gerada pelo usuário e utilizada no passo 4 do Procel. Guarda para cada peça, em cada máquina, quais são os tempos de setup relativos às outras peças da mesma subfamília.
CONFLI	Arquivo que armazena os conflitos encontrados quando da programação de operações efetuada pelo passo 6 do Procel. Os conflitos encontrados quanto a possibilidades de alocação de duas tarefas na mesma máquina no mesmo momento são aqui armazenadas para posterior tratamento pelo passo 7.
OS	Arquivo que contém o resultado da programação de operações obtida no passo 6 do Procel. As informações são: para cada tarefa de cada

	subfamília e para cada máquina, o tempo de início, tempo total de processamento e momento de conclusão de cada tarefa, seja ela pertencente à lista E, G ou N.
NODES	Contém a mesma estrutura do arquivo OS. O passo 7 tem como resultado final o arquivo NODES.

Os dois relatórios de saída gerados são: a exportação da programação de operações ao CA-Superproject (um conhecido *software* para gerenciamento de projetos) e a seqüência das peças de cada subfamília em cada máquina.

É importante esclarecer que a programação de operações obtida no arquivo NODES está colocada em termos de segundos, a partir do tempo zero. Assim, por exemplo, uma operação pode ter início no tempo 23.451, levar 200 segundos na máquina 1M1, e terminar no tempo 23.651. O que fizemos nessa opção de exportar o resultado para o Superproject foi converter este valor, calculado em segundos, numa data e hora de início da operação (dia, hora e minuto), bem como a duração e conclusão. Isso foi feito transformando esses valores a contar do dia e da hora de início de operação do PMP calculado.

O Superproject proporciona tanto a importação quanto a exportação de projetos por meio de sete arquivos DBF de *layout* específico para essa finalidade, na qual cada um desses sete arquivos contém um tipo de informação referente ao projeto (tarefas, recursos necessários, relações, *links* entre tarefas, etc).

Obtemos portanto uma programação de operações de excelente visualização gráfica, e cujo acompanhamento diário pode ser feito por meio do próprio Superproject.

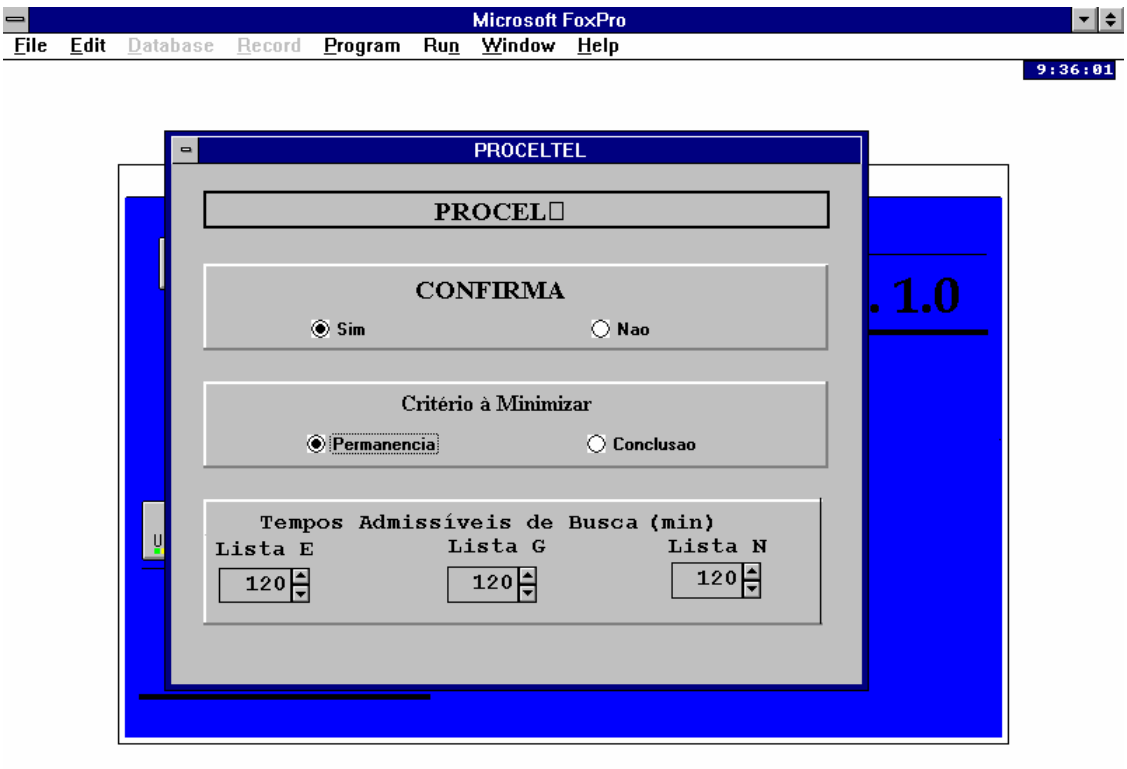

Figura 7: Tela de Chamada do Módulo de Programação de Operações

5.3 Alguns Testes Computacionais

Devido ao fato de que não é possível comparar o desempenho dos algoritmos de Emissão de Ordens e Programação de Operações (SEMPRO) com outro algoritmo (desenvolvido para outro ambiente de manufatura e em outra linguagem), procuramos testar o SEMPRO em várias situações alterando uma variável de cada vez e então concluir qual é a variação do desempenho do sistema em função da alteração nas variáveis do problema (número de itens, complexidade da estrutura de produtos, tempo de processamento das operações, etc).

Os exemplos resolvidos (PACHECO, 1995) não incluímos neste artigo, já que as tabelas de dados, mesmo para o menor exemplo, compreendem várias páginas. A partir dos testes realizados, uma primeira conclusão é a de que, independentemente da estrutura dos produtos e do PMP a ser produzido, o tempo de execução do módulo PBC Modificado é praticamente desprezível, ou seja, praticamente todo o tempo de processamento fica tomado pelo módulo de programação de operações.

No caso de se pretender minimizar o tempo médio de permanência, o critério de rejeição utilizado é o clássico: se LB>UB o nó é descartado. Para o outro caso (minimizar o *makespan*) criamos o seguinte critério de rejeição:

Sendo u= número de conflitos
c= conflito sendo analisado
Para u = 1 (só há um conflito)
Se LB > UB o nó é descartado
Para $U \geq 2$
Se LB > UB / {(1+0.10 * (u-c)) / (u-1)} o nó é descartado

Com este segundo critério a árvore a ser pesquisada tem um tamanho menor, porém, eventualmente, podemos estar descartando algum nó da árvore que nos forneceria uma solução com melhor desempenho que o daquela que vai ser obtida. Precisaríamos ter feito um número grande de testes para cada um dos casos de critério de desempenho, com cada um dos critérios de rejeição, para podermos concluir em que situação é preferível um ou outro critério de rejeição.

Na tabela 1, apresentamos os tempos gastos na resolução de três exemplos sob o mesmo critério de otimização, variando-se o tamanho dos problemas. O número de máquinas por célula foi 20 no terceiro exemplo (em geral, as células em problemas reais têm menos do que 15 máquinas).

Tabela 1: Tamanho do Problema e Tempo Dispendido na Solução

prod	peça	nodes E	nodes G	nodes N	conflitos	tempo	critério de minimização
5	12	6	1	18	0	216 seg	Tempo de Conclusão
12	22	8	3	41	13	3177 seg	Tempo de Conclusão
25	43	28	6	80	31	16069 seg	Tempo de Conclusão

Pela tabela 1, pode-se ver que o tempo necessário para a resolução do problema é mais fortemente influenciado pela quantidade de elementos no arquivo Nodes e

principalmente pelo número de conflitos (arquivo Conflito).

Todos os testes foram efetuados num **microcomputador AT 386 DX 40 MHz (sem coprocessador aritmético) com 8 Mb de memória RAM**. A execução em equipamentos mais velozes com certeza irá reduzir em muito esses tempos, pois em quase todo o tempo, as operações são efetuadas na memória principal, sendo que o número de acessos a disco é muito pequeno. Podemos afirmar que o sistema, executado num equipamento Pentium, deverá ter um desempenho muito melhor. Segundo HALFHILL (1994) "desde seu lançamento, o desempenho da arquitetura 80x86 tem aumentado num ritmo notavelmente constante. O primeiro 8086 tinha *clock* de 5 Mhz e executava cerca de 330.000 instruções por segundo. O atual 80x86 topo de linha é o Pentium P54C, de 100 Mhz que obteve 166,3 Mips em *benchmarks*" (figura 8).

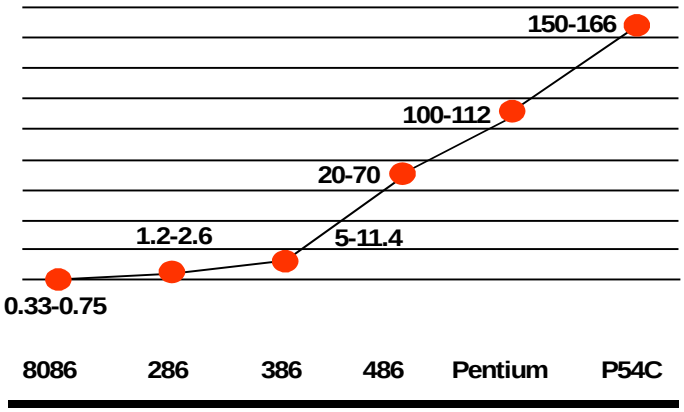

Figura 8: Evolução do 80X86: Desempenho
Extraído de: HALFHILL (1994)

Outra importante observação é que o programa foi executado em modo interpretado. Espera-se um desempenho bastante melhorado quando o mesmo programa for gerado em modo executável, utilizando-se o módulo chamado *Distribution Kit* para o Foxpro for Windows. Além disso, as rotinas de cálculo mais lentas poderão ser implementadas em C++, caso se deseje aumentar ainda mais a velocidade de execução do sistema.

Considerando-se que até sexta-feira à tarde pode ser feita a entrada dos pedidos que irão compor o PMP do período seguinte, o módulo de programação de operações poderá ser executado uma vez por semana (devido ao sistema de emissão de ordens empregado, o PBC Modificado), tendo para isso pelo menos 12 horas disponíveis (a partir de sexta-feira à noite até o sábado pela manhã).

Concluindo, não temos nenhuma razão para acreditar que o tempo de processamento seja empecilho para a utilização prática do sistema.

Na tabela 2 apresentamos uma comparação entre os tempos dispendidos, num mesmo problema, quando buscamos a minimização primeiramente do Tempo de Conclusão (*makespan*), e depois a minimização do Tempo Médio de Permanência. Em todos os casos deixamos o computador pesquisar todos os conflitos, ou

seja, o tempo de processamento foi o necessário para pesquisar todos os conflitos.

Tabela 2: Comparação de Tempos Entre as Duas Abordagens

prod	peça	nodes E	nodes G	nodes N	conflitos	tempo	critério de minimização
5	12	6	1	18	0	226 seg	Tempo Med permanência
5	12	6	1	18	0	216 seg	Tempo de conclusão
12	22	8	3	41	13	3257 seg	Tempo Med permanência
12	22	8	3	41	13	3177 seg	Tempo de Conclusão
25	43	28	6	80	31	16614 seg	Tempo Méd permanência
25	43	28	6	80	31	16069 seg	Tempo de Conclusão

Para finalizar a seção, vale ressaltar que, pelos testes acima apresentados, em vista dos resultados obtidos, pudemos constatar primeiramente a exatidão dos algoritmos propostos e a coerência do detalhamento e da implementação. Pudemos observar que o número de conflitos encontrados afeta fortemente o tempo de busca no passo sete. Entretanto, é importante observar também que a heurística do passo seis apresenta, nos exemplos executados, uma boa solução. Poderíamos dizer que nas situações em que há pouco tempo disponível para o processamento, podemos esperar uma solução satisfatória fornecida pelo passo seis.

6. Conclusões e Considerações Finais

Na extensa bibliografia pesquisada (algumas centenas de artigos lidos), não encontramos outro trabalho como este (este artigo e o artigo da Parte I), que visa integrar todas as atividades de programação da produção dentro do contexto de um Sistema de Controle da Produção concebido para operar na manufatura celular e, em particular, na manufatura celular semi-repetitiva.

O trabalho (este artigo e o artigo da Parte I) sustenta a hipótese de que um sistema de Controle da Produção especialmente concebido para operar no ambiente da Manufatura Celular semi-repetitiva é desejável e possível. O sistema concebido parte de alguns pressupostos que são bastante gerais e que, do ponto de vista prático, não restringem suas perspectivas de aplicabilidade. Tais pressupostos são:

(a) a integração das atividades de manufatura muito depende do desempenho do controle da produção;

(b) controle da produção é uma atividade gerencial que, para ser realizada com êxito, deve ser conduzida de forma hierárquica (nível de produto, de componente e de operação);

(c) esses 3 níveis devem ser conduzidos num horizonte de curto prazo;

(d) o processo decisório deve ser norteado pelos princípios e conceitos que envolvem a criação dos SADs.

O sistema proposto (neste artigo e no artigo da Parte I) compreende:

(a) o projeto de um Sistema de Apoio à Decisão em que, por meio da interface

usuário/sistema, o usuário interage com um modelo I, um algoritmo H e a base de dados para elaboração do Programa Mestre de Produção. Notemos que a elaboração de forma eficiente do programa mestre de produção é uma lacuna mesmo nos sistemas de controle da produção mais importantes (por exemplo, MRPII e OPT);

(b) o modelo I, que é um modelo de programação inteira cuja função-objetivo se diferencia das tradicionais usadas na programação da produção;

(c) o algoritmo H, que apresenta como novidade a forma de calcular a capacidade disponível de uma célula de manufatura, e um conjunto original de procedimentos para atribuição de peças às células, incluindo o conceito de *ranking* multiplicativo;

(d) o PBC Modificado, que por si só já representa uma significativa contribuição ao controle da produção;

(e) o algoritmo PROCEL, o qual leva em conta as peculiaridades da manufatura celular semi-repetitiva e do PBC Modificado, propõe modificações em alguns métodos consagrados e combina parcela significativa da teoria relevante sobre *scheduling*, PROCEL é um algoritmo original de dois estágios para a programação de operações, que procuramos conceber sob rigor lógico e técnico.

Com a implantação do sistema proposto em fábricas reais, podemos esperar os seguintes benefícios:

(a) ganhos com a melhoria na geração de receitas da empresa, devido ao objetivo colocado para a programação da produção: a maximização do valor presente da receita gerada;

(b) induzido pela adoção de conjuntos balanceados de peças, o volume de estoques fica represado ao nível do absolutamente necessário;

(c) melhor aproveitamento da capacidade das células dada a possibilidade de se trabalhar com células primárias e secundárias;

(d) atendimento mais rápido aos clientes, devido aos ciclos curtos de produção e ao esquema proposto de atribuição de prioridades às peças;

(e) para conseguir disciplina e eficiência produtiva no chão das células de fabricação, o supervisor de cada célula poderá contar com o PROCEL, que é uma ferramenta poderosa para a programação de operações;

(f) a implementação do PBC Modificado (Sistema de Emissão de Ordens) vem trazer os seguintes benefícios: 1) totalização dos lotes em termos de peças da mesma subfamília, gerando uma formação mais racional dos lotes a serem produzidos; 2) a flexibilidade proporcionada pelo seqüenciamento dos lotes de peças dentro de um esquema de prioridades, de acordo com a situação atual da fábrica, isto é, de acordo com a situação dos lotes de peças em atraso ou dos lotes com pouco tempo disponível antes da data de entrega prometida; 3) possibilidade de se optar pela célula primária ou secundária quando da fabricação de cada peça; 4) maior rapidez no atendimento dos clientes, por possibilitar que trabalhemos com um programa padrão de pouca duração; 5) e em decorrência de (4), temos pequenos estoques em processo. É importante observar que o sistema PBC, dada a sua simplicidade e as características do sistema de produção ao qual se aplica (no caso a manufatura celular), pode ser implantado sem necessidade de recursos computacionais (o que também é verdadeiro para o sistema de emissão de ordens Kanban). Já o PBC Modificado deve ser implementado computacionalmente para depois ser implantado e aplicado, já que se torna difícil gerenciar o esquema de

atribuição de prioridade às peças, e sem ele o PBC com período duplo de fabricação não funcionaria, conforme explicado anteriormente.

A consecução desses benefícios fica facilitada pelo envolvimento direto do usuário no processo decisório, devido às características de apoio à decisão do sistema proposto.

Um dos méritos do trabalho é abordar o problema como um todo, sem grande sofisticação matemática, com grande atenção à tratabilidade computacional dos modelos e algoritmos propostos e grande cuidado para que as hipóteses adotadas fossem o menos possível restritivas, uma vez que pretendemos que o sistema venha a ser implantado no futuro, com grande chance de êxito.

Vale também observar que embora o sistema implementado computacionalmente seja ainda um protótipo, consideramos que constitui um bom ponto de partida para uma evolução e melhorias, de modo a ficar pronto para ser utilizado em situações práticas. Para completar a implementação computacional devem ser desenvolvidos: a) a depuração total do código PLI em FoxPro; b) juntar dentro de um mesmo sistema o módulo PMP (o SADPMEP) e o módulo Emissão de Ordens + Programação de Operações (o SEMPRO); c) melhorar o desempenho computacional do PROCEL, que pertence ao SEMPRO, por exemplo emplementando em C++ determinadas subrotinas mais demoradas e/ou muito acionadas (o FoxPro permite que parte do projeto seja desenvolvido em C++); d) utilizar o *distribution kit* do FoxPro para obter o programa executável.

Após a implementação computacional completa do sistema, coordenaremos sua implantação em alguma fábrica pertinente, para efetuar uma avaliação prática do seu desempenho e procurar então aprimorá-lo, visando garantir que seja aplicável com êxito em situações reais. Uma observação que julgamos ser de interesse para aqueles que se deparam com o problema de implementar um SCP em computador é a de que, se fôssemos iniciar agora esta implementação, estaríamos realizando uma prévia comparação entre o Visual FoxPro e o Delphi, ambos recém lançados, para escolher um deles como *software* de desenvolvimento. Orientamos três bolsistas do Programa Especial de Treinamento (PET/CAPES) que realizaram comparações entre o FoxPro 2.5 e o Delphi 1.0. A conclusão foi a seguinte: o Delphi 1.0 é 10 a 30 vezes mais rápido do que o FoxPro 2.5 ao resolver o problema do caixeiro viajante. Segundo CORNELL & STRAIN (1996) o Delphi 1.0 tem um desempenho equivalente ao da linguagem C++ e, o tempo para desenvolver um sistema em Delphi 1.0 é um terço do tempo para desenvolvê-lo em C++.

Acreditamos que nosso trabalho atinge um ponto de equilíbrio entre os extremos: ser muito específico, e portanto com aplicabilidade muito reduzida; e ser muito geral, não criando vínculo com nenhuma situação prática.

Naturalmente, uma avaliação mais específica do sistema proposto só será possível após ser completada a implementação computacional e realizada então sua implantação em pelo menos algumas fábricas que se enquadrem na manufatura celular semi-repetitiva.

Pelo menos uma de nossas expectativas já se tornou uma realidade, a de ter criado com o tema uma linha de pesquisa dentro de um programa de Mestrado em Engenharia de Produção.

Referências Bibliográficas:

BAKER, K.R.: *Introduction to Sequencing and Scheduling.* John Wiley & Sons, New York, 1974.

BURBIDGE, J.L.: *The Introduction of Group Technology*. Heinemann, London, 1975.

BURBIDGE, J.L.: *Planejamento e Controle da Produção*. Atlas, 1983.

CORNELL, G. & STRAIN, T.: *Delphi - segredos e soluções*. Makron Books, São Paulo, 1996.

FERNANDES, F.C.F.: *Concepção de um Sistema de Controle da Produção para a Manufatura Celular*. Tese de Doutorado, Universidade de São Paulo - Escola de Eng. de São Carlos, 1991.

FERNANDES, F.C.F. & TAHARA, C.S.: "Um Sistema de Controle da Produção para a Manufatura Celular - Parte I: Sistema de Apoio à Decisão para a Elaboração do Programa Mestre de Produção". *Gestão & Produção*, v. 3, n. 2, p. 135-155, agosto de 1996.

FUSSE, A.A.: *Estudo da Implantação Conjunta do MRP e de Células de Manufatura*. Trabalho Final de Graduação, UFSCar-DEP, São Carlos, 1993.

HALFHILL, T.R.: "A Guerra dos 80x86". *Byte*, (6), p. 36-50, 1994.

KAKU, B.K. & KRAJEWSKI, L.J.: "Period Batch Control in Group Technology". *International Journal of Production Research*, 33 (1), p. 79-99, 1995.

LEE, L.C.: "A Comparative Study of the Push and Pull Production Systems". *Int. Journal of Production Management*, v. 9 (4), p. 5-18, 1989.

PACHECO, R.F.: *Um Sistema de Emissão de Ordens e Programação de Operações para a Manufatura Celular*. Dissertação de Mestrado, UFSCar - DEP, São Carlos, 1995.

SYSLO, M.M.; DEO, N. & KOWALIK, J.S.: *Discrete Optimization Algorithms with PASCAL Programs*. Prentice-Hall, Englewood Cliffs (NJ), 1983.

WHITE, C.H. & WILSON, R.C.: "Sequence Dependent Setup Times and Job Sequencing". *Int. J. of Production Research*, 17 (6), p. 631-641, 1979.

YANG, K.K. & JACOBS, R.: "Comparision of Make-to-Order Job Shops with Diferent Machine Layouts and Production Control Systems." *Int. Journal of Production Research*, v. 30 (6), p. 1269-1283, 1992.

ZACCARELLI, S.B.: *Programação e Controle da Produção* (8a. edição). Livraria Pioneira, São Paulo, 1987.

ZACCARELLI, S.B.: *Administração Estratégica da Produção*. Editora Atlas, São Paulo, 1990.

A PRODUCTION CONTROL SYSTEM TO THE CELLULAR MANUFACTURING. PART II: ORDERING SYSTEM AND OPERATIONS SCHEDULING

Abstract

This paper, along with that of Part I ("A decision support system for elaborating the master production scheduling"), constitutes part of a research project that aims to integrate all production scheduling activities in a semi-repetitive cellular manufacturing environment. In the paper of Part I we clarify what we mean by semi-repetitive cellular manufacturing, and in so doing, we make a production systems classification and a cellular manufacturing taxonomy. The proposed system was conceptualized and computationally implemented in three modules (product, components and operations levels). The first level is treated in the paper of Part I and the other two levels are treated in this paper.

Key words: production control, production scheduling, ordering system, operations scheduling, cellular manufacturing.