SIMULAÇÃO EM PLANILHAS PARA PROGRAMAÇÃO DE ORDENS DE PRODUÇÃO EM SISTEMAS JOB SHOP

Leonel Jose Girotti (POLI - USP)
leonel.girotti@usp.br
Viviane Satie Nishimura (POLI - USP)
vivi_nishimura@yahoo.com.br
Marco Aurelio de Mesquita (POLI - USP)
marco.mesquita@poli.usp.br

Sistemas de programação com capacidade finita dos recursos constituem uma alternativa interessante para a programação detalhada da produção. Neste artigo, apresenta-se um protótipo desenvolvido em planilhas que, via simulação por eventos diiscretos, realiza a programação detalhada das operações em um ambiente de produção do tipo Job Shop. Primeiramente, utilizam-se três das mais reconhecidas regras de despacho (FIFO, SPT e EDD), além do emprego da técnica chamada carregamento lateral, para gerar soluções iniciais do problema. A partir daí, o usuário pode modificar qualquer das sequências anteriores, gerando novas soluções. Todas podem ser comparadas mediante o uso de indicadores e gráficos de Gantt. Destaca-se que, devido à facilidade de visualização gráfica e interação com o usuário, o modelo proporciona uma visão integrada do problema de programação detalhada. Ainda que seja apenas um protótipo, oferece um panorama global das especificações e funcionalidades necessárias de um sistema profissional de programação da produção.

Palavras-chaves: programação de operações; job shop; regras de despacho; simulação em planilha.

1. Introdução

A habilidade de planejar e controlar as operações muitas vezes é a diferença entre sucesso ou fracasso em empresas de manufatura (VOLLMANN *et al*. 1997). A Teoria da Programação de Operações (*Scheduling*) contribui fornecendo modelos para tomada de decisão na Programação Detalhada da Produção, proporcionando maior eficiência na programação e controle das operações (CONWAY; MAXWELL & MILLER, 1967; MORTON & PENTICO, 1993; PINEDO, 1995; BAKER, 1997).

O enfoque predominante de programação nas empresas de manufatura envolve o uso de sistemas MRP. É sabido que esta lógica apresenta algumas deficiências especificamente na programação detalhada em ambientes de produção contra pedido com múltiplos roteiros. A mais importante decorre do fato de considerar capacidade *infinita* dos recursos, ao utilizar *lead times* fixos para a produção dos produtos.

Sistemas de programação com capacidade finita surgem como uma abordagem alternativa, considerando tempos de fila variáveis e a disponibilidade efetiva dos recursos. Atualmente, as ferramentas de programação finita são oferecidas em pacotes de *software* conhecidos no mercado pela sigla APS - *Advanced Planning and Scheduling*.

A complexidade da otimização na programação de operações está associada à natureza combinatória do problema. Para uma mesma configuração de layout de máquinas, podem ser definidos diferentes objetivos, como minimizar o tempo total de conclusão das ordens, minimizar atrasos, etc. Definidos a configuração e o objetivo do problema, busca-se um método eficiente que proporcione soluções boas (não necessariamente ótimas) dentro do geralmente vasto espaço de soluções.

Apesar de toda a contribuição acadêmica da área de *Scheduling*, o uso das técnicas e algoritmos desenvolvidos ainda é limitado em ambientes reais de produção. Frequentemente, os gerentes de produção tomam decisões de sequenciamento de forma relativamente empírica. Os custos e limitações dos softwares de programação finita disponíveis são fatores inibidores à difusão de técnicas de programação (HARRISON & PETTY, 2002).

A técnica da Simulação Computacional auxilia na tomada de decisões nos três níveis de planejamento da produção: estratégico, tático e operacional (SMITH, 2003). No nível operacional, a simulação pode ser aplicada para testar diferentes alternativas de solução de um problema específico de programação, possibilitando, em pouco tempo, verificar o resultado de diferentes métodos e regras de sequenciamento.

Regras de prioridade são critérios muito utilizados no sequenciamento de operações em problemas de programação da produção e podem simplificar bastante a geração de alternativas de solução. Eventualmente, podem até levar a soluções ótimas. No entanto, na maioria dos casos, não garantem a otimalidade (LUSTOSA *et al*., 2008).

O objetivo deste artigo é apresentar um modelo de sistema de programação finita, desenvolvido em planilhas, que proporcione uma visão geral da estrutura básica deste tipo de ferramenta. O modelo permite ao usuário definir problemas do tipo *job shop* e testar diferentes alternativas de solução, além das geradas inicialmente a partir do uso de regras de prioridade.

Na seção 2, apresentam-se algumas definições importantes sobre o objeto de estudo. Na seção 3, são descritos os elementos constituintes do sistema de programação desenvolvido. A seção

4 discute os resultados alcançados e as limitações do modelo. A seção 5 finaliza o artigo com as conclusões.

2. Revisão bibliográfica

Esta seção apresenta conceitos básicos e a notação usual da teoria de *Scheduling*, focando nos problemas referentes à configuração *job shop*.

2.1. Ambientes de programação

Do ponto de vista acadêmico, os problemas de *Scheduling* podem ser classificados, dentre outras formas, em função do número de máquinas e do padrão de fluxo das ordens em: máquina única, máquinas em paralelo, em série (*flow shop*) e oficina de máquinas (*job shop*) (GRAVES, 1981).

No primeiro caso, têm-se "n" ordens que devem ser processadas em uma única máquina, cada ordem com um tempo de processamento específico. No caso de máquinas em paralelo, há "n" ordens que devem ser processadas em uma de "m" máquinas usualmente consideradas idênticas. No *flow shop*, "n" ordens passam sequencialmente por "m" máquinas dispostas em série, cada operação com um dado tempo de processamento. Finalmente, no caso do *job shop*, as ordens de produção apresentam roteiros sequenciais diferentes, tanto em termos de quantidade de operações quanto de tempo de operação nas máquinas.

Além destes quatro casos básicos, as últimas duas configurações podem ser estendidas incorporando máquinas em paralelo em algum ou todos os estágios, dando origem aos ambientes *flow shop* flexível e *job shop* flexível, respectivamente. Mainieri e Ronconi (2010) apresentam um estudo de aplicação de regras de despacho no ambiente *flow shop* flexível. Existem sistemas não comerciais que consideram ambos os ambientes. Por exemplo, pode-se citar o software Lekin (versão acadêmica), desenvolvido pela Stern School of Business, NYU.

Neste artigo, considera-se apenas o *job shop* clássico (tendo o *flow shop* e "máquina única" como casos particulares), que envolve "m" máquinas e "n" pedidos (*jobs*). Existe só uma máquina em cada estágio. Cada *job* i compreende k_i operações em diferentes máquinas. Para cada *job*, definem-se:

- d_j (*due date*): prazo de conclusão do *job* (prazo de entrega);
- C_j (*completion time*): instante efetivo de término do job (término da última operação);
- F_j (*flow time*): tempo que o job permanece no sistema (*lead time*).

As operações possuem alguns atributos, sendo os principais apresentados a seguir:

- p_{ij} (*process time*): tempo de processamento da i-ésima operação do *job* j (incluindo tempo de *setup*) na máquina correspondente;
- r_{ij} (*ready time*): instante disponível para início do processamento da operação i do *job* j. Como os roteiros são sequenciais, a disponibilidade de uma operação corresponde ao instante de término da operação precedente, exceto a primeira operação do roteiro, cuja disponibilidade é a data de liberação da ordem de produção para a fábrica.

Devido à existência de múltiplos pedidos na fábrica, ocorre a formação de filas e ociosidade nas máquinas ao longo do tempo. A programação do *job shop* consiste em determinar as sequências de processamento das operações em cada máquina, respeitando as precedências das operações conforme o roteiro de produção. As diferentes soluções viáveis são avaliadas a partir de determinados objetivos referentes a prazo, estoques em processo e produtividade.

2.2. Regras de Prioridade

Uma forma simples de geração de soluções para programação é pela aplicação de regras de prioridade, que permitem decidir qual o próximo *job* a ser processado na máquina dentre aqueles que estão na fila. Existem várias regras de prioridade (mais de cem), dentre as quais se destacam: (BAKER, 1997):

- FIFO (*First In First Out*): seleciona o *job* que chegou primeiro na máquina;
- SPT (*Shortest Processing Time*): seleciona a operação de menor tempo de processamento;
- EDD (*Earliest Due Date*): seleciona o job com data de entrega mais próxima;
- LPT (*Longest Processing Time*): seleciona a operação de maior tempo de processamento;
- SST (*Shortest Setup Time*): seleciona a operação de menor tempo de preparação;
- MS (*Minimum Slack First*): seleciona o *job* com a menor folga;
- CR (*Critical Ratio*): seleciona o *job* com menor razão entre folga e número de operações;
- RND: seleciona-se aleatoriamente o próximo *job*, etc.

2.3. Medidas de Desempenho

A fim de avaliar a qualidade de uma solução, definem-se indicadores referentes a cada ordem isoladamente e ao conjunto das ordens (carteira). Os principais estão apresentados a seguir:

- L_j (*Lateness*): o desvio entre a data de conclusão e o prazo de entrega - $L_j = C_j - d_j$
- E_j (*Earliness*): a antecipação, se houver - $E_j = \max(0, -L_j)$
- T_j (*Tardiness*): o atraso, se houver - $T_j = \max(0, L_j)$
- n_T: número total de ordens atrasadas ($T_j > 0$)
- $T_{médio}$: atraso médio - $T_{médio} = \Sigma T_i / n$
- $T_{max:}$ atraso máximo - $T_{max} = \max(T_1,...,T_n)$
- C_{max} (*Makespan*): instante de término da última ordem - $C_{max} = \max(C_1,...,C_n)$
- $F_{médio}$: tempo médio de fluxo - $F_{médio} = \Sigma F_i / n$

2.4. Programação da Produção

Sule (1997) define programação da produção à função que consiste em planejar e priorizar as operações que necessitam ser executadas em sequência predeterminada. Suas atividades podem ser divididas em: (i) Programação: consiste no sequenciamento e programação das operações, e a elaboração da carga dos centros de trabalho; (ii) Execução: envolve a liberação das ordens, também chamada despacho, e (iii) Controle: inclui o apontamento da produção e o controle de seu status. A seguir, serão descritas com maior profundidade as etapas da programação, foco do presente artigo.

O *Sequenciamento* especifica a ordem em que devem realizar-se os trabalhos em cada centro. É muito comum utilizar regras de prioridade para a realização do sequenciamento.

O propósito da *Programação* é otimizar o uso de recursos para satifazer objetivos predefinidos, sujeito a múltiplas restrições. Stoop e Wiers (1996) detalham três tipos de perturbações, que afetam o real desempenho de uma programação, associadas a: (i) *Capacidade* (p.ex.: quebra de máquina); (ii) *Ordens* (p.ex.: indisponibilidade de materiais); e (iii) *Informação* (erros na medição de tempos e rendimentos).

Cabe ressaltar, de acordo com Stoop e Wiers (1996), que se a programação é realizada em nível de recursos, a mesma já contém uma sequência específica, com o que outras decisões acerca da ordem das operações por parte do pessoal de chão de fábrica não seriam necessárias.

Segundo McKay e Wiers (1999), a saída da programação é um conjunto de *jobs* sequenciados. Portanto, e em geral, isto consiste em agendar operações que habitualmente concorrem por recursos comuns. Devido a diversos fatores (absentismo, falhas de máquinas, etc.) essa determinação não assegura que as operações serão executadas exatamente de acordo com a programação. Cabe ao controle da produção monitorar e, quando necessário, executar ações corretivas.

Chase *et al.* (2006) distinguem duas abordagens para programação:

- Programação para trás: parte de alguma data futura (usualmente a data de entrega prometida) e programa as operações requeridas em sequência inversa. Indica o instante mais tarde que o pedido deve ser iniciado para estar acabado na data especificada. O MRP utiliza esta lógica com *lead times* estimados;
- Programação para frente: inicia a programação na data presente com as operações iniciais de cada *job*, avançando para o futuro com as operações subsequentes até a conclusão do *job* em produtos acabados. Indica a data mais próxima para o término do pedido. O modelo desenvolvido aqui utiliza este método.

A *Carga dos Centros de Produção* destina as ordens aos centros na medida em que as mesmas são liberadas de acordo com a programação. Chase *et al.* (2006) distingue aqui dois tipos, conforme é considerada a capacidade para determinar a carga.

- Método de carga infinita: lógica utilizada em sistemas do tipo MRP. Utilizam-se *lead times* fixos, que incluem um tempo de fila estimado; a programação e feita com base nestes tempos, e, mediante o módulo CRP (*Capacity Requirements Planning*), verifica-se posteriormente, para cada centro de trabalho, o nível de carga ao longo do tempo. Esta carga é comparada com a capacidade disponível, indicando ou não a necessidade de ajustes;
- Método de carga finita: programa cada recurso em detalhe, a partir do tempo de preparação da máquina e o tempo de processo requerido para cada operação. Em essência, o sistema determina com exatidão o trabalho de cada recurso durante a jornada. Teoricamente, salvo imprecisões nos parâmetros (tempos e rendimentos), todos os programas gerados são viáveis. Este é o método utilizado.

3. Apresentação do Modelo

O sistema foi desenvolvido inteiramente em planilhas de cálculo do software *MS Excel*, suportado por módulos programados mediante o complemento *Visual Basic for Applications* (VBA). A estrutura básica descrita a seguir, começa pelos dados de entrada da programação. Seguidamente, é apresentado como o modelo realiza a programação das operações, utilizando uma lógica de simulação por eventos discretos. As saídas (programação e indicadores, em gráficos e tabelas) são expostas no final da seção.

3.1. Entradas

Os dados de entrada básicos encontram-se em duas tabelas: a tabela de pedidos e a tabela de roteiros, descritas a seguir.

A tabela de pedidos, apresentada mediante um exemplo na Tabela 1, constitui a carteira de pedidos dos clientes. Cada pedido especifica um produto, a quantidade e o prazo de entrega (em dias úteis). No modelo, considera-se que todos os centros operam de 2ª a 6ª, com jornadas diárias de oito horas, exceto feriados previamente cadastrados. Isto permite uma programação das operações do tipo finito, contemplando a limitação dos recursos em cada dia de trabalho.

Pedido	Produto	Qtd	Prazo (dias)	Prazo (data)
1	P1	100	1	06/06/2011 08:00
2	P3	150	3	08/06/2011 08:00
3	P5	200	3	08/06/2011 08:00
4	P2	250	5	10/06/2011 08:00
5	P4	300	5	10/06/2011 08:00
6	P1	200	6	13/06/2011 08:00

Tabela 1 – Carteira de pedidos

Os roteiros correspondentes são apresentados na Tabela 2. Ela indica, para cada produto, a sequência de operações para sua produção. Os produtos e recursos (máquinas) são previamente cadastrados em tabelas separadas. A tabela inclui também os tempos de *setup* e de produção unitária. Tempos de fila não são requeridos, pois são na verdade resultados da programação. Tempos de movimentação de um *job* entre uma operação para outra não são considerados na versão atual. O sistema atualmente considera roteiros sequenciais, i.e., não contempla operações de montagem, e tampouco considera máquinas em paralelo (*job shop* flexível), conforme mencionado anteriormente.

Produto	Operação	Recurso	S [min]	T [min]
P1	1	M1	15	1,8
P1	2	M3	10	1,8
P1	3	M4	14	1,2
P2	1	M1	15	1,5
P2	2	M4	15	1,2
P3	1	M2	8	1,5
P3	2	M3	10	1,6
P3	3	M5	16	1,2
P4	1	M2	15	0,6
P4	2	M3	15	1,2
P4	3	M4	15	1,2
P4	4	M5	15	0,6
P5	1	M2	15	2,4
P5	2	M3	15	3
P5	3	M4	15	0,6

Tabela 2 – Tabela de roteiros

Especificados os produtos, quantidades e prazos de entregas, utilizam-se a tabela de roteiros e os tempos para gerar a lista de operações, apresentada parcialmente na Tabela 3, que devem ser programadas. A codificação utilizada para identificar uma operação específica apresenta a estrutura "JxPyOw", onde 'x' é o número de *job*, 'y' o produto, e 'w' a ordem da operação no roteiro do produto. Por exemplo: J2P3O5 identifica a operação 5 do produto 3 solicitado no *job* 2. Além da máquina, pedido, produto e ordem, para cada operação definem-se o tempo e o prazo (*due date*).

CodOp	Máq.	Ped.	Prod.	Op.	p [h]	dd
J1P1O1	M1	1	P1	1	3,25	06/06/2011
J1P1O2	M3	1	P1	2	3,17	06/06/2011
J1P1O3	M4	1	P1	3	2,23	06/06/2011
J2P3O1	M2	2	P3	1	3,88	08/06/2011
J2P3O2	M3	2	P3	2	4,17	08/06/2011

J2P3O3	M5	2	P3	3	3,27	08/06/2011

Tabela 3 – Lista de Operações

3.2. Programação

Uma vez adicionados todos os dados de entrada, o sistema os processa gerando de maneira automática três soluções iniciais por regras de despacho (FIFO, SPT e EDD) e mais uma solução denominada "carregamento lateral". Destaca-se que cada solução é gerada em uma aba diferente, evitando assim a perda de informação. Da forma como foi implementado, na plataforma *MS Excel / VBA*, usuários experimentados podem incluir no protótipo outros algoritmos específicos de solução (heurísticos ou otimizantes). Na Tabela 4, pode-se observar o resultado da programação de um exemplo com 5 máquinas e 6 *jobs*.

Seq.	CodOp	Máq.	Ped.	Prod.	Op.	p [h]	dd	tidh	tfdh
1	J1P1O1	M1	1	P1	1	3,25	06/06/2011	02/06/2011 08:00	02/06/2011 11:15
2	J4P2O1	M1	4	P2	1	6,50	10/06/2011	02/06/2011 11:15	06/06/2011 09:45
3	J6P1O1	M1	6	P1	1	6,25	13/06/2011	06/06/2011 09:45	07/06/2011 08:00
1	J2P3O1	M2	2	P3	1	3,88	08/06/2011	02/06/2011 08:00	02/06/2011 11:53
2	J3P5O1	M2	3	P5	1	8,25	08/06/2011	02/06/2011 11:53	06/06/2011 13:08
3	J5P4O1	M2	5	P4	1	3,25	10/06/2011	06/06/2011 13:08	06/06/2011 16:23
1	J1P1O2	M3	1	P1	2	3,17	06/06/2011	02/06/2011 11:15	02/06/2011 15:25
2	J2P3O2	M3	2	P3	2	4,17	08/06/2011	02/06/2011 15:25	06/06/2011 10:35
3	J3P5O2	M3	3	P5	2	10,25	08/06/2011	06/06/2011 13:08	07/06/2011 15:23
4	J5P4O2	M3	5	P4	2	6,25	10/06/2011	07/06/2011 15:23	08/06/2011 13:38
5	J6P1O2	M3	6	P1	2	6,17	13/06/2011	08/06/2011 13:38	09/06/2011 10:48
1	J1P1O3	M4	1	P1	3	2,23	06/06/2011	02/06/2011 15:25	06/06/2011 08:39
2	J3P5O3	M4	3	P5	3	2,25	08/06/2011	07/06/2011 15:23	08/06/2011 08:38
3	J4P2O2	M4	4	P2	2	5,25	10/06/2011	08/06/2011 08:38	08/06/2011 14:53
4	J5P4O3	M4	5	P4	3	6,25	10/06/2011	08/06/2011 14:53	09/06/2011 13:08
5	J6P1O3	M4	6	P1	3	4,23	13/06/2011	09/06/2011 13:08	10/06/2011 08:22
1	J2P3O3	M5	2	P3	3	3,27	08/06/2011	06/06/2011 10:35	06/06/2011 14:51
2	J5P4O4	M5	5	P4	4	3,25	10/06/2011	09/06/2011 13:08	09/06/2011 16:23

Tabela 4 – Tabela resultado da programação

A mesma oferece, para cada máquina, a sequência de operações a serem executadas. Destaca-se que dentre as informações se visualizam os tempos reais de início e fim de cada operação (carregamento finito), considerando uma jornada de trabalho de 40 horas semanais.

Continuando, o sistema permite que qualquer das quatro soluções iniciais seja manipulada pelo usuário, gerando novas soluções. Basta modificar a ordem das operações em cada máquina (primeira coluna da tabela) e atualizar a solução. Neste caso, o sistema adiciona uma aba onde se apresenta a nova programação, que também poderá servir de base para novas soluções do usuário.

3.2.1. Simulação por eventos discretos

O procedimento para obter a programação das operações no modelo pode ser entendido como uma simulação por eventos discretos (vide Banks; Carson e Nelson, 1996) onde os eventos correspondem ao início e término de operações nas máquinas ao longo do tempo.

A lógica de simulação por eventos discretos consiste basicamente no controle da Lista de Eventos Futuros. No modelo em questão, há apenas um tipo de evento futuro a ser monitorado: o término de uma operação.

Quando uma operação é concluída, dois novos eventos podem ocorrer: (i) uma nova operação está disponível e é carregada na máquina, gerando um evento futuro correspondente ao término previsto da operação recém-carregada; (ii) se a operação recém-concluída for a última do *job*, encerra-se o mesmo, caso contrário, o *job* e direcionado instantaneamente para a próxima máquina da sua sequência; na máquina aonde chega, o *job* aguarda em fila (máquina ocupada) ou entra imediatamente em operação (máquina ociosa), neste caso, gerando um evento futuro correspondente ao término da operação iniciada. Processado o evento, avança-se o relógio da simulação para o instante do próximo evento, isto é, a próxima operação em curso na oficina que irá terminar.

A simulação considera um calendário contínuo, i.e., utiliza tempo contínuo medido em horas. Feita a programação em tempo contínuo, para fornecer as datas de começo e fim das operações, o resultado é recalculado já considerando as restrições de calendário anteriormente mencionadas.

A seguir, apresentam-se as duas formas específicas de resolução do problema de programação das operações: (i) a aplicação de regras de prioridade e (ii) a programação pelo usuário (sequências predefinidas), que inclui o método "carregamento lateral".

3.2.2. Programação via regra de despacho

Neste caso, cada uma das máquinas possui uma "fila", onde os *jobs* permanecem enquanto a máquina estiver ocupada. Para cada máquina, há um estado inicial da fila contendo as primeiras operações a serem processadas nela e que estavam disponíveis no instante zero. Como mencionado anteriormente, a lógica do modelo é a simulação por eventos discretos, baseada na Lista de Eventos Futuros.

A inicialização da simulação é feita carregando cada máquina com a operação com maior prioridade dentre aquelas que estão na fila. Cada uma das operações carregadas gera um evento futuro correspondente ao término da operação, que é indicado na Lista de Eventos Futuros.

Uma vez gerados os eventos iniciais, prossegue-se uma rotina que deve ser repetida enquanto houver eventos futuros não processados, ou seja, ordens não concluídas. Esta lógica está apresentada no Quadro 1.

```
Enquanto houver eventos futuros faça:
    Início
        Avançar o Relógio para o instante do próximo evento futuro (término de uma operação)
        Retirar o job da máquina e registrar o instante de término (tf) da operação
        Se houver operações na fila, então:
            Tirar da Fila a operação com maior prioridade e carregar na máquina
            Gerar evento futuro correspondente ao término da operação ora carregada
        Senão: Máquina fica ociosa
        Se o job que estava na máquina não está concluído, então:
            Transferir o job para a próxima máquina
            Se a máquina está ociosa na chegada do job, então:
                Carregar o job na máquina
                Gerar evento futuro correspondente ao término da operação ora carregada
            Senão: incluir o job na Fila
    Fim
```

Quadro 1- Programação via regra de despacho

3.2.3. Programação com sequência predefinida e carregamento lateral

Nesta modalidade, todas as operações são carregadas no principio em cada uma das máquinas, já que a sequência em que elas vão ser processadas não depende da definição feita mediante uma regra de despacho.

No caso da modificação de uma solução anterior por parte do usuário (sequência predefinida), as operações são carregadas nas máquinas segundo a sequência definida por ele. O carregamento lateral é um caso particular onde as operações são sequenciadas *job* a *job*, i.e, primeiro as operações do *job* 1, depois as operações do *job* 2, e assim por diante.

Para cada máquina, carrega-se a primeira operação a ser executada, caso essa esteja habilitada (*ready time*). Cada uma das operações carregadas gera um evento futuro, que é indicado na Lista de Eventos Futuros, elaborada da mesma forma que no caso de regras de despacho.

Uma vez gerados os eventos iniciais, prossegue-se com uma rotina que deve ser repetida enquanto houver eventos futuros (operações restantes), conforme detalhado no Quadro 2. Comparando esta rotina com a anterior, vale destacar que, ao contrário do caso de uso das regras de despacho, o usuário pode definir uma sequência inviável onde não é possível atender as restrições de precedência das operações e a sequência definida por ele. Estas situações serão chamadas de "bloqueio" ou "travamento" da produção. Também, para compreender a diferença entre as lógicas empregadas, pode-se imaginar que, enquanto no caso do uso de regras de despacho, os *jobs* movimentam-se de uma máquina para outra ao término de cada operação, no segundo, as operações que serão executadas em cada máquina são conhecidas a priori e vão sendo "habilitadas" a medida que as operações precedentes no roteiro da ordem são completadas. Em outras palavras, a ideia de "movimentação" dos *jobs* é substituída pela "habilitação" da próxima operação a medida que as precedentes no roteiro e na máquina forem finalizadas.

```
Enquanto houver eventos futuros faça:
    Início
        Avançar o Relógio para o instante do próximo evento futuro (próxima operação concluída)
        Retirar o job da máquina e registrar o instante de término (tf) da operação
        Se houver jobs ainda não processadas na máquina, então:
            Se a próxima operação estiver habilitada (op. anterior no roteiro concluída), então:
                Carregar o job na máquina
                Gerar evento futuro correspondente ao término da operação ora carregada
            Senão: máquina fica ociosa
        Se o job que estava na máquina não estiver concluído, então:
            Habilitar próxima operação do roteiro dele (atualizar o ready time)
            Se a máquina da próxima operação no roteiro estiver ociosa, então:
                Carregar a operação na máquina
                Gerar evento futuro correspondente ao término da operação ora carregada
    Fim
```

Quadro 2 – Programação via sequência predefinida

3.3. Saídas

Para cada solução (programa) gerada, o sistema fornece três informações padrões: a tabela da programação das operações (Tabela 4), um gráfico de Gantt e os indicadores (por ordem e global). Isto pode ser visualizado na Figura 1.

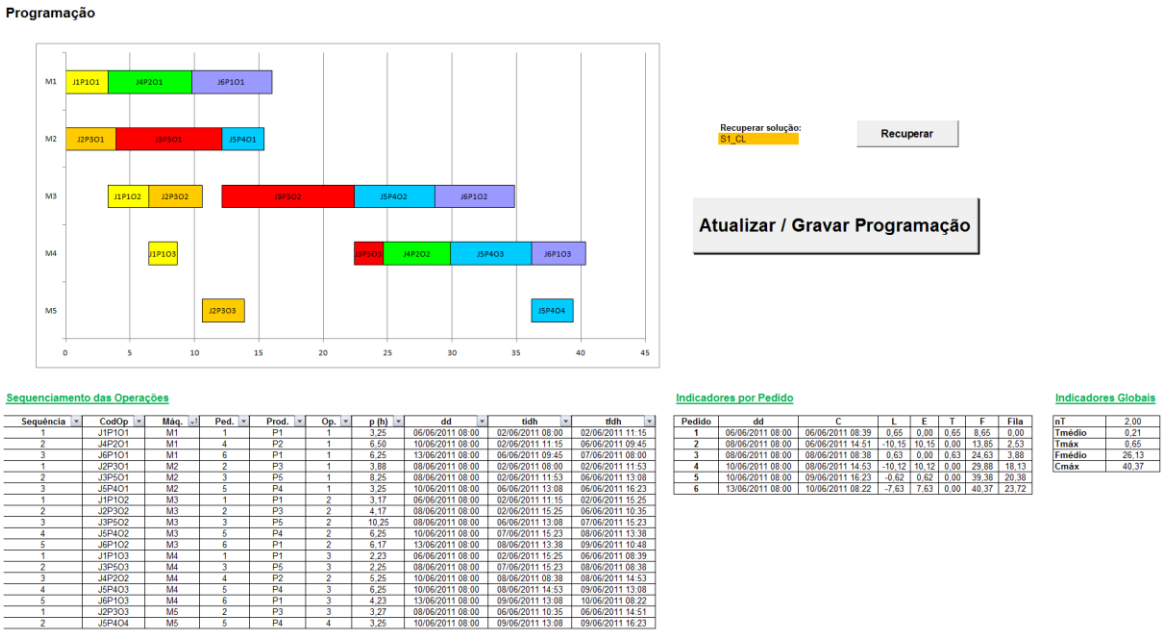

Figura 1 – Aba padrão do protótipo

A partir de uma programação dada, constrói-se o gráfico de Gantt, que é uma representação gráfica das operações já alocadas em cada um dos recursos. Um exemplo de gráfico, obtido com a regra SPT, é mostrado na Figura 2.

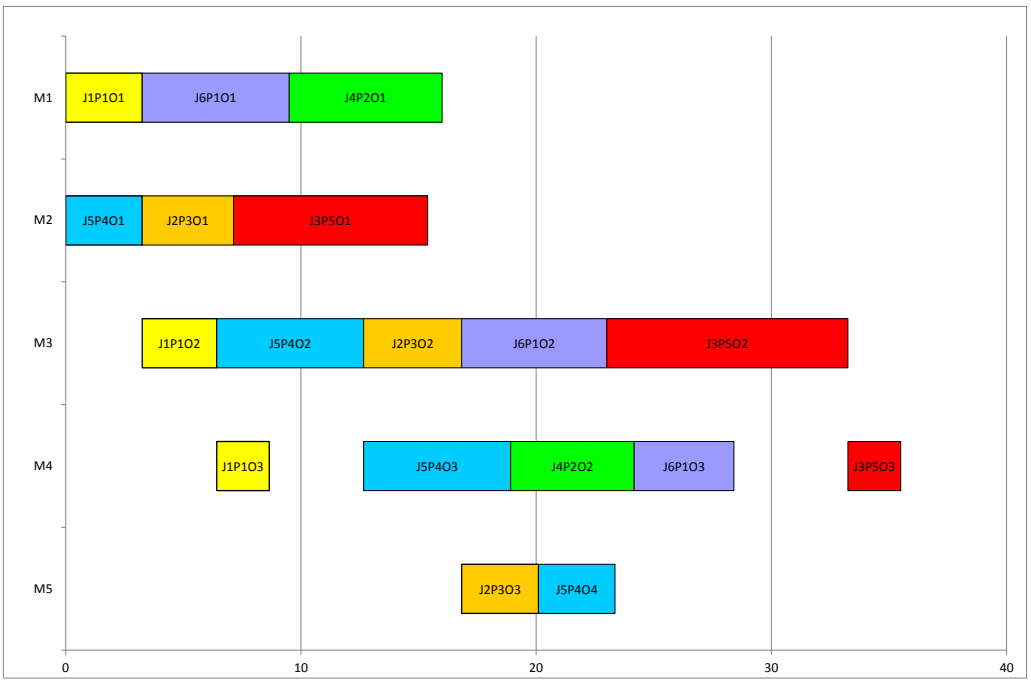

Figura 2 – Exemplo de Gráfico de Gantt para solução SPT

A construção do gráfico parte, evidentemente, da tabela da programação de cada máquina. Utiliza-se a opção "gráfico de barras horizontal empilhado" do *MS Excel*. Para elaborar o gráfico, cria-se uma tabela auxiliar que intercala períodos ociosos e ocupados, cada período com uma duração determinada. Quando não há ociosidade entre duas operações, ou seja, quando duas operações são realizadas consecutivamente, atribui-se valor zero à ociosidade. Os segmentos associados às ociosidades são representados "em branco" no gráfico, enquanto os segmentos referentes às operações são apresentados com o rótulo do código da operação.

A seguir são calculados os indicadores de desempenho. Primeiro são calculados os "Indicadores dos pedidos" (Tabela 5). Eles indicam o desvio com respeito à data de entrega do pedido, através de várias medições (antecipação, atraso, etc.), além do tempo que o *job* permaneceu no sistema e a fração do mesmo que esteve esperando alguma máquina.

Pedido	Due Date	Completion time	Lateness [h]	Earliness [h]	Tardiness [h]	Tempo de fluxo [h]	Fila [h]
1	06/06/2011 08:00	06/06/2011 08:39	0,65	0	0,65	8,65	0
2	08/06/2011 08:00	06/06/2011 14:51	-10,15	10,15	0	13,85	2,53
3	08/06/2011 08:00	08/06/2011 08:38	0,63	0	0,63	24,63	3,88
4	10/06/2011 08:00	08/06/2011 14:53	-10,12	10,12	0	29,88	18,13
5	10/06/2011 08:00	09/06/2011 16:23	-0,62	0,62	0	39,38	20,38
6	13/06/2011 08:00	10/06/2011 08:22	-7,63	7,63	0	40,37	23,72

Tabela 5 – Indicadores dos pedidos

Continuando, são calculados os valores chamados de "Indicadores Globais". Eles permitem avaliar a qualidade da programação como um todo. As medições utilizadas, apresentadas na Tabela 6, incluem: número de ordens atrasadas, atraso médio, atraso máximo, tempo médio de fluxo e tempo total de processamento das ordens (*makespan*).

Indicador	Valor
n_T	2
$T_{médio}$	0,21
T_{max}	0,65
$F_{médio}$	26,13
C_{max}	40,37

Tabela 6 – Indicadores globais

A planilha "Resumo" foi criada para facilitar as comparações dentre as diferentes soluções (as quatro originais mais quaisquer outras geradas pelo usuário). Nela se expõe uma síntese contendo apenas os indicadores numéricos. Visualiza-se também um quadro comparativo (Tabela 7) mediante o uso do formato condicional de três cores para estabelecer um "ranking" de cada um dos indicadores globais avaliados.

Solução	n_T	$T_{médio}$	T_{max}	$F_{médio}$	C_{max}
S1_CL	2	0,21	0,65	26,13	40,37
S2_FIFO	2	0,21	0,65	23,23	39,12
S3_SPT	2	2,02	11,50	23,36	35,50
S4_EDD	2	0,21	0,65	23,23	39,12
Sequência de usuário n° 1	2	1,21	6,58	24,21	37,48
Sequência de usuário n° 2	2	1,07	5,75	23,62	37,48

Tabela 7 – Indicadores globais para todas as soluções

Até aqui, foram apresentados os componentes básicos do modelo de sistema de programação

detalhada da produção considerando capacidade finita (provocada neste caso pela restrição nas horas trabalhadas). Na seguinte seção, serão discutidos os resultados e as limitações do modelo desenvolvido.

4. Discussão

4.1. Resultados

O modelo desenvolvido fornece uma visão geral do problema da programação, através da apresentação dos dados de entradas, procedimentos para geração das soluções e saídas do modelo. Além disso, o usuário pode modificar uma dada solução de forma iterativa até conseguir uma solução melhor para o problema em análise, conferindo os resultados mediante os indicadores desenvolvidos. Ainda, a planilha permite a recuperação de qualquer solução anterior.

Outro destaque é o fato de oferecer soluções iniciais (regras de despacho e carregamento lateral), o que facilita a análise do problema oferecendo múltiplas opções para começar o processo de melhoramento da solução. Quanto às regras de prioridade, foram disponibilizadas inicialmente três (FIFO, SPT e EDD). No entanto, a inclusão de novas regras e até mesmo o uso de métodos heurísticos mais elaborados podem ser implementados sobre o protótipo.

Ao ser desenvolvido numa plataforma bem conhecida, isto é, o aplicativo *MS Excel* em conjunto com a linguagem VBA, o modelo de planilha apresentado pode ser incrementado, o que significa que melhorias podem ser facilmente codificadas e incluídas no sistema. De qualquer forma, em seu estado presente, proporciona uma visão global das funcionalidades e requisitos que um sistema profissional deve oferecer para atender às necessidades reais da programação detalhada da produção.

4.2. Limitações

Algumas das limitações do modelo estão associadas à plataforma de desenvolvimento (Excel/VBA). Enquanto os cadastros de entrada de dados (pedidos, produtos, máquinas, roteiros) e a Lista de Operações têm o número praticamente ilimitado (número de linhas da planilha), para o gráfico de Gantt, o aplicativo oferece limitações maiores referentes à quantidade de máquinas e operações executadas em cada uma delas (número máximo de séries no diagrama de barras empilhadas).

Quanto à modelagem, o protótipo considera apenas o *job shop* clássico, onde há uma única máquina por estágio e cada operação tem no máximo uma operação precedente (roteiros sequenciais). Evidentemente, é possível utilizar o simulador também para os casos de *flow shop* clássico e máquina única. A inclusão de máquinas em paralelo exigiria um tratamento especial, pois não se trata apenas de determinar a sequência em cada máquina, mas também de alocar em qual máquina cada operação será executada. A possibilidade de considerar roteiros mais gerais constitui uma nova fronteira a ser explorada.

Uma limitação que se planeja suprimir em próximas versões diz respeito ao momento da liberação de um pedido. Atualmente o sistema considera que todas as ordens estão liberadas desde o momento zero, i.e., o *ready time* da primeira operação do *job* é sempre zero.

Outra limitação que se pretende suprimir diz respeito ao uso do calendário do tipo contínuo utilizado nos gráficos de Gantt. Esta limitação já está contemplada na tabela de programação das operações, onde as datas de início e fim das operações levam em consideração o cadastro de dias feriados, além do horário normal de trabalho, excluindo sábados e domingos. Também é desejável que em uma próxima versão o sistema considere outras variáveis referentes ao calendário, por exemplo, num problema real, têm-se turnos diferentes por centros, paradas

programadas para manutenção das máquinas, etc. A possibilidade de realização de horas-extras também altera o tempo disponível nos turnos de trabalho.

5. Conclusões

Neste artigo foram apresentados os componentes básicos de um sistema de programação da produção através da apresentação de um protótipo de *software* de programação utilizando capacidade finita.

O sistema desenvolvido atende a proposta inicial de criar um ambiente de programação que apresente soluções iniciais e permita ao usuário recuperar e modificar qualquer solução produzida anteriormente. Das várias soluções geradas, escolhe-se a que melhor atende aos objetivos da programação.

O modelo apresentado tem sido utilizado tanto para o ensino de Programação da Produção quanto de Simulação. Pode ser utilizado também com plataforma para entrada e saídas de dados em pesquisas na área de scheduling. Conforme mencionado anteriormente, uma nova heurística pode ser incorporada diretamento no prótipo, programando-a em um módulo específico do VBA.

A continuidade deste projeto aponta para o aprimoramento de algumas das limitações mencionadas no ponto anterior. A facilidade conhecida na programação modular via VBA e o uso do *MS Excel* são incentivos para o melhoramento contínuo.

Referências

ALBRIGHT, S.C. *VBA for modelers: developing decision support systems with Microsoft Excel*. Pacific Grove, CA: Duxbury; Thomson Learning, 2001.

BAKER, K. *Elements of sequencing and scheduling*. Hanover, N.H.: K.R. Baker, 1997.

BANKS, J.; CARSON II, J.S. & NELSON, B.L. *Discrete – event system simulation*. 2. ed. Upper Saddle River, N.J.: Prentice Hall, 1996.

CHASE, R.B.; JACOBS, F.R. & AQUILANO, N.J. *Administração da produção para a vantagem competitiva.* 10. ed. Porto Alegre: Bookman, 2006.

CONWAY, R.W.; MAXWELL, W.L. & MILLER, L.W. *Theory of Scheduling.* Reading, Mass. : Addison-Wesley Pub. Co., 1967.

GRAVES, S.C. *A Review of production Scheduling*. Operations Research, 1981.

HARRISON, P. K. & PETTY, D. J. *Systems for Planning and Control in Manufacturing*. Oxford: Elsevier Science, 2002

LEKIN. *Software académico de programação da produção.* Disponível em: <http://www.stern.nyu.edu/om/software/lekin/> . Acceso em: 10 de Outubro de 2010.

LUSTOSA, L. et al. *Planejamento e Controle da Produção.* Rio de Janeiro: Elsevier: Campus, 2008.

MAINIERI, G.B. & RONCONI, D.P. *Regras de despacho para a minimização do atraso total no ambiente flow shop flexível.* Revista Gestão e Produção. Vol. 17, n.4, p.683-692, 2010.

McKAY, K.N. & WIERS, V.C.S. *Unifying the Theory and Practice of Production Scheduling.* Journal of Manufacturing Systems. Vol. 18, n.4, p.241-255, 1999.

MORTON, T.E. & PENTICO, D.W. *Heuristic Scheduling Systems. With Applications to Production Systems and Project Management.* New York: Wiley, 1993.

MORTON, T.E. *Production Operations Management.* South-Western College, 1999.

PINEDO, M.L. *Planning and Scheduling in Manufacturing and Services*. New York, NY: Springer, 2005.

PINEDO, M.L. *Scheduling: theory, algorithms, and systems*. New Jersey: Prentice Hall, 1995.

RAGSDALE, C.T. *Spreadsheet modeling & decision analysis: a practical introduction to management science*.

4. ed. Mason, Ohio: Thomson; South-Western, 2004.

SMITH, J.S. *Survey on the use of simulation for Manufacturing System Design and Operation*. Journal of Manufacturing Systems. Vol. 22, n.2, p.157-171, 2003.

STOOP, P.P.M. & WIERS, V.C.S. *The complexity of scheduling in practice.* International Journal of Operations & Production Management. Vol. 16, n.10, p.37-53, 1996.

SULE D. R. *Industrial Scheduling.* Boston: PWS Pub. Co., 1997.

VOLLMANN, T.E.; BERRY, W.L. & WHYBARK, D.C. M*anufacturing Planning and Control Systems*. 4. ed. Irwin, Illinois, 1997.