Utilização da Busca Tabu para a Geração de um Modelo Aplicado ao *Job-Shop Scheduling Problem* Considerando um Sistema de Manufatura Flexível

Gilberto Irajá Müller
Universidade do Vale do Rio dos Sinos – Unisinos
irajamuller@terra.com.br

Arthur Tórgo Gómez
Universidade do Vale do Rio dos Sinos – Unisinos
breno@exatas.unisinos.br

Resumo

Este trabalho tem como objetivo a geração de um modelo visando a resolução do Job-Shop Scheduling Problem num Sistema de Manufatura Flexível. O modelo é baseado numa arquitetura que possuí cinco fases. A primeira fase baseia-se na extração da demanda de produção. Já a segunda fase, aplica a Tecnologia de Grupo para a geração de Famílias de Partes. Na terceira fase é utilizada a regra de despacho para a solução inicial e, na quarta fase, é aplicado o algoritmo Busca Tabu para o escalonamento. A última fase consiste na gravação do plano de produção. Os resultados e análises obtidas são apresentados no final deste trabalho.

***Palavras-chave**: job-shop, busca tabu, regras de despacho, tecnologia de grupo, sistemas de manufatura flexível.*

1. Introdução

Um *Job-Shop* consiste de um conjunto de "n" *jobs* a serem processados num conjunto de "m" máquinas. Cada *job* é composto de "i" operações que devem ser processadas através de um roteiro; para cada operação é definido uma máquina com tempo de processamento padrão[3][11]. O objetivo é minimizar o tempo total de produção [21].

Existem algumas restrições para os *jobs* e as máquinas: (i) não existe regra de precedência entre as operações de diferentes partes, (ii) as operações que iniciarem o processo não podem ser interrompidas e uma máquina pode somente processar um *job* por vez e (iii) um *job* pode somente ser processado em uma máquina por vez [3][11].

Das diversas formulações para o *Job-Shop* [3], é apresentada abaixo a formulação clássica utilizada para o modelo proposto [1], onde, V = {0, 1, ..., n} representa o conjunto de operações, onde "0" é a primeira operação de todos os *jobs* e "n" será a última operação para todos os *jobs*. O conjunto de "m" máquinas é representado por "M" e "A" é a representação para o conjunto de pares ordenados das restrições de operações pela precedência das relações de cada *job*. Para cada máquina "k", o conjunto de "Ek" descreve todos os pares de operações fornecidos pela máquina "k". Para cada operação "i" é processado num tempo "pi" (fixo) e o processo inicial de "i" é "ti", uma variável que tem sido determinada durante a otimização. A partir destas definições, o modelo pode ser definido como:

$$\min t_n \quad (1)$$

$$t_j - t_i \geq p_i \;\forall (i,j) \in A, \quad (2)$$

$$t_j - t_i \geq p_i \text{ ou } t_i - t_j \geq p_j \;\forall \{i,j\} \in E_k, \;\forall k \in M, \quad (3)$$

$$t_i \geq 0 \quad \forall i \in V. \quad (4)$$

Figura 1. Formulação clássica do JSSP [1].

A função objetivo (1) para o *Job-Shop* busca minimizar o tempo total de produção. A restrição (2) assegura que a seqüência de processamento das operações para cada *job* corresponde a uma ordem pré-determinada. Já a restrição (3) é a demanda que existe, ou seja, somente um *job* em cada máquina num determinado tempo e a restrição (4) assegura o término de todos os *jobs*.

O *Job-Shop* num Sistema de Manufatura Flexível é de difícil complexidade, pois é NP-Difícil [7].

Para o modelo proposto, foram utilizados como base na literatura existente, os trabalhos desenvolvidos em [9][20].

O artigo está organizado da seguinte forma: na Seção 2, apresenta-se os problemas abordados para o *Job-Shop Scheduling Problem* (JSSP) num Sistema de Manufatura Flexível (SMF); na Seção 3, é apresentado o conceito de Regras de Despacho; já na Seção 4, apresenta-se o algoritmo Busca Tabu; na Seção 5, é conceituado o modelo proposto; na Seção 6, são apresentados os resultados dos experimentos; e, na Seção 7, são realizados os comentários finais.

2. Problemas abordados

Nesta seção são apresentados dois problemas que são abordados na geração do modelo proposto aplicado ao JSSP. O primeiro problema está relacionado com a fase pré-operacional e consiste na seleção de Famílias de Partes (FP) [9][19]. Já o segundo problema é abordado na fase operacional, e diz respeito ao escalonamento das partes, respeitando os recursos de produção e datas de entrega [9].

2.1. Seleção de famílias de partes

A grande dificuldade em implementar a Tecnologia de Grupo (TG), num SMF é a seleção de FP [16].

Existem diversos métodos para a geração de FPs, sendo destacados a Inspeção Visual [15], Classificação por Codificação [11] e Análise por Fluxo de Produção (formulação matricial, programação matemática e particionamento de grafos) [5][13][18].

A Seleção de FP é resolvida através do agrupamento das partes, baseada nas similaridades, tais como: forma geométrica, processo, similaridade por um mesmo conjunto de ferramentas entre outros [12][15]. O problema da seleção de FPs é considerado NP-Completo [10].

Em indústrias, na existência de um grande número de estilos de partes diferentes e um grande número de operações que devem ser processadas, resulta na queda de desempenho do SMF. Através da seleção de FP é possível obter um processamento simultâneo por uma FP, respeitando assim a capacidade fabril [9].

Para resolver este problema, foi utilizado o método de Análise por Fluxo de Produção através da formulação matricial proposto por Kusiak, 1987 denominado de algoritmo de Identificação de Agrupamento.

2.2. Escalonamento de partes

Conforme mencionado anteriormente, o escalonamento das partes num *Job-shop* (JSSP) é um problema de difícil solução, da classe NP-Difícil [7]. Num SMF, o objetivo está em seqüenciar os *jobs* num conjunto de máquinas através de um roteiro pré-estabelecido em função do tempo [3].

O escalonamento das partes pode ser definido através de diversos objetivos, dependendo da realidade do foco da indústria. Num JSSP tradicional, o objetivo principal é a minimização do tempo total de produção (makespan) [3]. Contudo, podem-se utilizar outros objetivos, tais como: minimizar as trocas de ferramentas, minimizar os tempos de atraso, maximizar a produtividade, entre outros [11].

Inúmeros métodos de otimização foram desenvolvidos para a solução do JSSP [3][17][21], sendo destacado os métodos de otimização e os métodos aproximativos. Para os métodos de otimização são citadas a Programação Inteira, a Relaxação Lagrangeana, as Técnicas de Surrogate e o *Branch and Bound* [2][4][6]. Já nos métodos aproximativos são destacados os algoritmos iterativos Busca Tabu, as Redes Neurais, os Algoritmos Genéticos, a Têmpera Simulada e o GRASP [3][8][11].

Para resolver o problema do escalonamento de partes, foi utilizado o algoritmo Busca Tabu e as Regras de Despacho.

3. Regras de despacho

As Regras de Despacho são mecanismos que determinam qual o próximo *job* ou lote a ser processado de acordo com os objetivos e o plano de produção [14].

A utilização das Regras de Despacho para o escalonamento se dá ao fato destas obterem respostas computacionais em tempo polinomial, visto que muitos destes problemas devem ser resolvidos em tempo real. A seguir, algumas Regras de Despacho [14]:

- Regra randômica (RR): as partes são seqüenciadas aleatoriamente;
- Recursos mais dissimilares (RMD): o seqüenciamento é realizado de forma que uma parte compartilhe o menor número de recursos com a parte seguinte no seqüenciamento;
- Família de Partes (FP): as partes são seqüenciadas de forma crescente segundo o número de FP à qual pertencem;

- Processos mais longos primeiro (PMLP): a partir do tempo de processamento para cada parte, estas são seqüenciadas de forma crescente;
- Recursos de famílias mais similares (RFMS): o seqüenciamento é gerado tal que uma parte compartilhe o menor número de recursos com a próxima parte no escalonamento;
- Datas de entrega mais recente (DEMR): as partes que possuem a data de entrega mais recente, são as primeiras a serem seqüenciadas; e
- Datas de entrega mais longa (DEML): as partes que possuem o maior prazo de entrega, são as primeiras a serem seqüenciadas.

A partir da Regra de Despacho selecionada, o objetivo é gerar uma solução inicial que seja viável, ou seja, faça parte do espaço amostral das soluções possíveis.

Para o modelo proposto, será utilizado a Regra de Despacho FP, pois é possível agrupar e gerar lotes por similaridades e ainda a utilização de outra Regra de Despacho para cada FP gerada, como por exemplo, a regra DEMR.

4. Busca Tabu

A meta-heurística Busca Tabu (BT) teve origem a partir de uma solução para problemas de programação inteira; posteriormente foi dada uma descrição do método, para uso geral em problemas da área de Pesquisa Operacional (otimização combinatória) [1].

Basicamente, a BT, que foi projetada para encontrar boas aproximações para a solução ótima global de qualquer problema de otimização, possui três princípios fundamentais: (i) uso de uma estrutura de dados (lista) para guardar o histórico da evolução do processo de busca, (ii) uso de um mecanismo de controle para fazer um balanceamento entre a aceitação, ou não, de uma nova configuração, com base nas informações registradas na lista tabu referentes às restrições e aspirações desejadas e (iii) incorporação de procedimentos que alternam as estratégias de diversificação e intensificação [8].

Para a utilização do método BT, é fundamental a definição da função objetivo (F) do problema em questão. Após esta definição, é gerada uma solução inicial viável independentemente. Para geração da solução inicial, é obrigatório que esta faça parte do conjunto de soluções possíveis do espaço amostral. Sempre que uma solução "s" é obtida, é gerado um subconjunto V* de N(s) e realizado o movimento para a melhor solução s* em V*. Se N(s) não é muito grande é possível fazer V*= N(s) [8].

A Figura 2 ilustra o algoritmo BT para um problema de minimização.

```
s = solução inicial obtida através de uma regra de
despacho;
niter = 0;
melhiter = 0;
nbmax = 0;
melhsol = s;
Fmelhor = F(s);
Fmin = valor mínimo estimado da função;
Inicializar  a lista Tabu;
Inicializar a função do critério de aspiração A(z = F(s));

ENQUANTO (F(s) > Fmin)  OU  ( niter - melhiter <
nbmax)  FAÇA
            niter = niter +1;
            Gere um conjunto V* de soluções em N(s);
            Escolha a melhor solução s* em V* que não
seja tabu  ou F(s*) < A(F(s));
            Atualize a função do critério de aspiração A(z
= F(s))  e  a  lista  tabu
          retirando o movimento mais antigo;

            SE F(s*) < F(melhsol) ENTÃO
                        melhsol = s*;
                        melhiter = niter;
                        Fmelhor = F(s*);
            FIM SE
            s = s*;
FIM ENQUANTO

Onde,

s = solução inicial pertencente a S;
niter =  número de iterações;
melhiter = número da iteração correspondente a melhor
solução encontrada;
nbmax = número máximo de iterações sem
obter melhoria  na melhor solução obtida;
melhsol = melhor solução;
Fmelhor = valor da melhor solução;
Fmin = valor mínimo estimado da função;
V* = vizinhança gerada;
s* = melhor solução da vizinhança V*;
F = função objetivo;
A = função critério de aspiração;
```

Figura 2. Algoritmo Busca Tabu [8].

A cada iteração um ótimo local é escolhido e dele é gerada a nova vizinhança. Para evitar ciclos e mínimos locais, é implementada uma lista de movimentos proibidos chamada de lista tabu. Caso a iteração gere uma melhora na função F, utiliza-se a função critério de aspiração A, que admite o movimento proibido.

Para as condições de parada do algoritmo,

pode-se utilizar um número máximo de iterações (nbmax) sem que ocorra melhoria na função F ou executar até que alcance o valor mínimo (Fmin).

5. Modelo proposto

O modelo proposto apresenta uma formulação que é a função objetivo para a otimização do escalonamento das partes e uma arquitetura baseada no modelo hierárquico de SMF proposto por Stecke, 1986.

5.1. Formulação do modelo

A figura abaixo apresenta a formulação utilizada para a otimização do escalonamento das partes.

Minimizar

$$f(e,p) = p_1.makespan(e,p) + p_2.atraso(e,p) + p_3.setup(e,p) + p_4.ocioso(e,p) + p_5.(1 - produtividade(e,p)) \quad (5)$$

onde,

$$makespan(e,p) = \sum_{j=1}^{m} \sum_{i=1}^{n} makespan_{ij},\ makespan_{ij} > 0 \quad (6)$$

$$atraso(e,p) = \sum_{j=1}^{m} \sum_{i=1}^{n} atraso_{ij},\ atraso_{ij} > 0 \quad (7)$$

$$setup(e,p) = \sum_{j=1}^{m} \sum_{i=1}^{n} setup_{ij},\ setup_{ij} > 0 \quad (8)$$

$$ocioso(e,p) = \sum_{j=1}^{m} \sum_{i=1}^{n} ocioso_{ij},\ ocioso_{ij} > 0 \quad (9)$$

$$produtividade(e,p) = \sum_{j=1}^{m} \sum_{i=1}^{n} produtividade_{ij},\ produtividade_{ij} > 0 \quad (10)$$

$$p_1, p_2, p_3, p_4, p_5 \geq 0 \quad (11)$$

Figura 3. Função objetivo.

A função objetivo (5) a ser minimizada é definida pela variável "e" (escalonamento), que representa a dimensão temporal e a variável "p" (seleção de FP), representa a dimensão física para o modelo proposto. A função *makespan*(e,p) significa o tempo total de produção. Já a função atraso(e,p) representa o tempo total de atraso. O tempo de *setup* é representado pela função *setup*(e,p) e a função ocioso(e,p) significa o tempo não utilizado ao final dos turnos. A função produtividade(e,p) representa a produtividade, ou seja, é o coeficiente entre o tempo previsto e o tempo realizado. A restrição (6) assegura o somatório total do tempo de produção. Já a restrição (7) garante o somatório total do tempo de atraso. A restrição (8) assegura o tempo total de *setup*; o somatório do tempo total de produção ocioso, e conseqüentemente não utilizado ao final do turno é ilustrado através da restrição (9). A restrição (10) assegura a produtividade e, por fim, a restrição (11) garante a não-negatividade para os pesos associados a cada função.

O índice "m" ilustra a quantidade total de máquinas, já as partes, são representadas pelo índice "n".

A função objetivo conforme os valores definidos para os pesos (p_1, p_2, p_3, p_4, p_5), podem refletir as seguintes estratégias: minimizar o número de *setups*, ou seja, minimizar o número de lotes, minimizar o número de troca de ferramentas, minimizar o tempo total de produção, minimizar o tempo ocioso no final dos turnos, minimizar o tempo total de atraso e maximizar a produtividade [9].

5.2. Arquitetura do modelo

A arquitetura para a seleção de FP e escalonamento das partes está dividido em cinco fases de aplicação conforme é apresentado na Figura 4.

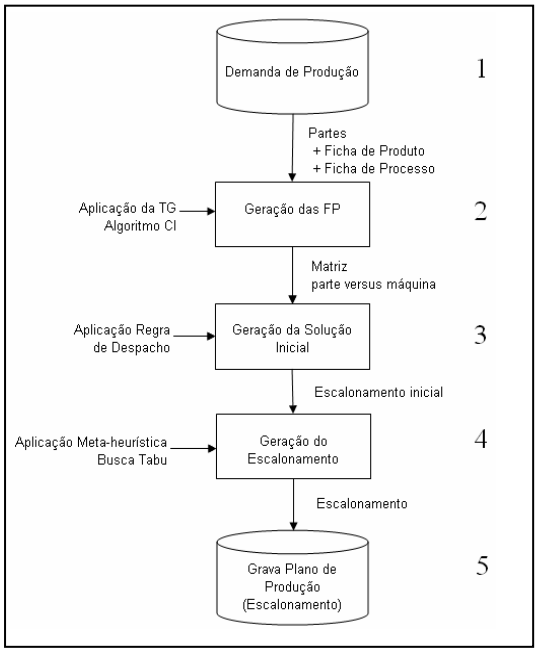

Figura 4. Arquitetura.

A arquitetura para a seleção de FP e escalonamento das partes está dividido em cinco fases de aplicação conforme é apresentado na Figura 4.

A primeira fase é responsável por obter a demanda de produção através das informações contidas no banco de dados, sendo recuperadas

as informações das partes e data de entrega. Já na segunda fase, é realizada através da TG, a geração da matriz parte versus máquina aplicando o algoritmo CI (*Cluster Identification*) para identificação de agrupamentos. Na terceira fase, é utilizado o algoritmo de Regras de Despacho para a geração do escalonamento inicial. As Regras de Despacho são aqueles algoritmos em que é obtido um escalonamento inicial através de regras relacionadas às partes e máquinas [14]. O escalonamento inicial deve ser viável, ou seja, deve ser uma solução que esteja contida no espaço amostral.

A partir do escalonamento inicial, é possível realizar a quarta fase que consiste na aplicação do algoritmo BT para a obtenção do escalonamento final. Após a geração do escalonamento final é gravado o escalonamento obtido em planos de produção, possibilitando assim observar seus históricos e compará-los com o escalonamento efetivo.

6. Resultados

Para os experimentos realizados, foi utilizado o banco de dados Oracle 8i em Linux para a extração e gravação dos dados (primeira e última fase da arquitetura, respectivamente). Após a extração dos dados, foi gerado o agrupamento através de uma matriz parte versus máquina. A utilização da Regra de Despacho FP (terceira fase da arquitetura) para o escalonamento inicial, baseado na função (formulação). O algoritmo BT para a otimização do escalonamento foi aplicado utilizando a geração de vizinhança a cada iteração. A geração da vizinhança, que está baseada a partir da seleção de um turno da máquina, utiliza a política de troca de lotes para a busca da melhoria da função. Dado que não encontre melhoria na função após 25% do "nbmax", a política é trocada para Inserção de partes e a geração da vizinhança aumentada para seleção de máquina, sendo possível inserir partes em outros turnos. Os algoritmos utilizados neste trabalho foram implementados em Delphi 7 e executados em ambiente Windows XP.

Foi utilizada uma lista tabu de 10 posições e um "nbmax" de 100 iterações. A matriz parte versus máquina está baseada na matriz proposta por Tang & Denardo, 1987. O objetivo do experimento é a validação do modelo proposto, bem como a aplicabilidade num SMF. A Figura 5 apresenta a matriz parte versus ferramenta utilizada nos experimentos deste trabalho. Em nosso modelo, é utilizado o conceito de matriz parte versus máquina. Contudo, o modelo proposto por Tang & Denardo, 1987 apresenta uma matriz parte versus ferramenta. A relação de máquina e ferramenta para efeitos de validação em nossos experimentos tem o mesmo conceito. A função que guiará a otimização para os experimentos é a minimização do número de trocas de ferramentas.

Parte \ Ferramenta	1	2	3	4	5	6	7	8	9
1	1			1				1	1
2	1		1		1				
3		1				1	1	1	
4							1		
5						1			
6			1						
7	1				1		1		1
8			1		1			1	
9					1		1		
10	1	1		1					

Figura 5. Matriz Parte x Ferramenta [7].

A Tabela 1 apresenta os resultados dos experimentos em relação aos resultados obtidos por Gómez, 1996. A função de otimização utilizada por Gómez, 1996 é a minimização do número de trocas.

Modelo	Solução inicial		Solução final	
	Setup	Nro de trocas	*Setup*	Nro de trocas
Gómez	5	11	4	7
Proposto	8	17	4	7

Tabela 1. Resultados.

Verifica-se na Tabela 1 que a solução proposta por Gómez, 1996 obteve resultado inicial melhor do que a solução proposta. Este fato ocorre, pois nos trabalhos desenvolvidos por Gómez, 1996 é utilizada a política de KTNS (*Keep Tool Needed Snoonest*). Contudo, a solução final obtida por ambos os modelos resultaram na mesma solução, fator este que valida o modelo proposto.

7. Comentários finais

Este trabalho teve como objetivo apresentar um modelo aplicado ao JSSP considerando um SMF. Esta nova abordagem visa a otimização do escalonamento de "m" máquinas num SMF

através de funções que guiam a BT.

Apesar do modelo proposto ter encontrado o mesmo resultado de Gómez, 1996, conforme Tabela 1, existem outras características no modelo que podem ser utilizadas, tais como a otimização pela função Produtividade e o escalonamento para cada máquina, visto que os modelos propostos por Tang & Denardo, 1987; Gómez, 1996 otimizam o conjunto de ferramentas numa CNC e não otimizam máquina por máquina num JSSP.

Embora não tenha sido considerada neste trabalho a comparação com trabalhos clássicos de JSSP em virtude das especificidades de cada ambiente, esta possa ser possível como extensão futura deste trabalho.

8. Referências bibliográficas

[1] Adams, J.; Balas, E.; Zawack, D. "The shifting bottleneck procedure for job shop scheduling", *Management Science*, 1988, 34, 57-73.

[2] Balas, E., Johnson, E.L., Korte, B. "Disjunctive programming. In: Hammer, P.L.", *Discrete Optimisation II*. North-Holland, Amsterdam, 1979, pp. 3±52.

[3] Blazewicz, Jacek; Domschke, Wolfgang; Pesch, Erwin. "The job shop scheduling problem: Conventional and new solution techniques", *European Journal of Operational Research*, 1996, 93, 1-33.

[4] Bowman, E.H., "The schedule-sequencing problem", *Operations Research 7*, 1959, 621±624.

[5] Choobineh, F. "A framework for the design of cellular manufacturing systems", *International Journal of Production Research*, 1988, 26(7), 1161–1172.

[6] Fisher, M.L., "A dual algorithm for the one-machine scheduling problem", *Math. Programming 11*, 1976, 229±251.

[7] Garey, M. R.; Johnson, D.S.; Sethi, R. "The complexity of flowshop and jobshop scheduling", *Mathematics of Operations Research* 1, 1976, 117-129.

[8] Glover, Fred; Laguna, Manuel. *Tabu Search*, Kluwer Academic Publishers, 1997, 382p.

[9] Gómez, Arthur T. "Seqüenciamento de Partes e Horários em um SMF composto de uma máquina com restrições de ferramentas", *Instituto Nacional de Pesquisas Espaciais*, São José dos Campos, São Paulo, 1996.

[10] Hwang , S. S. ; Shogan , A. W. "Modelling and solving an FMS part selection problem", *International Journal of Production Research* , 27 (8) : 1349 - 1366 , 1989.

[11] Jain, A.S.; Meeran, S. "Deterministic job-shop scheduling: Past, present and future", *European Journal of Operational Research*, 1998, 113(2), 390-434.

[12] Jha, Nand K. *Handbook of Flexible Manufacturing Systems*, Academic Press Limited, 1991, 328p.

[13] Kusiak, A.; Chow, W. "Efficient solving of the group technology problem", *Journal of Manufacturing Systems*, 1987, 6(2), 117–124.

[14] Kusiak, A. *Intelligent Design and Manufacturing*, John Wiley & Sons, Inc., 1992, 753p.

[15] Kusiak, A. Dorf, Richard C. *Handbook of Design, Manufacturing and Automation*, John Wiley & Sons, Inc., 1994, 1042p.

[16] Lorini, Flávio José. *Tecnologia de Grupo e organização da manufatura*, Editora da UFSC, 1993, 105p.

[17] Mascis, A.; Pacciarelli, D. "Job-shop scheduling with blocking and no-wait constraints", *European Journal of Operational Research*, 2002, 143, 498-517.

[18] Rajagopalan, R.; Batra, J. L. "Design of cellular production systems: a graph-theoretic approach", *International Journal of Production Research*, 1975, 13(6), 567–579.

[19] Stecke , K. E. "A hierarchical approach to solving machine grouping and loading problems of flexible manufacturing systems", *European Journal of Operational Research* , 24 : 369 - 375 , 1986.

[20] Tang , C. S. ; Denardo , E.V. "Models arising from flexible manufacturing machine, part I: minimization of the number of tool switches", *Operations Research*, 36 (5) : 767- 776, 1987.

[21] Zoghby, Jeriad; Barnes, Wesley L.; Hasenbein, John J. "Modeling the Reentrant job shop problem with setups for metaheuristic searches", *European Journal of Operational Research*, 2004, 167, 336-348.