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[1]
ARTHUR TÓRGO GÓMEZ[2]

Universidade do Vale do Rio dos Sinos - UNISINOS
PIPCA - Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Av. Unisinos, 950 - Bairro Cristo Rei - CEP 93.022-000 São Leopoldo - RS - Brasil
[1]irajamuller@terra.com.br
[2]breno@unisinos.br

Resumo. Neste trabalho é apresentado o desenvolvimento de um modelo de escalonamento aplicado ao *Job-shop Scheduling Problem*, num Sistema de Manufatura Flexível. O modelo considera como variáveis de decisão o tempo total de produção (*makespan*), o tempo total de atraso, o tempo total de paradas (*setup*) e o tempo total ocioso dos turnos de produção. O modelo proposto é composto por: (i) uma função objetivo que reflete, através de suas variáveis de decisão e seus respectivos pesos, as estratégias de otimização, e de (ii) uma arquitetura que está dividida em cinco fases. O modelo utilizou o algoritmo Busca Tabu que, através de duas estratégias de geração de vizinhanças, busca a otimização da função objetivo. A arquitetura do modelo baseia-se na extração da demanda da produção, em conceitos da Tecnologia de Grupo e de Regras de Despacho, no Algoritmo Busca Tabu e na gravação do plano de produção, para tratar os Problemas de Seleção de Partes (Famílias de Partes) e do Escalonamento. Os resultados e análises são apresentados ao final deste trabalho.

Palavras-Chave: Famílias de Partes, Busca Tabu, *Job-shop*, Sistemas de Manufatura Flexível.

Using Tabu Search for the Generation of Model Applied to Job-shop Scheduling Problem Considering a Flexible Manufacturing System

Abstract. In this paper it is presented the development of scheduling model applied to Job-shop Scheduling Problem, in a Flexible Manufacturing System. The model considers as variables of decision the makespan, total tardiness time, total stop times (setup) and total idle time of production turns. The model proposed is composed to: (i) an objective function that reflects, through its variables of decision and its respective weights, the optimization strategies, and an (ii) architecture that is divided in five phases. The model used the Tabu Search algorithm which, through two strategies neighborhoods generation, searching the objective function optimization. The model architecture is based on extraction of the production demand, in concepts of Group Technology and Dispatching Rules, in the Tabu Search algorithm and save production plan, to deal the Part Selections (Part Families) and Scheduling Problems. The results and analysis are presented in the end of this paper.

Keywords: Part Families, Tabu Search, Job-shop, Flexible Manufacturing Systems.

(Received March 23, 2006 / Accepted October 04, 2006)

1 Introdução

Um *Job-shop* consiste num conjunto de "n" *jobs* a serem processados em um conjunto de "m" máquinas. Cada *job* é composto de "i" operações que devem ser processadas através de um roteiro; para cada operação é definida uma máquina com tempo de processamento padrão [13]. O objetivo é minimizar o tempo total de produção [30]. Existem algumas restrições para os *jobs* e as máquinas: (i) não existe regra de precedência entre as operações de diferentes *jobs*, (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 [4][13]. Das diversas formulações para o *Job-shop* [4], é apresentada abaixo a formulação clássica utilizada para o modelo proposto [1][25], onde, V = 0, 1, ..., n representa o conjunto de operações; "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 "E_k" descreve todos os pares de operações fornecidos pela máquina "k". Para cada operação "i" é processado num tempo "p_i" (fixo) e o processo inicial de "i" é "t_i", uma variável que tem sido determinada durante a otimização. A partir destas definições, o modelo pode ser representado como:

$$min\ T_n \tag{1}$$

$$t_j - t_i \geq p_i \forall (i,j) \in A, \tag{2}$$

$$t_j - t_i \geq \ ou\ t_i - tj \geq p_j \forall i,j \in E_k, \forall k \in M, \tag{3}$$

$$t_i \geq 0\ \forall \in V. \tag{4}$$

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* corresponda 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 grande complexidade, pois é *NP-Hard* [8].

O artigo está organizado da seguinte forma: na Seção 2, são apresentados 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 e, na Seção 6, são apresentados os resultados da validação e as conclusões 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 do Sistema de Manufatura Flexível e consiste na Seleção de Partes [10][26]. Já, o segundo problema é abordado na fase operacional, e diz respeito ao Escalonamento de Partes, respeitando os recursos de produção e datas de entrega [10].

2.1 Seleção de Partes

A grande dificuldade em implementar a Tecnologia de Grupo (TG) em um SMF é a Seleção de Partes, ou seja, a geração das Famílias de Partes (FP) [21]. Existem diversos métodos para a geração de FPs, sendo destacados a Inspeção Visual [18], Classificação por Codificação [13] e Análise por Fluxo de Produção (formulação matricial, programação matemática, particionamento de grafos e heurísticas) [6][14][17]. A Seleção de Partes é resolvida, através do agrupamento das partes, baseado nas similaridades, tais como: forma geométrica, processo, similaridade por um mesmo conjunto de ferramentas entre outros [15][18]. O problema da Seleção de Partes é considerado NP-Completo [28].

Em indústrias, na existência de um grande número de estilos de partes e um grande número de operações, gera índices de baixa produtividade do SMF em virtude da flexibilidade exigida do sistema produtivo. Através da Seleção de Partes é possível obter o processamento conjunto de várias partes pertencentes a uma mesma FP, diminuindo assim a diversidade de estilos de partes a ser tratada pelo sistema [16]. Para resolver este problema, foi utilizado o método de Análise por Fluxo de Produção através da formulação matricial proposto por Kusiak [17] denominado de *Cluster Identification*.

2.2 Escalonamento de Partes

Conforme mencionado anteriormente, o escalonamento das partes num JSSP é um problema de difícil solução e é considerado pela teoria da complexidade de classe *NP-Hard* [8][20]. Em um SMF, o objetivo está em seqüenciar os *jobs* num conjunto de máquinas, através de um roteiro pré-estabelecido em função do tempo [4].

O Escalonamento de Partes pode ser definido por diversos objetivos, dependendo da realidade e do foco da indústria. Em um JSSP, o objetivo principal é a minimização do tempo total de produção (*makespan*) [4]. Contudo, pode-se utilizar outros objetivos, tais como: minimizar as trocas de ferramentas, minimizar os tempos de atraso, maximizar a eficiência, entre outros [13]. Inúmeros métodos de otimização foram propostos para a solução do JSSP [4][22][30], 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* [3][5][7]. Já nos métodos aproximativos são destacados os algoritmos iterativos Busca Tabu, as Redes Neurais, os Algoritmos Genéticos, a Têmpera Simulada e o GRASP [9][11][13][12]. Para resolver o problema do escalonamento de partes foi utilizado o algoritmo Busca Tabu em virtude da complexidade do problema e dos diversos trabalhos relacionados.

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 [16]. A utilização das Regras de Despacho para o escalonamento inicial se dá ao fato destas obterem respostas computacionais em tempo polinomial, dado que muitos destes problemas devem ser resolvidos em tempo real. A seguir, algumas Regras de Despacho [16]:

- regra aleatória (RA): 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;
- famílias 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 mais similares (RMS): o seqüenciamento é realizado 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, foi 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 de 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 [9]. Na figura 1, é possível visualizar a lista tabu e as interações entre os componentes de diversificação e intensificação [9][27].

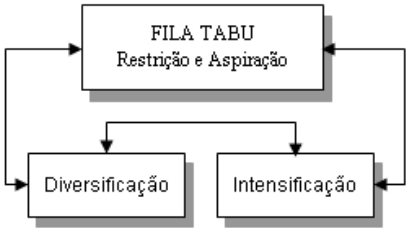

Figura 1: Estratégia de busca da BT.[27]

Para a utilização da BT, é fundamental a definição da função objetivo f do problema em questão. Após

esta definição, é gerada uma solução inicial viável independente (Regra de Despacho). Sendo que, para a geração da solução inicial, é obrigatório que esta faça parte do conjunto de soluções possíveis do espaço amostral. O Algoritmo 1 apresenta o BT modificado para o modelo proposto.

Algorithm 1 Algoritmo Busca Tabu Modificado

while $(niter - melhiter < nbmax)$ **do**
  for *"m"máquina* **do**
    $f' \leftarrow fmelhor$;
    $niter \leftarrow melhiter + 1$;
    $p \leftarrow p^*$;
    $Gerar\ V_e^*$ *de soluções* $(e, p^*)_i\ em\ N_e(e, p^*)$
    $ou\ f((e, p^*)_i) < A(f(e, p^*))$;
    $Atualizar\ Lista\ Tabu\ L\ e\ A(z)$;
    if $f(e', p^*) < fmelhor$ **then**
      $fmelhor \leftarrow f(e', p^*)$
    end if
    $Gerar\ Vp^*$ *de soluções* $(e, p)_i\ em\ N_p(e, p)$
    $ou\ f((e, p)_i) < A(f(e, p))$;
    $Atualizar\ Lista\ Tabu\ L\ e\ A(z)$;
    if $f(e', p') < fmelhor$ **then**
      $fmelhor \leftarrow f(e', p')$
    end if
    $p \leftarrow p'$;
    $e \leftarrow e'$;
  end for
  if $fmelhor < f'$ **then**
    $melhiter \leftarrow niter$
  end if
end while

Conforme algoritmo mostrado acima, é realizada a primeira busca com a troca de posicionamento de todos os lotes dois a dois, aqui denominado de movimento de troca entre dois lotes. Nesta busca, através da máquina selecionada, a vizinhança N_e(e, p*) é formada por todas as seqüências que podem ser obtidas pela troca de posição de dois lotes pertencentes aos turnos de produção. O conjunto Ve* de soluções (e, p*)$_i$ é considerado igual a N_e (h, p*), ou seja, toda a vizinhança é considerada N_e (e, p*)=V_e*. Melhorando a definição de lote, pode-se agora definí-lo como um conjunto de partes que estão encadeadas em um mesmo turno e processadas por uma mesma máquina. Na realização desta primeira busca, a designação das partes às Famílias de Partes permanece constante (p*). Na segunda Busca, uma parte passa a ser processada em outro lote de mesma FP por movimentos de retirada e inserção da parte. Neste caso, a vizinhança N_p(e, p) é formada por todas as seqüências possíveis que podem ser obtidas pela retirada de uma parte pertencente a um lote e por sua inserção em outro lote na mesma máquina. Analogamente, a pesquisa anterior, Vp* é igual a N_p (e,p). A parte poderá ser processada em outro turno e dia da mesma máquina, possibilitando assim a redução do tempo de atraso. Em ambas buscas, é realizado o procedimento de re-escalonamento dos tempos de saída das partes, de forma a calcular as paradas resultantes deste escalonamento e os tempos ociosos dos turnos. A cada iteração um ótimo local é escolhido e a partir dele é gerada uma nova vizinhança. Para evitar ciclos e mínimos locais é implementada uma lista de movimentos reversos proibidos chamada de lista tabu. Caso a iteração obtenha uma melhora na função f, utiliza-se a função critério de aspiração A, que admite o movimento até então 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.

4.1 Geração da Vizinhança

Conforme apresentado no algoritmo Busca Tabu Modificado, o algoritmo apresenta duas gerações de vizinhança. A primeira geração de vizinhança baseia-se na troca de lotes, ou seja, o movimento de troca das posições de dois lotes. Já, a segunda vizinhança é com o movimento de retirada e inserção de partes. No critério para a geração da primeira vizinhança que por sua vez é definida por N_e(e,p*), utiliza-se a troca de todos os lotes possíveis dentro de uma máquina, neste caso, toda a vizinhança é considerada. Nesta geração de vizinhança, as partes permanecem constantes e são representadas por p*. Na segunda geração de vizinhança uma parte pode ser processada em outro lote, através do movimento de retirada e inserção da parte. A vizinhança, representada por N_p(e,p), é gerada através da remoção de uma determinada parte de um lote e inserindo-a em outro lote na mesma máquina.

4.1.1 Movimento de Troca de Lotes

A realização do movimento através da troca de dois lotes reflete na redução do tempo total de paradas e no makespan dado que possa ocorrer união de lotes de mesma FP, eliminando assim o tempo de parada entre os dois lotes. Segundo Gómez [10], o movimento de troca de dois lotes ocorre em três etapas: (i) retirada dos dois lotes da seqüência, (ii) inserção dos lotes em posições inversas em que estavam anteriormente e (iii) redefinição dos lotes. Na etapa de retirada dos dois lotes, a seqüência dos demais lotes não é alterada. Os dois lotes deixam de existir nesta etapa e os tempos de processa-

mento das demais partes ficam na espera da conclusão do movimento.

4.1.2 Movimento de Retirada e Inserção de Parte

O movimento de retirada e inserção de uma parte consiste na retirada de uma parte qualquer de um lote e a sua inserção num lote de mesma FP. Este tipo de movimento pode refletir na redução do tempo de atraso em virtude da inserção da parte poder ocorrer num lote que o antecede. Segundo Gómez [10], o movimento de retirada e inserção de partes possui três etapas: (i) retirada da parte referente um lote origem, (ii) inserção da parte num lote destino e a (iii) redefinição dos lotes. O processo de inserção da parte pode simplesmente entrar num lote destino em que a FP seja igual. Neste caso, leva-se em consideração somente o tempo de processamento do turno e a precedência das operações da parte à ser processada.

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 de planejamento de produção para o SMF proposto por Stecke [26].

5.1 Formulação

A formulação utilizada para a otimização do escalonamento das partes é apresentada a seguir. Sejam:
m = número de máquinas;
n = número de partes;
e = escalonamento;
p = FP;
i = índice para a parte;
j = parte que é processada após a parte i;
k = índice para a máquina;
h= máquina que precede a máquina k;
De_i = data de entrega da parte i;
Dsp_{ik} = data da saída de produção para a parte i na máquina k;
C_{ik} = tempo inicial da parte i na máquina k;
T_{ik} = tempo de processamento da parte i na máquina k;
M = número positivo maior que o tempo total de processamento para n;

$$P_{ik} = \begin{cases} 1, & \text{se a parte i possui operação na máquina k} \\ 0, & \text{caso contrário} \end{cases}$$

$$X_{ijk} = \begin{cases} 1, & \text{se a parte i precede a parte j na máquina k} \\ 0, & \text{caso contrário} \end{cases}$$

$$A_{ihk} = \begin{cases} 1, & \text{se a máquina h precede a k para a parte i} \\ 0, & \text{caso contrário} \end{cases}$$

$Minimizar$

$$f(e,p) = p_1.makespan(e,p) + p_2.atraso(e,p) + p_3.parada(e,p) + p_4.ociosidade(e,p) \quad (5)$$

$$makespan(e,p) = \sum_{k=1}^{m}\sum_{i=1}^{n} makespan_{ik}, \quad \textit{tal que } makespan_{ik} > 0, para \; i = 1,2,\ldots,n \quad k = 1,2,\ldots,m; \quad (6)$$

$$atraso(e,p) = \sum_{k=1}^{m}\sum_{i=1}^{n} (Dsp_{ik} - De_i), \quad \textit{tal que } (Dsp_{ik} - De_i) > 0 \; e \; P_{ik} = 1, \; para \; i = 1,2,\ldots,n \quad k = 1,2,\ldots,m; \quad (7)$$

$$parada(e,p) = \sum_{k=1}^{m}\sum_{i=1}^{n} parada_{ik}, \quad \textit{tal que } parada_{ik} > 0, para \; i = 1,2,\ldots,n \quad k = 1,2,\ldots,m; \quad (8)$$

$$ociosidade(e,p) = \sum_{k=1}^{m}\sum_{i=1}^{n} ociosidade_{ik}, \quad \textit{tal que } ociosidade_{ik} > 0, para \; i = 1,2,\ldots,n \quad k = 1,2,\ldots,m; \quad (9)$$

$$C_{ik} - T_{ik} + M(1 - A_{ihk}) \geq C_{ih}, \quad \textit{tal que } i = 1,2,\ldots,n \quad h,k = 1,2,\ldots,m; \quad (10)$$

$$C_{ik} - C_{ik} + M(1 - X_{ijk}) \geq T_{ik}, \quad \textit{tal que i,j} = 1,2,\ldots,n \quad k = 1,2,\ldots,m; \quad (11)$$

$$C_{ik} \geq 0, \textit{tal que } i = 1,2,\ldots,n \quad k = 1,2,\ldots,m; \quad (12)$$

$$X_{ijk} \in \{0,1\}, \textit{tal que i,j} = 1,2,\ldots,n \quad k = 1,2,\ldots,m; e \tag{13}$$

$$p_1, p_2, p_3, p_4 \geq 0. \tag{14}$$

A função objetivo (5), que busca a minimização, é definida pela variável "e" (escalonamento), que representa a dimensão temporal e a variável "p" (Famílias de Partes), representa a dimensão física para o modelo proposto. É formada por quatro variáveis de decisão que refletem uma estratégia de otimização. A variável de decisão *makespan*(e,p) representa o tempo total de produção, ou seja, é o tempo inicial da primeira parte processada em produção até o tempo final da última parte processada. Já a variável de decisão atraso(e, p) representa o tempo total de atraso. Já o tempo total de paradas é representado pela variável de decisão parada(e,p) e traduz o tempo entre dois lotes de produção.

A variável de decisão ociosidade(e,p) representa o tempo não utilizado nos turnos. A restrição (6) assegura o somatório total do tempo de produção. Já a restrição (7) garante o somatório total do tempo de atraso através da data de entrega da parte e a data de saída desta. A restrição (8) assegura o tempo total de paradas e a restrição (9) assegura o somatório do tempo total ocioso. A restrição (10) garante que a seqüência das operações (rota) para cada parte seja respeitada e, a restrição (11) assegura que cada máquina processa somente uma parte por vez. As restrições (12) e (13) garantem o limite inferior e superior para "i,j e k" e, por fim, a restrição (14) assegura a não-negatividade.

Através dos valores definidos para os pesos (p_1, p_2, p_3 e p_4) de cada variável de decisão, a função objetivo pode refletir as seguintes estratégias de otimização: (i) minimizar o tempo total de produção, (ii) minimizar o tempo total de atraso, (iii) minimizar o tempo total parado e (iv) minimizar o tempo total ocioso [10]. Além das suposições mencionadas no JSSP, também são respeitados os tempos de processamento para cada turno e as datas de entrega.

5.2 Arquitetura

A arquitetura para a Seleção e Escalonamento de Partes está dividida em cinco fases de aplicação onde são abordados os problemas da fase pré-operacional (Seleção de Partes, Alocação de Recursos, Formação de Células de Manufatura) e a fase operacional (Escalonamento, Roteiro da Parte, Controle de Processo) [26]. A arquitetura é apresentada na Figura 2.

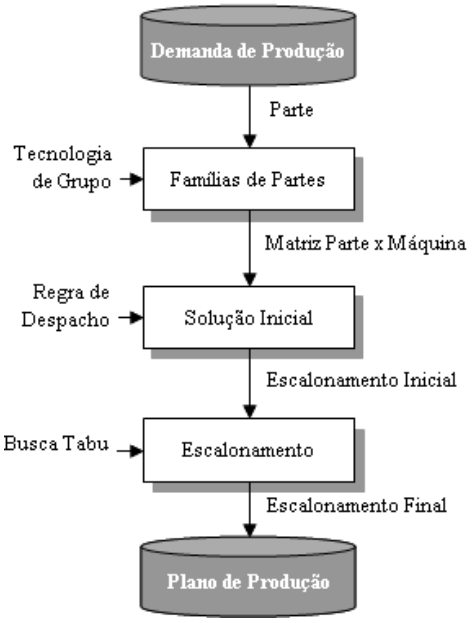

Figura 2: Arquitetura do Modelo.

A primeira fase é responsável por obter a demanda de produção através das informações contidas no banco de dados, sendo recuperadas as informações técnicas da parte e a data de entrega. Já na segunda fase, é realizada através da TG, a geração da matriz partes versus máquinas, aplicando o algoritmo de identificação de agrupamentos. Na terceira fase, é utilizada a Regra de Despacho FP para a geração do escalonamento inicial.

As Regras de Despacho são algoritmos que geram um escalonamento inicial, através de regras relacionadas às partes e máquinas [16]. O escalonamento inicial deve ser viável, ou seja, deve ser uma solução que esteja contida no espaço amostral.

A partir do desenvolvimento 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, este é gravado no plano de produção, possibilitando assim observar seus históricos e compará-los com o escalonamento efetivo.

6 Resultados

A etapa de validação do modelo proposto é realizada através da comparação com trabalhos clássicos que propõem a solução do JSSP [4][13][30]. Para testar o desempenho da arquitetura são utilizadas diversas instâncias do JSSP de modo a realizar a validação em diversos ambientes (número de partes x número de máquinas).

Para a validação, foram utilizadas as instâncias propostas por Muth [23] denominada de FT06, FT10 e FT20, e as instâncias propostas por Lawrence [19] denominadas de LA01, LA06, LA11, LA16, L21, LA29 e LA40. A Figura 3 ilustra a estrutura genérica de uma instância.

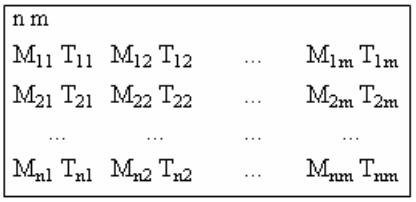

Figura 3: Estrutura Genérica de uma Instância do JSSP.

A instância genérica para o problema do JSSP possui as seguintes definições: (i) a primeira linha define a dimensão da matriz (n x m), onde o "n" representa o número de partes para o problema em questão e o "m" representa o número de máquinas, (ii) as demais linhas representam a rota para cada parte, onde: a variável "M" ilustra a máquina e a variável "T" ilustra o tempo de processamento, (iii) as colunas ímpares representam o número da máquina que a parte deverá ser processada e (iv) as colunas pares representam o tempo de processamento da parte em cada máquina.

6.1 Trabalhos selecionados

Através dos trabalhos que propõem a resolução do JSSP [4][13][30], foram selecionados alguns autores em função do tipo de algoritmo utilizado e da sua importância na evolução do estado da arte. Cabe lembrar que, em função da validação do modelo, levou-se em consideração o fato do autor ter resolvido as instâncias selecionadas. A Tabela 1 mostra os autores selecionados e o algoritmo utilizado.

Tabela 1: Autores selecionados para a validação do modelo

Autor	Algoritmo	Ano
Nowicki [24]	Busca Tabu	1996
Wang [29]	Genético híbrido e Têmpera simulada	2001
Aiex [2]	GRASP	2003
Gonçalves [11]	Algoritmos genéticos híbridos	2005

6.2 Configuração

O modelo foi implementado em Delphi 7 e seus experimentos executados em um *hardware Athlon XP 2500+ 512 RAM* com sistema operacional *Windows XP Pro*. Para a configuração do algoritmo BT modificado foi atribuído o valor de 100 para o número máximo de iterações sem obter melhora na função objetivo (nbmax) e, para o tamanho da lista tabu, foi atribuído 15. Em virtude dos autores selecionados utilizarem o *makespan* para os seus experimentos, os pesos (p_1, p_2, p_3 e p_4) para a função objetivo foram fixados em 100, 1, 1 e 1, respectivamente, de modo a privilegiar a variável de decisão *makespan*.

É importante ressaltar que o modelo proposto, além de otimizar o *makespan*, possui uma arquitetura que tem como objetivo o gerenciamento entre as datas de entrega (minimização do atraso) e a produtividade (minimização das paradas de produção). As demais abordagens supracitadas otimizam somente o *makespan*, desconsiderando as datas de entrega e a produtividade do sistema.

Como as instâncias não possuem as datas de entrega para a parte, e o modelo proposto às utiliza como estratégia de otimização, foram fixadas todas as partes com a mesma data de entrega.

6.3 Validação

Os resultados da validação são mostrados na Tabela 2. Nela é apresentada a instância do problema, a dimensão do problema (número de partes x número de máquinas), a solução ótima conhecida (SOC), a solução obtida pelo modelo proposto (MP) e a solução dos autores selecionados.

Tabela 2: Estratégia que privilegia a produtividade

Instância	NxM	SOC	MP	Gonçalves	Aiex	Wang	Nowicki
FT06	6x6	55	55	55	55	55	55
FT10	10x10	930	930	930	930	930	930
FT20	20x5	1165	1165	1165	1165	1165	1165
LA01	10x5	666	666	666	666	666	666
LA06	15x5	926	926	926	926	926	926
LA11	20x5	1222	1222	1222	1222	1222	1222
LA16	10x10	945	945	945	945	945	945
LA21	15x10	1046	1047	1046	1057	1058	1047
LA29	20x10	1157	1160	1196	1203		1160
LA40	15x15	1222	1229	1241	1244		1229

Através da Tabela 2 é possível fazer um comparativo entre o modelo proposto e a solução ótima conhecida. Nota-se que o *makespan* com a maior diferença em relação à SOC diferencia-se em 7 unidades de tempo (LA40), equivalendo a uma diferença de 0,57%. De maneira a validar o modelo, os resultados esperados atingiram o objetivo, ou seja, em muitos casos alcançando a SOC e, em outros casos, aproximando-se.

As instâncias que tiveram diferenças em relação à SOC, sofreram influência da função objetivo do modelo proposto. Ao avaliar a função objetivo, considerou-se

para este problema do *makespan*, um peso significativamente maior para a variável de decisão *makespan* do que as demais variáveis, direcionando assim a BT para a otimização do *makespan*.

Observa-se também que, após a validação, além de contemplar o *makespan*, o modelo proposto realizou o gerenciamento entre as datas de entrega (atraso) e a produtividade (velocidade de produção) do sistema de produção, mesmo privilegiando a variável de decisão *makespan*. Já as demais abordagens consideram somente o *makespan* e desconsideram qualquer outra variável em questão, como por exemplo as datas de entrega, tempo parado em produção, tempo ocioso, entre outros.

A capacidade do modelo em gerenciar políticas diferentes, e não somente o *makespan*, ocorre em virtude das variáveis de decisão que compõem a função objetivo e que abordam estratégias de otimização do escalonamento, tais como: (i) *makespan*, (ii) tempo total em atraso, (iii) tempo total parado em produção e (iv) o tempo total ocioso. A seguir são apresentadas algumas estratégias na utilização do modelo proposto e que não são consideradas nas demais abordagens.

6.3.1 Estratégia que privilegia a otimização das paradas de produção

Dado o privilégio da variável de decisão parada (peso p_3=100), objetiva-se a produtividade e, conseqüentemente, a velocidade de produção. A Tabela 3 apresenta o resultado dos experimentos.

Tabela 3: Estratégia que privilegia a produtividade

Instância	SOC	MP	Diferença %	Partes em atraso
FT06	55	55	0,00	2
FT10	930	930	0,00	3
FT20	1165	1165	0,00	4
LA01	666	666	0,00	2
LA06	926	926	0,00	3
LA11	1222	1222	0,00	3
LA16	945	945	0,00	4
LA21	1046	1047	0,09	5
LA29	1157	1160	0,26	5
LA40	1222	1229	0,57	4

Através dos experimentos que visam a otimização da produtividade, presencia-se um conflito em relação às datas de entrega (atraso) e as paradas de produção, ou seja, quanto menor é o tempo de paradas de produção (maior magnitude), menor é a magnitude da variável de decisão atraso no direcionamento da BT. Constatou-se que estas variáveis são independentes e, ao privilegiar uma, causa influência negativa na outra. Este comportamento ocorre, pois a variável de decisão atraso perde a influência na função objetivo, privilegiando assim a escolha de movimento na vizinhança (algoritmo BT) em função da variável de decisão parada e o fato também destas variáveis serem independentes.

Outro aspecto importante neste experimento consiste na constatação da dependência de duas variáveis de decisão, ou seja, ao reduzir o tempo parado, ocorreu influência na redução do *makespan*. A estratégia que privilegia a redução do tempo parado consistiu com os mesmos valores obtidos pelo *makespan* calculado na etapa de validação do modelo.

6.3.2 Estratégia que privilegia a otimização das datas de entrega

Dado o privilégio da variável de decisão atraso (peso p_2=100), objetiva-se a entrega pontual das partes, desconsiderando assim as demais estratégias de otimização. A Tabela 4 apresenta os resultados dos experimentos considerando a entrega pontual das partes.

Tabela 4: Estratégia que privilegia a entrega pontual

Instância	SOC	MP	Diferença %	Partes em atraso
FT06	55	60	9,09	0
FT10	930	1003	7,84	0
FT20	1165	1243	6,69	0
LA01	666	722	8,40	0
LA06	926	1037	11,98	0
LA11	1222	1313	7,45	0
LA16	945	1020	7,94	0
LA21	1046	1145	9,46	0
LA29	1157	1259	8,82	0
LA40	1222	1356	10,96	0

A partir da Tabela 4, observa-se que o modelo obteve a entrega pontual de todas partes. Porém, o *makespan* sofreu aumento significativo, conseqüência do aumento das paradas de produção. Dessa forma, constatou-se o conflito existente entre respeitar as datas de entrega e manter a produtividade.

6.3.3 Estratégia não-tendenciosa

A solução não-tendenciosa é aquela em que o peso não privilegie nenhuma das variáveis de decisão da função objetivo. É obtida através da proporção das variáveis de decisão dada a execução de 100 experimentos variando os pesos (p_1, p_2, p_3, p_4), seguindo uma distribuição normal com intervalo (0, 100].

Com a estratégia não-tendenciosa, ocorreu a redução do conflito entre as variáveis de decisão *makespan* e parada, ou seja, a relação entre a produtividade e as datas de entrega. A Tabela 5 ilustra os resultados obtidos.

Tabela 5: Estratégia não-tendenciosa

Instância	SOC	MP	Diferença %	Partes em atraso
FT06	55	57	3,63	0
FT10	930	961	3,33	2
FT20	1165	1198	2,83	4
LA01	666	684	2,70	1
LA06	926	951	2,75	2
LA11	1222	1267	3,68	3
LA16	945	976	3,25	3
LA21	1046	1086	3,28	4
LA29	1157	1206	4,23	3
LA40	1222	1272	4,09	2

Esta estratégia pode ser utilizada como política de escalonamento, dado que obteve uma relação consistente entre as variáveis de decisão *makespan* e atraso, sendo observado que o *makespan* aproximou-se da SOC e poucas partes foram entregues em atraso. Ao ser utilizada uma estratégia não-tendenciosa, foi possível verificar a redução do *makespan* e não houveram diferenças significativas em relação ao número de partes em atraso ao ser comparado com a estratégia que privilegia a entrega pontual.

6.4 Conclusões

Este trabalho teve como objetivo apresentar um modelo aplicado ao JSSP considerando um SMF. O modelo implementado está baseado numa função objetivo que reflete, através das variáveis de decisão e seus pesos respectivos, estratégias de escalonamento. A arquitetura do modelo é responsável por tratar os problemas de Seleção de Partes e do Escalonamento de Partes. Foram utilizadas na validação, diversas instâncias para o problema do JSSP e conclui-se que o modelo proposto pode ser considerado mais uma abordagem para o escalonamento de produção, dado que as outras abordagens citadas na etapa de validação não gerenciam o conflito entre as datas de entrega (atraso) e a produtividade do sistema de produção (velocidade de produção). Como extensão futura deste trabalho, sugere-se a calibração dos pesos da função objetivo, validação utilizando instâncias de grande escala, aprimoramento dos métodos de geração e escolha de vizinhança e, por fim, propor instâncias para o JSSP que considerem as datas de entrega e paradas de produção dado que as instâncias utilizadas foram adaptadas para a validação do modelo.

Referências

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

[2] Aiex, R., Binato, S., and Resende, M. G. C. Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing*, 29(4):393–430, 2003.

[3] Balas, E., Johnson, E. L., and Korte, B. Disjunctive programming. In: Hammer, P.L. *Discrete Optimisation II*, pages 49–55, 1979.

[4] Blazewicz, J., Domschke, W., and Pesch, E. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93:1–33, 1996.

[5] Bowman, E. H. The schedule-sequencing problem. *Operations Research*, 7:621–624, 1959.

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

[7] Fisher, M. A dual algorithm for the one-machine scheduling problem. *Math Programming*, 11:229–251, 1976.

[8] Garey, M. R., Johnson, D. S., and Sethi, R. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1:117–129, 1976.

[9] Glover, F. and Laguna, M. Tabu Search. *Kluwer Academic Publishers*, page 382, 1997.

[10] Gomez, A. 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*, 1996.

[11] Gonçalves, J. F., Mendes, M. J. J., and Resende, M. G. C. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167:77–95, 2005.

[12] Hurink, J. and Knust, S. Tabu Search algorithms for job-shop problems with a single transport robot. *European Journal of Operational Research*, 162:99–111, 2004.

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

[14] Jeon, G. and Leep, H. R. Forming part families by using genetic algorithm and designing machine cells under demand changes. *Computers and Operations Research*, 33:263–283, 2006.

[15] Jha, N. K. Handbook of Flexible Manufacturing Systems. *Academic Press Limited*, page 328, 1991.

[16] Kusiak, A. Intelligent Design and Manufacturing. *John Wiley and Sons, Inc.*, page 753, 1992.

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

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

[19] Lawrence, S. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. *Graduate school of industrial administration, Carnegie Mellon University: Pittsburgh*, 1984.

[20] Leung, J. Y.-T. Handbook of Scheduling: Algorithms, Models, and Performance Analysis. *CRC Press*, page 1224, 2004.

[21] Lorini, F. J. Tecnologia de Grupo e organização da manufatura. *Editora da UFSC*, page 105, 1993.

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

[23] Muth, J. F. and Thompson, G. L. Industrial Scheduling. *Englewood Cliffs, Prentice Hall*, pages 225–251, 1963.

[24] Nowicki, E. and Smutnicki, C. A Fast Taboo Search Algorithm for the Job Shop. *Management Science*, 42:797–813, 1996.

[25] Pezzella, F. and Merelli, E. A tabu search method guided by shifting bottleneck for job shop scheduling problem. *European Journal of Operational Research*, 120:297–310, 2000.

[26] 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.

[27] Viana, G. V. R. Meta-heurísticas e programação paralela em otimização combinatória. *EUFC*, page 250, 1998.

[28] Wang, J. and Kusiak, A. Computational Intelligence in Design and Manufacturing Handbook. *CRC Press*, page 800, 2001.

[29] Wang, L. and Zheng, D. An effective hybrid optimization strategy for job-shop scheduling problems. *Computers and Operations Research*, 28:585–596, 2001.

[30] Zoghby, J., Barnes, W. L., and Hasenbein, J. J. Modeling the Reentrant job shop problem with setups for metaheuristic searches. *European Journal of Operational Research*, 167:336–348, 2004.