Antonio Gabriel Rodrigues

Análise do Comportamento dos Tempos de Produção em um Sistema de Manufatura Flexível em um Problema de Escalonamento em um Job Shop: Abordagem utilizando Conceito de Caminho Crítico

São Leopoldo

2007

Antonio Gabriel Rodrigues

Análise do Comportamento dos Tempos de Produção em um Sistema de Manufatura Flexível em um Problema de Escalonamento em um Job Shop: Abordagem utilizando Conceito de Caminho Crítico

Dissertação submetida a avaliação como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada

Orientador:
Arthur Tórgo Gómez

UNIVERSIDADE DO VALE DO RIO DOS SINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

São Leopoldo

2007

AGRADECIMENTOS

Foi dito a mim um ano atrás que “fazer uma dissertação é um trabalho solitário”. Recordando da maioria das horas passadas no desenvolvimento dela, poderia dizer que essa frase é válida, mas, sei que isso não é de todo verdade. No decorrer desse tempo de desenvolvimento, recebi ajuda de entidades e amizade de uma série de pessoas, às quais me esforço para agradecer a seguir.

Agradeço ao CNPq por ter financiado o meu mestrado, tornando esse possível.

Agradeço aos meus amigos, fonte das minhas alegrias nesse período, cujas inúmeras poesias e pensamentos que recebi por e-mail, referenciam como “o maior tesouro”. Fato. Sem essas amizades, a vida não tem brilho e o sorriso não é possível. Aos amigos que fazem parte do PIPCA (alguns já não tão ligados à faculdade), os quais nomeio alfabeticamente: Denise Penchmann, Fábio Carvalho, Fábio Mierlo, Júlio Ferronato, Priscila Severino, Rejane Weissheimer, Rossana Queiroz, Tana Malacarne. Obrigado pela ajuda, pelas palavras alegres e pela paciência que disporam nesse período. Aos irmãos Toss Hoffmann, Cléber e Leandro, e a Laura Jung, agradeço pela amizade, pelos momentos divertidos e pelos passeios que pudemos realizar juntos. A Srta Bárbara Quadros, pela boa companhia e pelas risadas. Ao Sr. Gilberto Irajá Müller, obrigado pelo apoio, pelas discussões e pelas sugestões dadas no desenvolvimento desse trabalho. Aos meus colegas de Mestrado, agradeço pela amizade e pelo companheirismo nas disciplinas cursadas, Em especial agradeço a Etiene Lazzeris Simas (por me acompanhar nos risos nervosos e na re-engenharia das respostas do livro “well writen” do Çinlar) e Leandro Motta Barros (pelas divertidas discussões no almoço).

Aos professores do PIPCA, nossos mestres, que nesta trajetória passaram o conhecimento necessário para que este trabalho pudesse ser desenvolvido, meu profundo agradecimento. Agradeço ao Prof Leonardo Chiwiacowsky, José Vicente dos Santos e João Mocellin por avaliarem o meu trabalho, sugerindo as correções para o melhoramento deste. Agradeço especialmente ao orientador e amigo Arthur Gómez, pelos anos produtivos de pesquisa e de mestrado.

Agradeço a minha família, que confiou em minhas capacidades e tanto me apoiou. Ao meu pai Flávio, minha mãe Terezinha e meu irmão João, obrigado pela formação e por todo o

apoio que me deram. O que de bom tenho, devo a vocês. Aos meus tios Irani, José, Djalma e Anita, com os quais tive o imenso priviléio de conviver sob o mesmo teto, desde o início da graduação. Sem a sua boa vontade, talvez não estivesse redigindo essas palavras agora. Aos meus primos, todos muito obrigado pela convivência e pelos momentos divertidos juntos, sejam estes conversando, estourando bombinhas para assustar os vizinhos ou vendo algum tiroteio. A minha grande amiga, a qual considero minha irmã, Tati Evers e ao meu "cunhado" André, muito obrigado pelas horas de RPG, pelas conversas divertidas, eventos de *anime*, pelas palavras de coragem e por terem me extendido a mão nas horas que julguei mais precisar.

Enfim, agradeço a Deus pela ajuda nas horas difíceis, pela luz de esperança que me fez ver, através de pessoas e coisas. Agradeço pelas oportunidades, pelas dificuldades que me fizeram aprender tanto sobre mim e sobre os outros, e por ser tão bom permitindo que eu pudesse viver e partilhar momentos da minha vida com as pessoas tão especiais que me rodeiam. Obrigado.

Que existe mais, senão afirmar a multiplicidade do real?

A igual probabilidade dos eventos impossíveis?

A eterna troca de tudo em tudo?

A única realidade absoluta?

Seres se traduzem.

Tudo pode ser metáfora de alguma outra coisa ou de coisa alguma.

Tudo irremediavelmente metamorfose!

(A Multiplicidade do real - *Paulo Leminski*)

RESUMO

Neste trabalho é abordado o Problema de Escalonamento em um *Job Shop*, considerando restrições de datas de entrega, turnos de produção e tempo de *setup* entre operações. Considera-se um ambiente de Sistema de Manufatura Flexível, que dado ao alto nível de automação, permite a previsibilidade dos processos de carregamento dos recursos à área de processamento. O problema foi modelado através de uma Função Objetivo f_π composta de três variáveis de decisão. A importância da contribuição de cada variável para o valor de f_π é gerida pela atribuição de valores aos pesos associados às variáveis. Na abordagem proposta, são utilizadas técnicas de Tecnologia de Grupo e Busca Tabu. O modelo implementado é uma modificação da técnica i-TSAB, proposta por Nowicki e Smutnicki, a qual apresenta bons resultados no tratamento do Problema de Escalonamento em um *Job Shop* (PEJS) clássico. A consideração das restrições adicionais ao PEJS aumenta a complexidade do modelo implementado, porém, deixa o problema mais próximo da realidade. O modelo foi validado com problemas-teste de *bechmark* modificadas para comportarem os dados das restrições adicionais. Foi realizada uma análise do comportamento das variáveis de decisão frente a definição de políticas de minimização dos tempos de produção. Nesta análise pode-se observar que a minimização do *makespan* acarreta na redução do atraso, sendo que o *Setup* não sofre alterações significativas. O modelo apresenta resultados aceitáveis quando as variáveis são minimizadas uma de cada vez. A utilização do conceito de caminho crítico reduz a vizinhança de busca para a minimização do tempo total de produção, mas prejudica a minimização de variáveis que não dependam exclusivamente dele.

Palavras-chave: Problema de Escalonamento em *Job Shop*. Sistema de Manufatura Flexível. Meta-heurística. Tecnologia de Grupo. Busca Tabu.

ABSTRACT

In this work the Job Shop Scheduling Problem is studied, considering due dates, production turns and tooling constraints. This problem is applied in a Flexible Manufacturing System, which possesses high degree of automation, allowing previsibility in the processes of loading and unloading jobs on the machines. The problem is modeled through a objective function f_π composed by three weighted decision variables. The importance of each variable in the f_π final value is managed through assignment of values to the weights of these variables. In the proposed approach, it was used Group Technology and Tabu Search techniques. The implemented model is a modification of the *i*-TSAB technique, proposed by Nowicki and Smutniki. The consideration of adicional constraints in the Job Shop Scheduling Problem increases the complexity of the implementation, otherwise, makes the problem closer to the industrial reality. The model was validated using benchmark instances, in which the data from the addional constraints were added. It was made a annalysis of the behavior of the decision varibles considering the definition of three minimization policies. In this analysis it was noticed that the minimization ok Makespan contributes to the Tardiness reducion, while Setup times is not improved significantly. The model shows good results when each variable is minimized at time. The use of the critical path concept reduces the neighborhood search size, contributing for the minimization of Makespan, but in other hand, harm variables which do not exclusivelly deppend of the critical path.

Key-words: Job Shop Scheduling Problem. Flexible Manufacturing System. Tabu Search. Group Technology. Metaheuristic.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

TG (Tecnologia de Grupo), p. 13

FPs (Famílias de Partes), p. 13

CMs (Células de Manufatura), p. 13

CC (Classificação e Codificação), p. 18

OIR (*Organization for Industrial Research*), p. 21

AFP (Análise de Fluxo de Produção), p. 21

SLCA (*Single Linkage Cluster Algorithm*), p. 23

ALC *Average Linkage Cluster*, p. 24

ROC (*Rank Order Cluster*), p. 26

ABE (Algoritmo *Bond Energy*), p. 26

AIG (Algoritmo de Identificação de Grupos), p. 27

AIGE (Algoritmo de Identificação de Grupos Extendido), p. 28

SMF (Sistema de Manufatura Flexível), p. 34

CMF (Célula Flexível de Manufatura), p. 35

NC ((Numerical Control)), p. 35

CNC (*computer Numerial Control*), p. 36

CIM (*Computer Integrated Manufacturing*), p. 36

DNC (*Distributed Numerical Control*), p. 41

STMM (Sistema de Transporte e Manuseio de Materiais), p. 41

AVG (*Automated Guided Vehicle*), p. 42

PSP (Problema de Seleção de Partes), p. 47

FF (Família de Ferramentas), p. 47

SUMÁRIO

1 INTRODUÇÃO

Um dos problemas clássicos da área de Otimização Combinatória é o Problema de Escalonamento em um *Job Shop* (PEJS), que consiste em definir, para uma série de produtos, sua seqüência de processamento em uma planta fabril, de forma que o tempo total de produção seja o menor possível. Estudado desde a década de 60, esse problema é considerado bastante complexo, sendo que alguns casos desse problema levaram cerca de 2 décadas para terem o resultado ótimo atingido. Para algumas instâncias de grande porte (100 produtos e 20 máquinas, por exemplo), o resultado ótimo ainda não pôde ser comprovado.

O PEJS advém da área de Manufatura, sendo comum encontrar variações que retratem particularidades de um sistema produtivo (a exemplo, PEJS aplicado a um ambiente com uma máquina apenas, considerando movimentos de carregamento por robôs, entre outros) ou que tenham foco em um objetivo diferente de minimizar o tempo total de produção (a exemplo, minimizar datas de entrega, minimizar o número de paradas de uma máquina etc). O PEJS pode ser abordado estocasticamente (quando trabalha-se com distribuições de probabilidade para a chegada dos pedidos de produtos e para os tempos de processamento, carregamento e deslocamento dentro da planta fabril) ou deterministicamente (assumindo que os tempos de processamentos dos produtos são conhecidos, assim como os pedidos desses, os tempos de carregamento nas máquinas e deslocamento dentro da fábrica).

Dado a alta complexidade do PEJS, os métodos exatos para a resolução de problemas de otimização combinatória demonstraram ser ineficientes e computacionalmente inviáveis, dada a enorme quantidade de tempo e recursos computacionais (como memória e processamento) que exigem. Assim sendo, os estudos que envolvem heurísticas e meta-heurísticas ganharam força. Mesmo não garantindo um resultado ótimo, esses métodos mostraram conseguir bons resultados com baixos custos computacionais. Atualmente, o PEJS serve como problema de *benchmark* para novas meta-heurísticas, sendo estudado por diversas áreas, como engenharia e computação.

Este trabalho trata o Problema de Escalonamento em um *Job Shop* considerando

restrições de turnos de produção, datas de entrega e troca de ferramentas. Considera-se o problema aplicado a um Sistema de Manufatura Flexível (SMF), que, dado o alto grau de automação, confere o determinismo necessário para uma abordagem discreta desse problema. Aliado a esse fator, um estudo do estado da arte mostra que a implementação de SMFs passa a ser um fator de competitividade das empresas, dada a importância das vantagens oferecidas por esse tipo de sistema.

A desafiadora complexidade do problema motiva a proposição de um modelo computacional que emprega técnicas de Tecnologia de Grupo e a meta-heurística Busca Tabu, adaptando o trabalho de Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 2002) para o PEJS tratado. A consideração das variáveis de datas de entrega e troca de ferramentas deixa o problema mais complexo, o que pode penalizar o desempenho do modelo, porém deixa o problema mais próximo da realidade industrial.

No capítulo 2 são descritos conceitos importantes no desenvolvimento deste trabalho. É feita uma descrição dos componentes básicos que definem um Sistemas de Manufatura. É feita a definição de Complexidade Computacional e dos tipos e classes de problemas. A seguir é apresentado o conceito de grafo, seus tipos e sua notação. A definição de heurísticas e meta-heurísticas é feita no final do capítulo.

No capítulo 3 é mostrada a definição da Tecnologia de Grupo, conceito que torna possível a projeção das Células de Manufatura e dos Sistemas de Manufatura Flexíveis. A Tecnologia de Grupo é uma filosofia aplicada em diversas áreas do conhecimento, sendo neste trabalho descritas aplicações no ambiente industrial. Pela Tecnologia de Grupo formula-se o Problema de Seleção de Partes, o qual pode ser tratado segundo três tipos de métodos tradicionais: Inspeção Visual, classificação e Codificação e Análise de Fluxo de Produção. Cada um destes métodos conta com uma série de ferramentas disponíveis, as quais são descritas no decorrer do capítulo.

No capítulo 4 é abordado o resultado da aplicação da Tecnologia de Grupo no ambiente industrial, somado aos avanços tecnológicos na área de automação: o Sistema de Manufatura Flexível. Esse sistema é baseado na Manufatura Celular, com um alto nível de automação, fazendo intensivo uso de sistemas de informação. Sistemas de Manufatura Flexíveis possuem alta capacidade de processamento distribuído e podem lidar com uma série de imprevistos no ambiente fabril (KAIGHOBADI; VENKATESH, 1993). Neste capítulo é apresentada a definição, componentes e histórico desse tipo de sistema produtivo.

No capítulo 5 são descritos os Problemas de Seleção de Partes e Escalonamento em um *Job Shop*. É apresentada uma revisão do estado da arte para o PEJS, abordando principalmente

as meta-heurísticas utilizadas no seu tratamento. Logo a seguir é feita a definição do Problema de Escalonamento em *Job Shop* considerando a minimização de três objetivos: tempo total de produção, tempo de troca de ferramentas e atraso dos produtos. É apresentada nesse item a modelagem do problema por grafos dirigidos. É também apresentada a modelagem proposta por Hertz e Widmer (HERTZ; WIDMER, 1996), que considera a troca de ferramentas entre operações.

No capítulo 6 é feita a descrição das técnicas utilizadas no tratamento do PEM. É feita uma descrição dos trabalhos de Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 2002) e Watson, Bowe e Whitley (WATSON; HOWE; WHITLEY, 2006), que apresentam bons resultados no tratamento do PEJS e serviram de base para o modelo proposto neste trabalho.

No capítulo 7 são feitas as descrições da arquitetura do modelo computacional implementado e a estrutura dos problemas-teste do PEM. O modelo é composto de quatro módulos encarregados da adaptação de problemas-teste de *benchmark* ao PEM, geração de Famílias de Operações, geração de uma solução inicial e geração do escalonamento que leve em consideração as restrições do PEM.

A validação do modelo e os experimentos realizados são apresentados no capítulo 8. A validação do modelo é feita em duas etapas: validação da geração de Famílias de Operações e validação do escalonamento final, considerando o tempo total de produção e o tempo de *setup*. A primeira etapa de validação é feita utilizando um problema-teste de Tang e Denardo (GÓMEZ, 1996). Na segunda etapa de validação, comparam-se os resultados gerados pela ferramenta implementada com o trabalho de Hertz e Widmer (HERTZ; WIDMER, 1996) e com o resultado ótimo de problemas-teste de *benchmark*. Os experimentos realizados tem por objetivo analisar o comportamento do modelo frente a políticas de minimização de *makespan*, atraso e tempo de *setup*.

Por fim, no capítulo 9 são apresentadas as conclusões deste trabalho e as sugestões de trabalhos futuros.

2 CONCEITOS

Neste capítulo são apresentados os conceitos mais importantes no contexto deste trabalho. Inicialmente são apresentados os conceitos de sistemas de manufatura, seus componentes principais e sua classificação, seguido do conceito de complexidade computacional. Os conceitos de heurísticas e meta-heurísticas, que atualmente são as ferramentas mais utilizadas no tratamento de problemas complexos, são apresentados ao final do capítulo.

2.1 Sistemas de Manufatura

Groover (GROOVER, 2001) define como Sistema de Manufatura uma coleção integrada de equipamentos e recursos humanos, cuja função é executar uma ou mais operações de montagem ou processamento em um conjunto de partes ou matéria-prima. Um sistema de manufatura pode ser visto como uma parte do Sistema Produtivo, que engloba Controle de Qualidade, Tecnologias de Automação, Tecnologias de Manipulação de materiais, entre outras.

Um Sistema de Manufatura pode possuir vários componentes, os quais, segundo Groover, são dos seguintes tipos:

- Máquinas de produção, ferramental, paletes e equipamentos relacionados. As máquinas de um sistema de manufatura podem ser classificadas quanto ao nível de automatização: (*i*) manualmente operadas, (*ii*) semi-automatizadas, que contam com um programa para a execução de um processo, mas requerem que um operário faça o carregamento das partes, o descarregamento e a troca de programação; (*iii*) completamente automatizadas, que requerem pouca assistência humana, pois dispõem de dispositivos de armazenamento de partes processadas.

- Sistema de Manipulação de Material. Esse sistema tem funções de: (*i*) carregar, posicionar e descarregar partes e produtos: partes que necessitam ser processadas podem ser armazenadas temporariamente em paletes perto das estações de trabalho, até o momento que são carregadas e posicionadas na estação. Após o processamento, as partes

são descarregadas e seguem para o próximo processo. São exemplos de equipamentos que realizam essas tarefas os robôs industriais, alimentadores e trocadores automáticos de paletes. (*ii*) transporte de materiais entre estações de trabalho: depende da estrutura da planta e do tipo de produto. Em alguns casos essa função pode ser desempenhada por operários, assistidos por guindastes. Se a rota das partes é fixa, podem ser utilizadas esteiras industriais. Para partes com diversas rotas, o sistema de transporte deve possuir flexibilidade para alterar facilmente as rotas, podendo ser utilizados carros teleguiados. Paletas de fixação das partes para manipulação pelo sistema de transporte e manuseio devem levar em conta os atributos geométricos das partes. A maioria dos sistemas de manipulação de materiais permitem armazenamento temporário entre as estações.

- Sistema computacional. Esse sistema é necessário para a coordenação de máquinas automáticas e semi-automáticas. As principais funções do Sistema Computacional são: carregamento de programas para as máquinas, controle do Sistema de Manipulação de Materiais, escalonamento da produção, diagnóstico de falhas, controle de qualidade e gerenciamento das operações.

- Recursos humanos. Os operários executam as tarefas que agregam valor aos produtos acabados. Essas tarefas podem ser classificadas como trabalho direto (quando os operários processam manualmente as partes) ou indireto (quando operam máquinas para o processamento). Mesmo em sistemas com alto grau de automação, recursos humanos são necessários para carregamento e descarregamento de partes, controle de estações de trabalho, manutenção e suporte.

Além dos componentes, o Sistema de Manufatura pode ser classificado por:

- Tipos de operações executadas: podem ser realizadas operações de montagem (onde vários componentes são combinados para formar um produto) ou operações de processamento de unidades individuais.

- Número de estações de trabalho e *layout* do sistema: podem ser classificados em sistemas de: uma única estação, múltiplas estações com roteamento variável e múltiplas estações com roteamento fixo.

- Nível de Automação: os sistemas podem ser manuais, semi-automatizados ou completamente automatizados. Também é considerado o número de estações de trabalho.

- Variedade de produtos: caracteriza a capacidade de lidar com diferentes tipos de produtos. A variação de produtos pode ser classificada em (*i*) modelo único: todos os produtos

são idênticos. (*ii*) lote de modelos: diferentes modelos de partes são reunidos em lotes de produção, para reduzir o tempo de troca de ferramentas (*setup*). Esse tempo é requerido porque as diferenças entre as partes são maiores que a capacidade de processamento simultâneo das máquinas, logo, ocorre a troca de ferramentas, (*iii*) modelo misto: diferentes tipos de partes são produzidos, mas as máquinas do sistema têm a capacidade de processar essa variedade sem a necessidade de *setup*.

2.2 Complexidade Computacional e Classes de Problemas

Algoritmo pode ser definido como uma seqüência bem definida de passos que, dada uma determinada entrada, produz uma determinada saída (CORMEN, 2001). Uma das formas de mensurar a eficiência de um algoritmo é a análise dos recursos computacionais que este consome. Um dos fatores que são mais considerados é o tempo que o algoritmo leva para executar a tarefa determinada. A função de tempo de execução dado uma determinada entrada é a medida de complexidade do algoritmo (VIANA, 1998).

2.2.1 Complexidade de um Algoritmo

Para realizar a análise de complexidade de um algoritmo, considera-se um modelo genérico de computador que possui apenas um processador, a Máquina de Acesso Aleatório (*Random-Access Machine - RAM*). Neste modelo as instruções são executadas uma após a outra, sem concorrência de operações. O tempo de execução é o número de instruções primitivas executadas pelo algoritmo. A velocidade de execução de instruções varia de acordo com o tipo de processador e forma de implementação. Costuma-se considerar então o modelo RAM de computador para definir uma unidade de tempo padrão (CORMEN, 2001).

A análise da complexidade de um algoritmo consiste em calcular o número de instruções que este executará para um determinado tamanho de entrada. A partir do número de instruções pode-se fazer uma estimativa de tempo de execução, considerando a RAM. A função de tempo, dado o tamanho de entrada, serve como parâmetro para a comparação de diferentes algoritmos e mensuração da eficiência computacional. Geralmente, a eficiência é analisada no pior caso, onde o tamanho da entrada tende a ser um número relativamente grande. A análise do pior caso fornece um limite superior para o tempo de execução de um algoritmo, enquanto que a análise do melhor caso fornece um limite inferior (CORMEN, 2001) (VIANA, 1998).

A complexidade $T(n) = an^2 + bn + c$ de um determinado algoritmo informa a taxa de

crescimento do seu tempo de execução. Considera-se o elemento de maior grau de $T(n)$, que é n^2, dado que para um n suficientemente grande as outras parcelas de $T(n)$ tornam-se pouco significativas. Assim sendo, a complexidade desse algoritmo, no pior caso, é n^2. A notação mais utilizada para representar complexidade é a notação assintótica O, que define um limite superior para a complexidade de um algoritmo (CORMEN, 2001).

2.2.2 Tipos de Problemas

A classificação dos problemas permite definir o tipo de resposta que o algoritmo que o resolve busca retornar. Um problema pode ser classificado como (VIANA, 1998):

- Problema de Decisão: para este tipo de problema os resultados possíveis são "sim" ou "não". Se um problema pode ser representado através de uma expressão E na forma Normal Conjuntiva (uma expressão lógica composta de conjunções e disjunções), E é *satisfatível* se existe uma atribuição de valores às suas váriáveis que torne E "verdadeira". Se tal atribuição existe, a resposta "sim" é encontrada; caso contrário, deve ser provado, para todos os casos, que E é "falso".

- Problema de Localização: este tipo de problema consiste em encontrar (localizar) uma resposta que satisfaz o problema de decisão associado, ou demonstrar que tal resposta não existe. Essa resposta pode ser um conjunto de valores associados às parcelas de E que a torne "verdadeira".

- Problema de Otimização: encontrar uma resposta ao problema definido que satisfaça algum critério de otimização, ou seja, dentre todas as respostas para o problema, deve-se encontrar a menor (minimização) ou a maior (maximização) delas.

2.2.3 Classes de Problemas

Algoritmos que possuem complexidade $O(n)$ tem um tempo de execução assintoticamente menor que algoritmos $O(n^2)$. Em geral, algoritmos que são executados em tempo polinomial são preferíveis a algoritmos que são executados em tempo exponencial, ditos intratáveis. Convenciona-se que problemas que possam ser tratados por algoritmos de ordem polinomial são problemas *tratáveis* enquanto que, problemas que não podem ser solucionados por algoritmos dessa ordem são intratáveis (CORMEN, 2001) (VIANA, 1998). Problemas com complexidade polinomial pertencem à classe P, enquanto que os de complexidade exponencial pertencem à

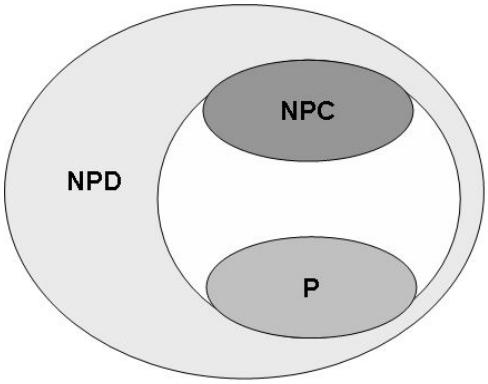


Figura 1: Relação das classes de problemas *P*, *NP*-Completo (NPC) e *NP*-Difícil (NPD). Fonte: (VIANA, 1998)

classe *NP*. Atualmente considera-se que $P \neq NP$, dado que não existe um algoritmo que consiga resolver um problema classificado como *NP* em tempo polinomial. Se tal algoritmo for descoberto, então todos os problemas *NP* poderiam ser reduzidos (transformados) ao problema resolvido, ou seja, $P = NP$. Se existe uma transformação em tempo polinomial de um problema de decisão π_1 para outro π_2 ($\pi_1 \propto \pi_2$), então se $\pi_1 \in P \Rightarrow \pi_2 \in P$. Considera-se π_1 um caso particular de π_2, ou seja, π_2 é pelo menos tão difícil quanto π_1.

A denominação *NP*-Completo foi apresentada por Cook (VIANA, 1998) que estabeleceu o problema de Satisfabilidade (*SAT*), na forma normal conjuntiva, como o problema mais difícil da classe *NP*, sendo que todos os problemas dessa classe podem ser reduzidos a ele. Para verificar se um problema π é *NP*-Completo, utiliza-se a redução mestre, onde tenta-se fazer $SAT \propto \pi$. Se um problema $\pi_0 \propto \pi$ para todo $\pi_0 \in NP - \text{Completo}$, então o problema π é considerado *NP*-Difícil (*NP-Hard*), ou seja, um problema desse tipo é pelo menos tão difícil quanto qualquer problema da classe *NP* (VIANA, 1998). A figura 1 mostra a relação entre as classes *P*, *NP*-Completo e *NP*-Difícil.

2.3 Grafos

Um grafo $G = (V, E)$ é uma estrutura formada por um conjunto V não-vazio de nós ou vértices (*vertex*) e de um conjunto E de pares de vértices (u, v), com $u, v \in V$ denominados arcos ou arestas (*edges*). A representação pictórica mais utilizada para os vértices é a figura de um círculo com um identificador (geralmente um número), e para os arcos é a figura de uma seta ou um traço que une dois círculos (VIANA, 1998) (CORMEN, 2001).

Um grafo é dito *não-dirijido* ou não-orientado quando o conjunto E de arcos for composto de pares não-ordenados, ou seja, os conjuntos de vértices (u, v) e (v, u) representam o

mesmo arco, que é chamado de arco incidente em u e v. Um grafo *dirigido* ou orientado é aquele cujo conjunto E é formado por pares ordenados de vértices (u,v), onde o arco deixa o vértice u e chega no vértice v. O arco é dito incidente de quando deixa o vértice u e incidente a quando chega no vértice v. Um grafo é dito valorado quando possui valores associados a vértices ou arestas.

Sendo o arco $(u,v) \in G = (V,E)$ diz-se que o vértice v é adjacente ao vértice u. Em um grafo não-dirigido a adjacência é simétrica. Já em um grafo dirigido, a adjacência respeita a ordem do par, podendo ser representada por $u \rightarrow v$.

O grau de um vértice v de um grafo não-dirigido é o número de arcos incidentes em v e o grau máximo entre todos os vértices será o grau do grafo. Em grafos dirigidos, o grau-de-saída de um vértice v é o número de arcos que deixam v e o grau-de-entrada é o número de arcos que chegam a v. O grau de um grafo dirigido é a máxima soma dos graus de saída e de entrada dentre todos os vértices do grafo.

Um caminho de tamanho k do vértice u até o vértice u' em um grafo $G = (V,E)$ é uma seqüência $\{v_0,\ldots,v_k\}$ de vértices tal que $u = v_0$, $u' = v_k$ e $(v_{i-1},v_i) \in E$ para $i = 1,\ldots,k$. O tamanho do caminho é a quantidade de arcos que este possui. Se o grafo for valorado o tamanho (ou custo) de um caminho é a soma dos valores das arestas percorridas. Um caminho pode ser considerado simples se todos os vértices do caminho são distintos.

Em um grafo dirigido, um ciclo ou circuito é um caminho $\{v_0,\ldots,v_k\}$ em que $v_0 = v_k$ e o caminho contém pelo menos 1 vértice. Um ciclo é dito simples se todos os vértices que o compõem forem distintos. Uma aresta que liga um vértice a ele mesmo é um ciclo de tamanho 1. Em um grafo não-dirigido, um ciclo ou cadeia é um caminho $\{v_0,\ldots,v_k\}$ em que $v_0 = v_k$ e $v_1,\ldots,v_k$ são distintos. Um ciclo é dito Euleriano se passa somente uma vez por todas as arestas de um grafo. O ciclo é dito Hamiltoniano se passa por todos os vértices somente uma vez. Um grafo que não contém ciclos é chamado de acíclico.

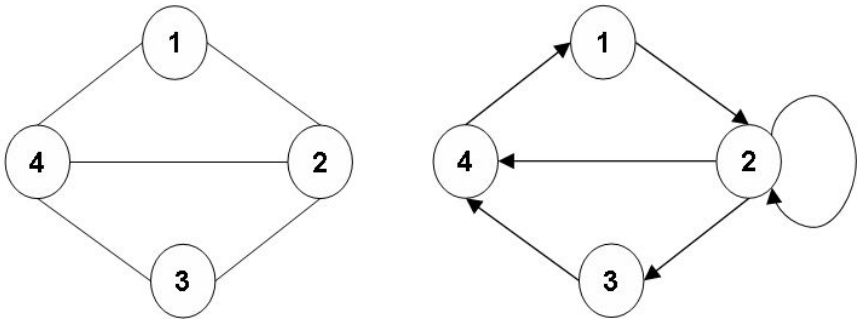
Um grafo é dito conexo se cada par de vértices é conectado por um caminho, sendo que se existir pelo menos um par de vértices conectados por uma aresta que não pertença a um caminho, o grafo é dito não-conexo. Os componentes conexos de um grafo são equivalentes a uma classe de vértices na qual, a partir de um vértice, pode-se alcançar qualquer outro dessa classe. Um grafo dirigido é fortemente conexo se cada dois vértices puderem ser alcançados mutuamente.

Dois grafos $G = (V,E)$ e $G' = (V',E')$ são ditos isomórficos se existe uma bijeção $f : V \rightarrow V'$ tal que $(u,v) \in E$ se e somente se $(f(u),f(v)) \in E'$. Ou seja, os vértices de V são

renomeados para serem vértices em V', mantendo os mesmos arcos.

O grafo $G' = (V', E')$ é considerado subgrafo de $G = (V, E)$ se $V' \subseteq V$ e $E' \subseteq E$.

Um grafo completo é um grafo não-dirigido no qual cada par de vértices é adjacente. Um grafo bipartido é um grafo não-dirigido $G = (V, E)$ que pode ser dividido em dois conjuntos V_1 e V_2 tal que $(u, v) \in E$ implica em $u \in V_1$ e $v \in V_2$ ou $u \in V_2$ e $v \in V_1$ (VIANA, 1998) (CORMEN, 2001).



(a) (b)

Figura 2: Exemplos de grafos (a) não dirigido, com $V = \{1,2,3,4\}$ e $E = \{(1,2),(2,3),(2,4),(3,4),(4,1)\}$ e (b) dirigido, com $V = \{1,2,3,4\}$ e $E = \{(1,2),(2,3),(2,4),(2,2),(3,4),(4,1)\}$

2.4 Heurísticas e Meta-heurísticas

Os problemas de otimização podem ser estruturados sob a forma de uma Função Objetivo, composta por variáveis de decisão, sujeitas ou não a restrições. Uma forma muito utilizada para representar essa estruturação é:

Otimize $f(x)$

Sujeito a $g_i(x) \geq b_i,\ i = 1, \ldots, m,$

onde x é um vetor de variáveis de decisão e $f(\cdot)$ e $g_i(\cdot)$ são funções. Problemas de Otimização Combinatorial são problemas que envolvem a busca da solução ótima entre um número significativo de possíveis soluções (RAYWARD-SMITH et al., 1996).

Um dos primeiros métodos estudados para resolução desse tipo de problema foi a enumeração completa, que consiste em verificar todas as soluções viáveis para uma determinada instância de um problema. Esse método, apesar de encontrar a solução ótima (ou ótimo global) com exatidão, é ineficiente dado o alto custo computacional. Mesmo métodos exatos que derivam dessa idéia, como o Método Simplex, ainda demonstram ter alto custo computacio-

Escolher $x \in X$ para iniciar o processo;
repetir
  Encontrar $x' \in N(x)$ tal que $f(x') < f(x)$;
  se $x' \neq \varnothing$ **então**
    $x \leftarrow x'$
  fim se
até x' não poder ser encontrado

Figura 3: Método da Descida

nal. A alta complexidade inerente aos problemas de otimização incentivou a busca por métodos computacionalmente mais eficientes, porém não exatos, chamados de heurísticas.

O termo heurística vem da palavra grega *heuriskein*, que significa "buscar" ou "descobrir". Heurística é um método desenvolvido para resolver um determinado problema a um custo computacional razoável, não sendo capaz de garantir a otimalidade do resultado (VIANA, 1998) (RAYWARD-SMITH et al., 1996).

O método heurístico de busca pode basear-se na busca em vizinhança. Assume-se que uma solução para um determinado problema de otimização é representada pelo vetor x, sendo X o conjunto de todas as soluções viáveis para o problema e o custo de cada x sendo $c(x)$. Cada solução $x \in X$ tem associado um conjunto de vizinhos $N(x) \subset X$, chamada de vizinhança de x. Cada $x' \in N(x)$ pode ser alcançado diretamente de x a partir de uma operação chamada movimento (RAYWARD-SMITH et al., 1996). Um exemplo de heurística com exploração de vizinhança é o Método da Descida (*Descent Method*) (GLOVER; LAGUNA, 1997), apresentado na figura 3. Neste método o objetivo é achar uma solução que minimize o valor da função objetivo de um determinado problema.

Uma falha do método heurístico é a propensão a encontrar ótimos locais. A fim de superar essa limitação, surgiu o conceito de meta-heurística (GLOVER; LAGUNA, 1997), que pode ser definida como uma estratégia mestre que guia e modifica um outro procedimento heurístico para buscar resultados além de um ótimo local. Uma meta-heurística possui estruturas que permitam guiar de maneira eficiente a heurística subordinada no espaço de soluções viáveis, como: memória adaptativa que impede que soluções sejam visitadas com freqüência (a exemplo, Busca Tabu), manipulação de conjuntos de soluções a cada iteração (a exemplo, Algoritmos Genéticos), aceitação de soluções não-atrativas a uma dada probabilidade (a exemplo, Têmpera Simulada).

A figura 4 ilustra o comportamento de uma busca realizada por uma meta-heurística.

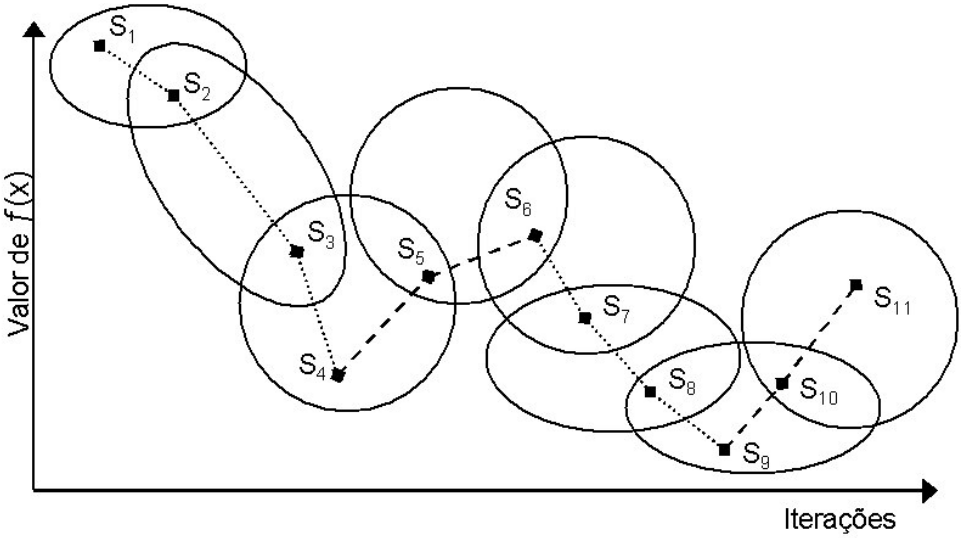


Figura 4: Comportamento de uma busca no espaço de soluções realizada por uma meta-heurística. Fonte: (OSMAN; LAPORTE, 1996)

Assume-se uma função $f(x)$ a ser minimizada. A busca encontra resultados mais promissores dentro das vizinhaças de soluções geradas a cada iteração. A admissão de resultados piores que os anteriormente encontrados ajuda na fuga de ótimos locais.

As meta-heurísticas são procedimentos de nível de abstração mais alto que as heurísticas, característica que deixa esses métodos mais propensos a adaptações e uniões com diversos tipos de técnicas de exploração do espaço de busca. Duas técnicas de exploração do espaço são muito utilizadas: intensificação e diversificação. Segundo Glover (GLOVER; LAGUNA, 1997) a técnica de intensificação refere-se a seguir uma linha de busca em particular, até que esta não seja mais viável. Diversificação refere-se a definição de séries de mudanças aleatórias antes de seguir por uma linha de pesquisa em particular.

Path Relinking é uma técnica utilizada na aplicação de estratégias de intensificação e diversificação. Basicamente essa técnica gera novas soluções pela exploração de trajetórias que conectem entre si soluções de elite, as quais apresentam bons atributos para a solução do problema. A partir de uma solução inicial é gerado um caminho, no espaço de busca gerado pela vizinhança, que leve a uma determinada solução destino. Isto é feito iterativamente, pela seleção de movimentos que contenham atributos presentes na solução destino.

Grupos de soluções podem ser considerados nesta abordagem, onde são definidos conjuntos de atributos de interesse que guiam a busca a partir do conjunto de soluções iniciais, para o conjunto de soluções finais. Estratégias de intensificação podem ser implementadas pela geração de soluções intermediárias obtidas na geração de caminhos entre soluções iniciais e finais semelhantes. Diversificação pode ser obtida gerando caminhos entre soluções dissimilares.

3 TECNOLOGIA DE GRUPO

Tecnologia de Grupo (TG) é uma filosofia em que problemas semelhantes são agrupados segundo alguma similaridade, de forma que se possa obter uma única solução para todos os elementos agrupados (LORINI, 1993). Aplicada ao ambiente de manufatura, a TG é utilizada principalmente na tarefa de reunir partes em grupos, segundo similaridades de projeto e/ou produção. Uma vez agrupadas em Famílias de Partes (FPs), estas partes passam a ser consideradas não mais individualmente, mas sim como um único lote produtivo, composto pela combinação de atributos delas. A produção de um lote, ao invés de vários, incorre em redução de custos de manufatura, ou seja, com a aplicação da TG, pode-se obter vantagens econômicas da produção em massa em um ambiente de pequenos lotes (LORINI, 1993) (GROOVER, 2001).

Para a implantação da TG na organização, o método de produção deve ser o de produção em lotes e deve existir a possibilidade de agrupar os itens a serem produzidos em FPs (GROOVER, 2001). Para realizar a implantação, algumas etapas devem ser consideradas (LORINI, 1993):

- adequação do *layout* físico: as máquinas necessárias para o processamento de uma FP devem ser agrupadas em Células de Manufatura (CMs), reduzindo assim os custos com manipulação de material e o tempo de deslocamento das partes.
- estruturação dos dados: as informações sobre as partes a serem processadas e os recursos necessários devem estar disponíveis para a formação das FPs e CMs. Esta estruturação pode ser feita através de implantação de um Sistema de Classificação e Codificação, que auxilia na integração dos dados para *design* e manufatura.
- formação das FPs: nesta etapa trata-se o principal problema do processo de implantação. Para grandes quantidades de partes, a classificação das mesmas pode ser uma tarefa complexa. É a partir da definição das FPs que as CMs devem ser arranjadas.

Das vantagens a serem obtidas com a implantação da TG, podem ser citadas (GROOVER, 2001) (LORINI, 1993):

- controle mais eficiente do processo produtivo;
- racionalização da produção e simplificação do escalonamento;
- padronização de partes, ferramentas e outros recursos produtivos;
- melhor acesso e precisão às informações;
- tempos de *setup* reduzindos;
- *work-in-process* reduzido;
- significativa economia nos custos de produção.

3.1 Histórico

O conceito de Tecnologia de Grupo vem sendo usado informalmente desde a década de 20. Nos Estados Unidos, em 1925, Flanders apresentou um trabalho à Sociedade Americana de Engenharia Mecânica, onde mostrou um método de organização de manufatura que aplicava a filosofia da hoje conhecida TG. Em 1930, Taylor notou em seus estudos que existiam similaridades entre as tarefas realizadas em uma linha de montagem que poderiam ser categorizadas. Em 1937 na União Soviética, A. Sokolovskiy descreveu as principais características da TG, propondo que partes similares poderiam ser produzidas por uma seqüência padronizada de operações, permitindo que técnicas de *flow shop* pudessem ser aplicadas em indústrias que trabalhavam com pequenos lotes. Em Paris, em 1949, o sueco A. Korling apresentou um trabalho de "grupos de produção" (*group production*), o qual era uma adaptação das técnicas de linha de produção para manufatura em lotes (GROOVER, 2001) (BEDWORTH; HENDERSON; WOLFE, 1991). Em 1959 o pesquisador russo S. Mitrofanov, no livro intitulado *Scientific Principles of Group Technology*, formalizou o conceito de TG:

> ... um método de manufaturar partes pela classificação dessas em grupos e, subseqüentemente, aplicando a cada grupo operações tecnológicas similares. O maior benefício deste método é referente à redução de custos associados à produção em larga escala em uma situação de produção em pequena escala, o que é de fundamental importância na produção em lotes e nas seções de "jobbing" da indústria (Mitrofanov *apud* (BEDWORTH; HENDERSON; WOLFE, 1991)).

Esse livro foi amplamente lido e contribuiu para a implantação dos conceitos de TG em cerca de 800 fábricas na União Soviética, até 1965. Durante essa década o conceito de TG evoluiu para agrupamentos de partes e de máquinas, formando as FPs e as CMs. O livro foi

difundido na Europa inteira, dando origem a vários estudos na Holanda, Noruega e Suíça. Na Alemanha, Opitz realizou um estudo das partes manufaturadas nas indústrias alemãs, desenvolvendo o seu conhecido sistema de classificação e codificação. Na Inglaterra, destacaram-se trabalhos como o sistema de classificação Brisch. Na Itália, Burbidge apresentou o método de análise de fluxo de produção, considerando as rotas das partes durante o processo produtivo (GROOVER, 2001) (LORINI, 1993).

Nos Estados Unidos a primeira aplicação de TG foi feita em 1969 na divisão de Langston da Harris-Intertype, em New Jerse. O tradicional parque de máquinas no *layout* funcional foi reorganizado em um *layout* de "linhas de FPs", sendo cada linha de máquinas responsáveis pela produção de um tipo de parte. As partes eram reunidas por inspeção visual e a aplicação da TG melhorou a produtividade em 50%.

Desde a década de 60 os japoneses vem adotando a filosofia de TG na indústria. A Sociedade Japonesa para a Promoção da Mecânica incentivou, entre 1969 e 1973 várias indústrias metalúrgicas a utilizarem TG para sua organização. A Sociedade Japonesa para a Promoção de Máquinas realizou importantes estudos na área de TG, desde 1967. Como resultado desses estudos foram desenvolvidos os sistemas de classificação KK-1 (1970), KK-2 (1973) e KK-3 (1976) (LORINI, 1993).

Atualmente os estudos na área de TG combinam conceitos de simplificação e integração. Novas técnicas, como heurísticas e inteligência artificial estão sendo usadas para a organização do ambiente produtivo, na integração das informações na indústria e na formação das FPs e CMs.

3.2 Formação de Famílias de Partes

Uma Família de Partes é um conjunto de partes agrupados segundo alguma similaridade. As partes dentro de uma FP não são necessariamente iguais, mas têm similaridade suficiente para que possam ser reunidas, respeitando limitações impostas, como restrições físicas de maquinário. Geralmente o critério de similaridade pelo qual as partes são agrupadas é o de forma geométrica ou o de processo produtivo. As figuras 5 e 6 ilustram o conceito de FP com esses dois critérios de agrupamentos.

As partes agrupadas segundo o critério de processo produtivo são aquelas que podem ser processadas nas mesmas máquinas. Isso permite que seja criado um processo padrão para a manufatura da FP gerada. Com base na formação das FPs, pode-se organizar o parque produtivo em CMs, o que minimiza os custos e o tempo de manipulação e carregamento das partes nas

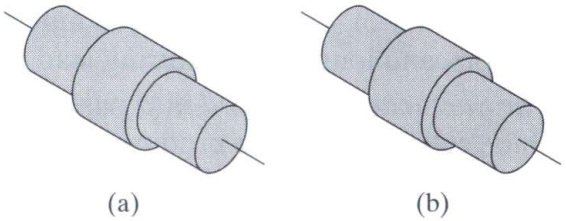


Figura 5: Partes agrupadas segundo critério geométrico: as partes (a) e (b) possuem forma semelhante, mas diferentes requerimentos de manufatura. Fonte: (GROOVER, 2001)

máquinas.

As partes agrupadas segundo similaridades geométricas podem vir a precisar de máquinas diferentes para o seu processamento, em função de parâmetros como tipo de material, tamanho e processos específicos. Em uma planta com *layout* de manufatura por função, isso acarretaria em maiores custos de manipulação de materiais, tempo de carregamento, tempo de *setups* e filas de espera de partes, se comparados com plantas organizadas em CMs. Nas figuras 7 e 8 podem ser vistas as diferenças de quantidade de manipulação de material nesses dois tipos de plantas (GROOVER, 2001) (LORINI, 1993).

A formação das FPs é o ponto crítico da implantação dos princípios de TG na organização. Para a formação das FPs, existem três métodos amplamente conhecidos, apresentados nos itens a seguir: inspeção visual, classificação e codificação e análise de fluxo de produção.

3.2.1 Inspeção Visual

A Inspeção Visual consiste em analisar visualmente as partes ou fotografias destas, reunindo-as em FPs segundo alguma similaridade. Este método depende basicamente da ex-

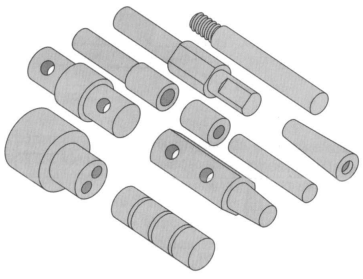


Figura 6: Famílias de partes reunidas segundo processos de manufatura similares, possuindo atributos geométricos diferentes. Todas as partes são manufaturadas em blocos cilíndricos. Algumas delas requerem furação ou fresa. Fonte: (GROOVER, 2001)

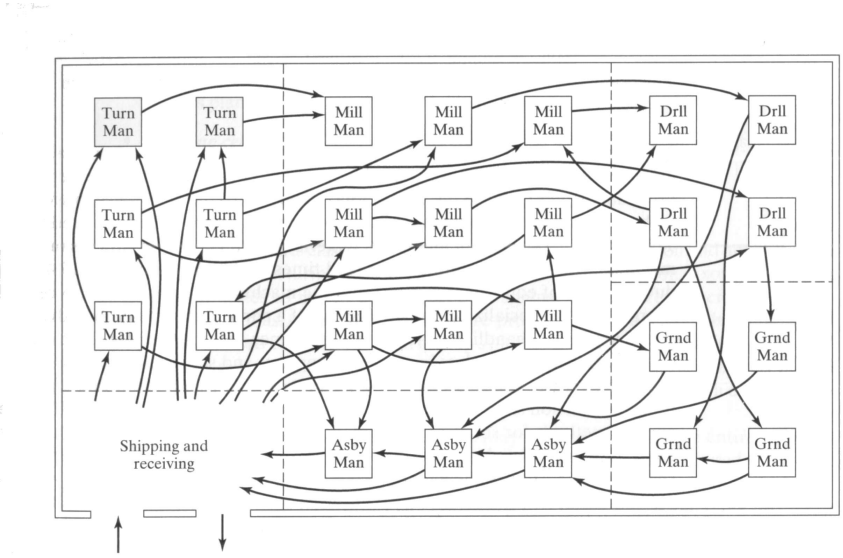


Figura 7: Planta com *layout* funcional. (Símbolos: "Turn" = tôrno, "Mill" = fresa, "Drll" = furação, "Grnd" = afiação, "Asby" = montagem, "Man" = operação Manual. Flexas indicam o fluxo de material através da planta. Linhas pontilhadas indicam a separação das máquinas em departamentos). Fonte: (GROOVER, 2001)

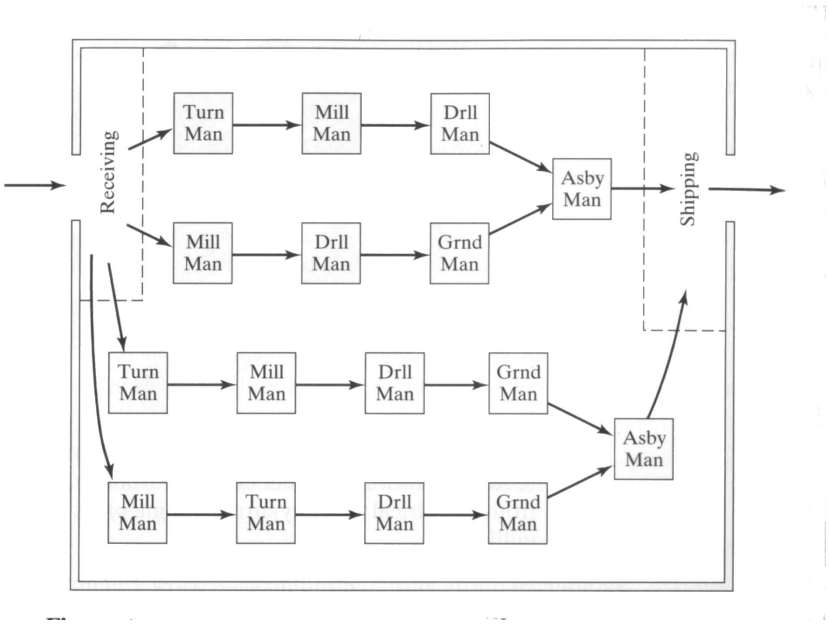


Figura 8: Planta com *layout* de Célula de Manufatura. (Símbolos: "Turn" = tôrno, "Mill" = fresa, "Drll" = furação, "Grnd" = afiação, "Asby" = montagem, "Man" = operação Manual. Flexas indicam o fluxo de material entre as CMs.) Fonte: (GROOVER, 2001)

periência do usuário, sujeito a baixa precisão e limitado pelo número de partes que podem ser analisadas no tempo disponível. Historicamente foi um dos primeiros métodos de formação de FPs utilizados (GROOVER, 2001) (LORINI, 1993) (BEDWORTH; HENDERSON; WOLFE, 1991).

3.2.2 Classificação e Codificação

Na técnica de Classificação e Codificação (CC) os atributos geométricos e de processo das partes são descritos segundo um sistema de codificação, que é um código numérico ou alfanumérico. Esses atributos podem ser divididos em dois grupos: (*i*) atributos de projeto, referentes aos atributos geométricos, ao tamanho e ao material e (*ii*) atributos de manufatura, que consideram a seqüência de passos necessários para fazer a parte. Uma vez que as partes estão codificadas, pode-ser formar códigos que representem agrupamentos de partes similares. Algumas vantagens advindas do uso de um sistema de CC, de acordo com Groover (GROOVER, 2001) e Lorini (LORINI, 1993) são:

- agrupamentos de partes segundo características de projeto permitem uma rápida recuperação de desenhos de partes, o que auxilia o projetista no desenvolvimento.
- códigos das partes podem ser usados para a formação das FPs, permitindo assim que as CMs sejam formadas.
- o uso do sistema de CC contribui para uma padronização dos projetos, dos produtos, do ferramental e dos processos da organização, permitindo a construção de uma base de dados que organize as informações segundo o tipo de atributo. Esse fator permite que sejam feitas consultas mais rápidas e planejamentos mais precisos.

Para realizar a classificação das partes deve ser feita uma análise de atributos geométricos e de manufatura de cada parte, e de acordo com eles, os campos referentes a esses atributos são preenchidos com um determinado código. Esse processo pode ser feito manualmente ou auxiliado por computador, o que é recomendável.

Os sistemas de CC podem ser classificados segundo o tipo de atributo, em três categorias (GROOVER, 2001) (LORINI, 1993):

1. sistemas baseados em atributos de projeto;
2. sistemas baseados em atributos de manufatura;
3. sistemas baseados em atributos de projeto e manufatura.

Quanto ao tipo de estrutura do código, os sistemas de CC podem ser divididos nas seguintes categorias (GROOVER, 2001) (LORINI, 1993) (JHA, 1991):

1. Estrutura Hierárquica ou Monocódigo: neste tipo de estrutura, a interpretação de cada dígito depende do valor dos dígitos anteriores. A estrutura hierárquica geralmente é representada por uma estrutura do tipo árvore, como apresentado na figura 9.
2. Estrutura tipo Cadeia ou Policódigo: nesta estrutura, cada símbolo pode ser interpretado independentemente dos anteiores, ou seja, um símbolo não depende do valor dos símbolos anteriores. Essa independência permite que a interpretação dos símbolos possa ser feita sempre na mesma seqüência.
3. Estrutura Combinada ou Híbrida: esta estrutrura constitui-se da associação de monocódigos interligados a policódigos.

3.2.2.1 Exemplos de Sistemas de Classificação e Codificação

Alguns dos sistemas de CC que podem ser citados são (GROOVER, 2001) (LORINI, 1993): o Sistema Opitz, o Sistema Brish (*Brisch-Birn Inc*), CODE (*Manufacturing Data Systems Inc*), CUTPLAN (*Metcut Associates*), DCLASS (Universidade de Brighman Young), MultiClass (OIR - Organização para Pesquisa Industrial), MICLASS (*Metal Institute Classification System*) e o KK3 (Sociedade Japonesa para a Promoção de Máquinas Industriais). Nos itens a seguir serão apresentados a estrutura do Sistema Opitz e do Sistema MultiClass.

3.2.2.1.a Sistema Opitz Esse sistema foi desenvolvido por H. Opitz da Universidade de Aachen, na Alemanha, sendo um dos pioneiros na área de TG. O Sistema Opitz usa o seguinte esquema de codificação:

12345 6789 ABCD

O código básico consiste de nove dígitos, os quais podem ser extendidos pela adição de mais quatro dígitos. Os primeiros nove dígitos representam as características de *design* e manufatura, e sua interpretação pode ser vista na figura 10. Os dígitos de 1 a 5 são chamados de código de forma, referindo-se às características primárias de projeto (como formato externo, presença de furos, dentes etc). Os dígitos de 6 a 9 constituem a parte suplementar do código, que indica os atributos usados para a manufatura da parte (como material, dimensões, forma e precisão do processamento). Os quatro últimos dígitos (ABCD) são chamados de código secundário e podem ser usados de acordo com a aplicação.

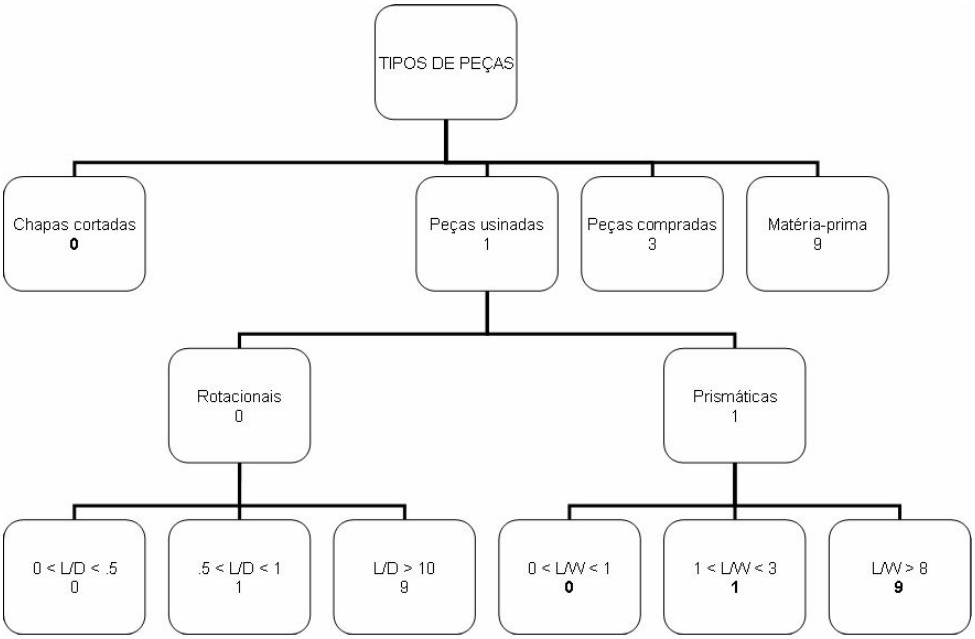


Figura 9: Estrutura Monocódigo Fonte (LORINI, 1993)

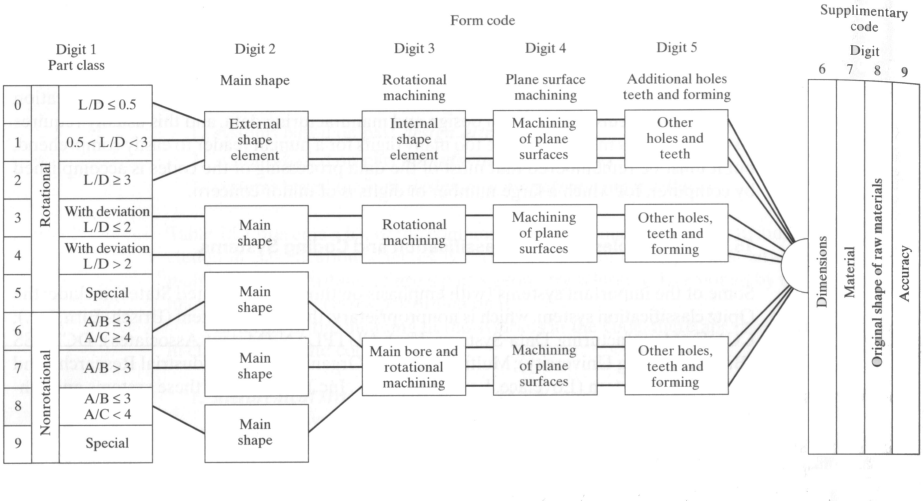


Figura 10: Estrutura básica do Sistema Opitz Fonte: (GROOVER, 2001)

3.2.2.1.b Sistema MultiClass Esse sistema foi desenvolvido pela Organização para Pesquisa Industrial (OIR), sendo usado para codificar uma variedade de tipos de itens de manufatura, incluindo partes de metal usinadas, ferramental, dispositivos eletrônicos, máquinas de montagens e outros elementos (GROOVER, 2001). Esse sistema utiliza uma estrutura hierárquica, baseada em codificação por árvore de decisão. A aplicação consiste de uma série de menus iterativos que auxiliam na codificação da parte. A estrutura de codificação consiste em 30 dígitos divididos em duas regiões: uma pré-definida pela OIR e outra atribuída pelo usuário para fins específicos. A figura 11 mostra a significado dos primeiros 21 dígitos.

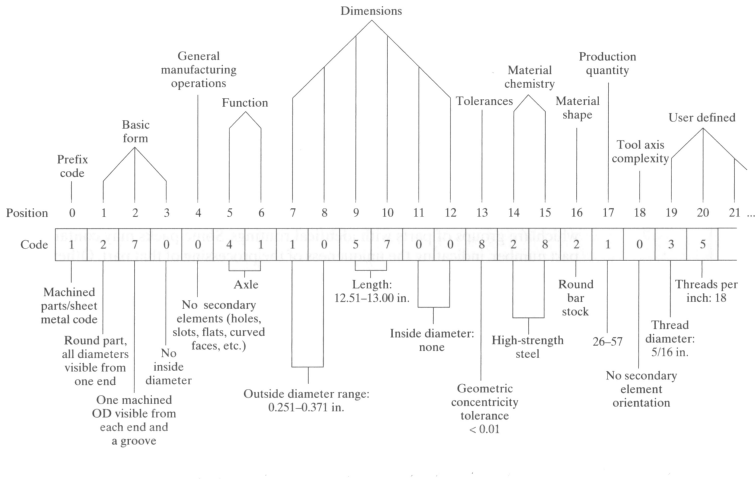


Figura 11: Codificação do Sistema MultiClass para os primeiros 20 dígitos Fonte: (GROOVER, 2001)

3.2.3 Análise de Fluxo de Produção

Para a formação de FPs em Análise do Fluxo de Produção (AFP), são utilizadas informações contidas em planilhas de rotas das partes ao invés das informações dos desenhos das partes. Partes com rotas idênticas ou similares são agrupadas em FPs, independente do tamanho ou atributos geométricos, permitindo a formação de CMs lógicas em uma planta com *layout* baseado em TG. Por se basear nas planilhas de rotas das partes, a precisão desse método depende da confiabilidade dos dados registrados nessas planilhas. Convém ressaltar que essas planilhas podem ser preparadas por diferentes projetistas, podendo conter operações que não sejam ótimas ou que tenham operações desnecessárias ou ilógicas (GROOVER, 2001) (LORINI, 1993).

Com base nas informações de planilhas de rotas é feita a formação das FPs, segundo três tipos de formulação: matricial, programação matemática e grafos.

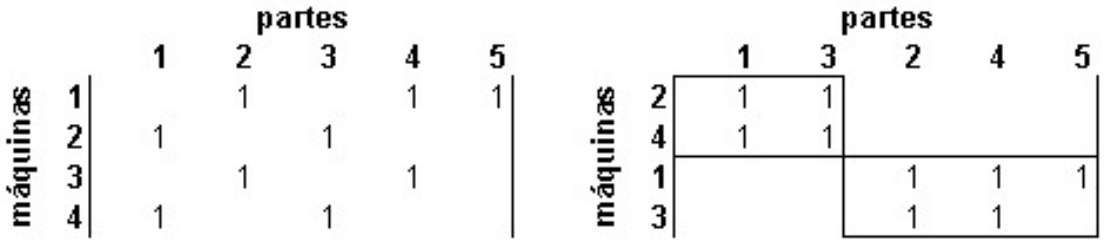
3.2.3.1 Formulação Matricial

A formulação matricial representa os dados colhidos na AFP sob a forma de uma matriz incidente $[a_{ij}]$, geralmente do tipo parte *versus* máquinas. Esta matriz consiste de entradas 0 e 1, onde o elemento $a_{ij} = 1$ indica que a máquina i processa a parte j, e $a_{ij} = 0$ caso contrário.

Algoritmos de agrupamento (*cluster algorithms*) são utilizados para transformar a matriz no estado inicial, ou seja, sem que as máquinas e as partes estejam agrupadas em FPs, em uma matriz do tipo bloco-diagonal, onde os agrupamentos de máquinas-partes são facilmente identificados.

Por exemplo, considerando a matriz inicial apresentada na figura 12 (a), aplicando um método de rearranjar as linhas e colunas dessa matriz, pode-se obter a matriz bloco-diagonal (b). Nesta matriz, pode-se identificar as famílias $FP(1) = \{1,3\}$ e a $FP(2) = \{2,4,5\}$ com as células de máquinas $CM(1) = \{2,4\}$ e $CM(2) = \{1,3\}$. A matriz resultante possui agrupamentos ditos mutuamente separados, pois nenhuma parte da $FP(1)$ precisa de uma máquina da $FP(2)$ e vice-versa.

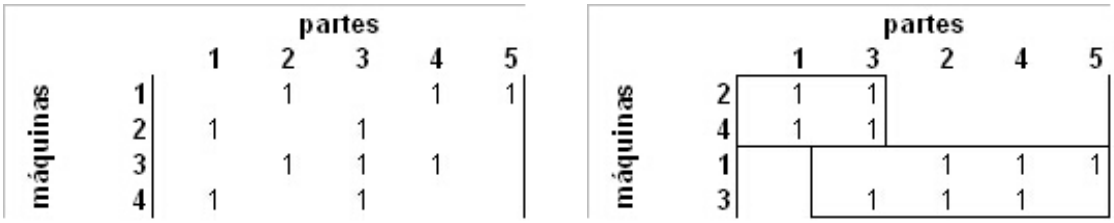
Mas nem sempre é possível obter agrupamentos mutuamente separáveis; é o caso das matrizes apresentadas na figura 13. Neste caso, os agrupamentos obtidos possuem elementos excepcionais (entradas "1" pertencentes a duas FPs ao mesmo tempo). Esses agrupamentos são ditos parcialmente separáveis. A ocorrência de agrupamentos parcialmente separáveis pode ser interpretada como gargalos de produção (*bottleneck*), onde os elementos excepcionais representam máquinas que recebem fluxo de partes de duas ou mais CMs (GROOVER, 2001) (KUSIAK, 1992) (LORINI, 1993) (JHA, 1991).



(a) Matriz inicial (b) Matriz agrupada

Figura 12: Matrizes Inicial e Agrupada. A matriz (a) informa a rota das partes e a matriz (b), com as linhas e colunas rearranjadas, mostra as FPs formadas. Nota-se que as FPs são mutuamente separáveis.

Para realizar o agrupamento das partes em FPs segundo a formulação matricial, foram desenvolvidas técnicas como (JHA, 1991):



(a) Matriz inicial (b) Matriz agrupada

Figura 13: Matrizes Inicial e Agrupada. A matriz (a) informa a rota das partes e a matriz (b), com as linhas e colunas rearranjadas, mostra as FPs formadas. Nota-se que a $FP(1) = \{1,3\}$ e a $FP(2) = \{2,4,5\}$ são parcialmente separáveis, dado que compartilham a máquina 1.

- métodos baseados em coeficientes de similaridade;
- algoritmos baseados em ordenação;
- algoritmo de *Bond-Energy*;
- algoritmos baseados em custos;
- algoritmos de identificação de agrupamentos;

Nos itens a seguir serão descritos alguns desses métodos de formação de FPs.

3.2.3.1.a Coeficientes de similaridade O método de geração de FPs e CMs por coeficientes de similaridade utiliza um algoritmo de agrupamento baseado em uma métrica que mensura a similaridade entre as máquinas analisadas Dependendo da aplicação, os coeficientes de similaridade são usados para fazer a análise das partes, ou seja, as partes com o coeficiente de similaridade maior são agrupadas formando uma FP.

Para ilustrar o uso dessa técnica, Kusiak (JHA, 1991, p.153) apresenta o algoritmo *Single Linkage Cluster Algorithm* (SLCA). Esse algoritmo gera CMs aplicando um determinado coeficiente de similaridade entre todos os possíveis pares de máquinas computados. O exemplo a seguir utiliza o coeficiente de similaridade de McAuley, o qual mensura a similaridade s_{ij} entre as máquinas i e j, segundo a seguinte expressão:

$$s_{ij} = \frac{\sum_{k=1}^{n} \delta_1(a_{ik}, a_{jk})}{\sum_{k=1}^{n} \delta_2(a_{ik}, a_{jk})} \tag{3.1}$$

onde

$$\delta_1(a_{ik}, a_{jk}) = \begin{cases} 1 & \text{se } a_{ik} = a_{jk} = 1 \\ 0 & \text{caso contrário} \end{cases} \tag{3.2}$$

$$\delta_2(a_{ik}, a_{jk}) = \begin{cases} 0 & \text{se } a_{ik} = a_{jk} = 1 \\ 1 & \text{caso contrário} \end{cases} \tag{3.3}$$

No SLCA as FPs são geradas segundo um limiar de aceitação definido para o coeficiente de similaridade. Considera-se a matriz (b) da figura 13 (JHA, 1991) para o cálculo do coeficiente de similaridade, apresentado na figura 14.

$$s_{12} = s_{34} = \frac{2}{2+1} = 0,75$$

$$s_{13} = \frac{1}{4+1} = 0,25$$

$$s_{14} = s_{23} = s_{24} = \frac{0}{5} = 0$$

Figura 14: Cálculo do coeficiente de similaridade para a matriz 12.

Se for considerado um limiar de aceitação de 60% de similaridade, as CMs formadas são: $CM(1) = \{1,2\}$ e $CM(2) = \{3,4\}$. Kusiak cita algumas limitações do SLCA, como a de não considerar a duplicação de máquinas gargalo (JHA, 1991).

Existem outros algoritmos de formação de FPs baseados em coeficientes de similaridade. Um dos algoritmos que demostra ter bastante robustez é o *Average Linkage Cluster* (ALC), que mensura a similaridade entre agrupamentos de máquinas, supondo que a similaridade entre os agrupamentos é a mesma para qualquer um dos elementos de cada agrupamento analisado (JHA, 1991) (YIN; YASUDA, 2005). O algoritmo ALC é composto dos seguintes passos:

Passo 1. computar o coeficiente de similaridade para todos os pares de máquinas e armazená-los em uma matriz de similaridade ;

Passo 2. unir os dois objetos mais similares (duas máquinas, uma máquina e um grupo ou dois grupos de máquinas), formando um novo grupo ;

Passo 3. avaliar o coeficiente de similaridade entre o novo grupo de máquinas e as máquinas restantes da seguinte forma:

$$s_{tv} = \frac{\sum_{i \in t} \sum_{j \in v} s_{ij}}{N_t N_v}, \tag{3.4}$$

onde i é uma máquina do grupo t; j é uma máquina do grupo v; N_t é o número de máquinas do grupo t e N_v é o número de máquinas do grupo v .

Passo 4. quando todas as máquinas estiverem agrupadas em um grupo ou tenha sido atingido um número pré-definido de grupos, ir para o passo 5; caso contrário, voltar ao passo 2.

Passo 5. designar cada parte à uma CM, onde o número de gargalos seja mínimo .

As métricas usadas pelos algoritmos de agrupamento geralmente são razões onde, o numerador mensura quão similares dois recursos são um do outro e o denominador mensura o quão similares espera-se que sejam (MOSIER; YELLE; WALKER, 1997).

Mosier, Yelle e Walker (MOSIER; YELLE; WALKER, 1997) apresentam um levantamento de métricas usadas para resolver o problema de geração de FPs, classificando-as segundo requerimentos de informação e estrutura funcional. Yin e Yasuda (YIN; YASUDA, 2005) realizam um estudo comparativo entre 20 coeficientes de similaridade, usando o algoritmo de agrupamento ALC. Para realizar a comparação, os autores utilizam medidas de performance para agrupamentos gerados pelo ALC. Duas dessas medidas são bem conhecidas:

- Eficiência de Agrupamento (*Group Efficiency*): desenvolvida por Chandresekharan e Rajagopalan (JHA, 1991), é uma razão ponderada dada pela expressão:

$$\eta_1 = w\eta_1 + (1-w)\eta_2 \tag{3.5}$$

onde

$$\eta_1 = \frac{o-e}{o-e+v} \tag{3.6}$$

$$\eta_2 = \frac{MP-o-v}{MP-o-v+e} \tag{3.7}$$

sendo M o número de máquinas; P o número de partes; o é o número de operações "1" na matriz de incidência máquina-parte $[a_{ij}]$; e o número de elementos excepcionais na solução; v o número de "0" na solução; w é o peso atribuído ao movimento inter-celular. Esse índice tem escala de 0 a 1. Para grandes quantidades de máquinas e partes, esse índice apresenta baixa taxa de discriminância, o que se torna um problema (YIN; YASUDA, 2005).

- Eficácia de Agrupamento (*Group Efficacy*): esse índice foi criado por Kumar e Chandra-

sekharan:

$$\tau = \frac{(1-\varphi)}{1+\phi} \tag{3.8}$$

onde φ é a razão entre o número de elementos excepcionais e o número total de elementos; ϕ é a razão entre número de "0" dentro dos agrupamentos e o número total de elementos.

3.2.3.1.b Algoritmos baseados em ordenação Essa técnica consiste em formar FPs ordenando as linhas e as colunas da matriz de incidência. Um algoritmo que implementa essa técnica é o *Rank Order Cluster* (ROC) (JHA, 1991):

Passo 1. para cada linha da matriz de incidência, atribuir um peso binário e calcular o peso decimal equivalente ;

Passo 2. ordenar as linhas da matriz binária em ordem decrescente de pesos decimais ;

Passo 3. repetir os passos acima para cada coluna ;

Passo 4. repetir os passos acima até que a posição de cada elemento em cada linha e cada coluna não mude. O peso para cada linha i e cada coluna j é calculado da seguinte forma:

$$\text{linha}_i = \sum_{k=1}^{n} a_{ik} 2^{n-k} \tag{3.9}$$

$$\text{coluna}_j = \sum_{k=1}^{m} a_{kj} 2^{n-k} \tag{3.10}$$

Na matriz final gerada pelo algoritmo ROC, os agrupamentos são identificados visualmente.

3.2.3.1.c Algoritmo *Bond-Energy* Esse algoritmo foi desenvolvido por McCormick (JHA, 1991) e visa gerar blocos diagonais pela maximização da medida de efetividade ME dada pela seguinte expressão:

$$ME = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} [a_{i,j-1} + a_{i,j+1} + a_{i-1,j} + a_{i+1,j}] \tag{3.11}$$

Os passos do Algoritmo *Bond-Energy* (ABE) são descritos a seguir:

Passo 1. atribuir $j = 1$. Selecionar uma das colunas arbitrariamente.

Passo 2. colocar cada uma das $n - j$ colunas restantes, uma de cada vez, em cada uma das $j + 1$ posições, uma a uma, e computar a contribuição de cada coluna segundo a ME.

Colocar a coluna que tem a maior contribuição incremental na ME na sua melhor posição. Fazer $j = j + 1$ e repetir os passos acima até $j = n$.

Passo 3. quando todas as colunas estiverem movidas, repetir o processo para as linhas.

3.2.3.1.d Método Baseado em Custo O método baseado em custos leva em consideração determinados custos de produção na formação das FPs e CMs. Askin e Subramanian (Askin *apud* (JHA, 1991)) desenvolveram um método que considera os seguintes custos de manufatura:

1. custo fixo e variável do maquinário;
2. custo de *setup*;
3. custo do inventário no ciclo de produção;
4. custo do inventário em-processo (*work-in-process*);
5. custo de manipulação de material.

Esse método consiste de três etapas: (*i*) as partes são classificadas por meio de um sistema de codificação; (*ii*) é feita uma tentativa de gerar FPs viáveis baseado nos custo de manufatura; (*iii*) o *layout* das CMs geradas é analisado.

3.2.3.1.e Algoritmo de Identificação de Agrupamentos Esse algoritmo é um processo iterativo de seleção de linhas e colunas para a formação de FPs e CMs simultaneamente. Kusiak e Chow (KUSIAK; CHOW, 1987) desenvolveram um Algoritmo de Identificação de Grupos (AIG) que tenta formar agrupamentos mutuamente separáveis. Os passos desse algoritmo são os seguintes:

Passo 1. atribuir o número da iteração $k = 1$.

Passo 2. selecionar qualquer linha i da matriz de incidência A^k e desenhar uma linha horizontal h_i através dela (A^k é a matriz incidente A na iteração k).

Passo 3. para cada entrada 1 cruzada pela linha horizontal h_i, desenhar uma linha vertical v_j.

Passo 4. para cada entrada 1 cruzada pela linha vertical v_j, desenhar uma linha horizontal h_k.

Passo 5. repetir os passos 3 e 4 até que não restarem mais entradas 1 cruzadas. Todas as entradas 1 cruzadas por duas vezes formam as respectivas $FP-k$ e $CM-k$.

Passo 6. transformar a matriz incidente A^k em A^{k+1} removendo as linhas a as colunas correspondentes às linhas traçadas nos passos 3 e 4.

Passo 7. Se a matriz $A^{k+1} = \varnothing$, parar; caso contrário, atribuir $k = k+1$ e voltar ao passo 2.

Kusiak e Chow mostram que o AIG visita cada elemento da matriz duas vezes, ou seja, se a matriz possui *mn* elementos, o número de operações computacionais elementares será $2mn$, ou seja, a complexidade do AIG é $O(2mn)$.

Apesar de ser computacionalmente eficiente, o AIG não considera matrizes que resultem em agrupamentos parcialmente separáveis. Dado essa limitação, Kusiak e Chow apresentam no mesmo trabalho (KUSIAK; CHOW, 1987) o Algoritmo de Identificação de Grupos Extendido (AIGE).

O AIGE utiliza uma matriz incidente $[a_{ij}]$ do tipo máquina-parte com uma modificação: à cada coluna (parte) está associado um custo. Esse custo pode significar:

- custo de subcontratação (custo de processar a parte em outra planta).
- custo de produção.
- fluxo de partes (partes com maior fluxo entre as máquinas aumentam os custos de manipulação de material).

O AIGE utiliza-se do custo de cada parte para decidir a formação das FPs e CMs, como mostrado nos passos a seguir:

Passo 1. atribuir o número de iteração $k = 1$ e atribuir o limite superior N de máquinas em uma CM.

Passo 2. na matriz incidente A^k, selecionar a coluna j que possua no máximo N máquinas e tenha o maior custo possível. Desenhar uma linha vertical v_j na coluna selecionada.

Passo 3. para cada linha i correpondente às entradas 1 cruzadas pela linha vertical v_j, desenhar uma linha horizontal h_i. As máquinas correspondentes a essas linhas integrarão a $CM-k$.

Passo 4. Seja $V^{(k)}$ o conjunto de todas as colunas cruzadas exatamente uma vez por qualquer linha horizontal h_i. No conjunto $V^{(k)}$, selecionar a coluna com o custo máximo (uma de cada vez) e aplicar o AIG. Se essa coluna não aumenta o número de máquinas na $CM-k$ acima do limite N imposto, desenhar uma linha sobre ela; caso contrário,

adicionar essa parte ao conjunto de partes a serem deletados de A^k e selecionar a coluna com o próximo custo mais alto. Repetir esse processo até que todos os elementos de $V^{(k)}$ sejam visitados.

Passo 5. para todas as entradas 1 cruzadas duas vezes, formar a $CM-k$ e a $FP-k$.

Passo 6. transformar a matriz A^k em A^{k+1} pela remoção das linhas e das colunas cruzadas duas vezes pelas linhas horizontais e verticais desenhadas nos passos 2 a 4.

Passo 7. se a matriz $A^{k+1} = \varnothing$, parar; caso contrário, atribuir $k = k+1$ e ir para o passo 2.

3.2.3.2 Formulação por Programação Matemática

Geralmente os métodos baseados em programação matemática utilizam-se da distância d_{ij} entre as partes i e j para a formação das FPs. Essa distância é um valor real que obedece aos seguintes axiomas:

- Reflexividade: $d_{ij} = 0$;
- Simetria: $d_{ij} = d_{ji}$;
- Desigualdade triangular: $d_{iq} = d_{ip} + d_{pq}$.

Existem várias métricas usadas para medir a distância entre partes, sendo que podem ser utilizados alguns dos coeficientes de similaridade para esse fim. Como exemplo de medida de distância, pode-se citar (JHA, 1991):

- a medida de distância de Minkowski:

$$d_{ij} = \left[\sum_{k=1}^{n} \mid a_{ik} - a_{jk} \mid^r \right]^{\frac{1}{r}} \tag{3.12}$$

  onde r é um inteiro positivo e n é o número de partes.

- a medida de distância ponderada de Minkowski:

$$d_{ij} = \left[\sum_{k=1}^{n} w_k \mid a_{ik} - a_{jk} \mid^r \right]^{\frac{1}{r}} \tag{3.13}$$

  onde w_k é um peso inteiro para a distância.

- Distância de *Hamming* dada pela expressão:

$$d_{ij} = \sum_{k=1}^{n} \delta(a_{ik}, a_{jk}) \tag{3.14}$$

onde

$$\delta(a_{ik}, a_{jk}) = \begin{cases} 1, & \text{se } a_{ik} \neq a_{jk}; \\ 0, & \text{caso contrário.} \end{cases} \tag{3.15}$$

Os métodos de programação matemática que utilizam esse tipo de medida, apresentados nos itens a seguir, são: Modelo das p-Medianas, Modelo Generalizado das p-Medianas e Programação Quadrática.

3.2.3.2.a Modelo das p-Medianas Esse modelo é utilizado para reunir partes em FPs. Considerando os seguintes parâmetros

$m =$ número de máquinas

$n =$ número de partes

$p =$ número de FPs

$d_{ij} =$ medida de distância entre as partes i e j

$$x_{ij} = \begin{cases} 1, & \text{se a parte } i \text{ pertence à FP } j \\ 0, & \text{caso contrário} \end{cases}$$

Podemos definir esse modelo como a minimização do total das somas das distâncias entre quaisquer partes i e j.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{3.16}$$

Sujeito a

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ para todo } i = 1, \ldots, n \tag{3.17}$$

$$\sum_{j=1}^{n} x_{ij} = p \tag{3.18}$$

$$x_{ij} \leq x_{ji} \text{ para todo } i = 1, \ldots, n \text{ e } j = 1, \ldots, n \tag{3.19}$$

$$x_{ij} = 0, 1 \text{ para todo } i = 1, \ldots, n \text{ e } j = 1, \ldots, n \tag{3.20}$$

A restrição 3.17 assegura que cada parte pertença apenas a uma FP. A restrição (3.18) especifica o número requerido de FPs. A restrição (3.19) assegura que a parte i pertença à FP j somente quando essa FP tenha sido formada. A última restrição (3.20) assegura a integralidade. Esse modelo necessita que seja pré-definida a quantidade de FPs. Como exemplo, considera-se a matriz da figura 15, procura-se formar p=2 FPs e as CMs correspondentes, utilizando a medida d_{ij} de distância de *Hamming*.

		partes				
		1	2	3	4	5
	1	0	4	0	4	3
	2	4	0	4	0	1
[dij] =	3	0	4	0	4	3
	4	4	0	4	0	1
	5	3	1	3	1	0

partes

Figura 15: Matriz inicial no exemplo do Método das p-Medianas

Resolvendo o modelo das p-Medianas, obtém-se a solução a seguir:

$x_{11} = 1,\ x_{31} = 1,$

$x_{24} = 1,\ x_{44} = 1, x_{54} = 1.$

Baseado na definição de x_{ij}, duas FPs são formadas: $FP-1 = \{1,3\}$ e $FP-2 = \{2,4,5\}$. Para as FPs formadas, as CMs correspondentes são $CM-1 = \{2,4\}$ e $CM-2 = \{1,3\}$.

3.2.3.2.b Modelo Generalizado das p-Medianas O modelo das p-medianas, apresentado anteriormente, limita-se a trabalhar com partes que pertençam apenas a um plano de processo. Kusiak modificou o modelo relaxando essa restrição, podendo ser considerado para cada parte, mais de um plano de processo. Além disso, um custo de produção é associado a cada plano de processo. O objetivo é minimizar o total das somas das distâncias e os custos de produção.

Considerando F_k o conjunto dos planos de processo para a parte k, tal que $k = 1,\ldots,l$, p é o número de FPs desejado, d_{ij} é a distância entre o plano de processo i e j e c_j é o custo de produção do plano de processo j.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} + \sum_{j=1}^{n} c_j x_{ij} \tag{3.21}$$

Sujeito a

$$\sum_{i \in F_k} \sum_{j=1}^{n} x_{ij} = 1 \text{ para todo } k = 1, \ldots, l \tag{3.22}$$

$$\sum_{j=1}^{n} x_{ij}^{n} \leq p \tag{3.23}$$

$$x_{ij} \leq x_{ji} \text{ para todo } i = 1, \ldots, n \; j = 1, \ldots, n \tag{3.24}$$

$$x_{ij} = 0, 1 \text{ para todo } i = 1, \ldots, n \; j = 1, \ldots, n \tag{3.25}$$

A restrição 3.22 assegura que, para cada parte (um conjunto de planos de processo), somente um plano é selecionado; a restrição 3.23 impõe um limite superior para o número das FPs; restrições 3.24 e 3.25 correspondem às restrições 3.19 e 3.20 do modelo 3.16 (JHA, 1991).

3.2.3.2.c Programação Quadrática Esse modelo permite que se possa lidar com número restrito de agrupamentos de diversos tamanhos. Esse modelo foi desenvolvido por Kusiak *et al* (JHA, 1991) e sua formulação pode ser expressa na forma de programação quadrática 0-1:

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{l=1}^{p} d_{ij} x_{il} x_{jl} \tag{3.26}$$

Sujeito a

$$\sum_{j=1}^{p} x_{ij} = 1 \text{ para todo } i = 1, \ldots, n \tag{3.27}$$

$$\sum_{i=1}^{n} x_{ij} = m_j \text{ para todo } j = 1, \ldots, p \tag{3.28}$$

$$x_{ij} = 0, 1 \text{ para todo } i = 1, \ldots, n \text{ e } j = 1, \ldots, p \tag{3.29}$$

A restrição 3.27 garante que cada parte pertença a uma única FP; restrição 3.28 impõe que a FP j contenha exatamente m_j partes; a última restrição garante integralidade.

3.2.3.3 Formulação por Grafos

Na formulação por grafos a matriz incidente é representada por um grafo, no qual são definidas as FPs e CMs. (JHA, 1991). Como exemplo, pode-se citar a modelagem por Grafos Bipartidos. No Grafo Bipartido os nós podem ser divididos em dois subconjuntos: um de partes e outro de máquinas. Na figura 16 é apresentado o grafo referentes à matriz da figura 12 (a). Esse grafo pode ser decomposto em 2 grafos disjuntos, como apresentado na figura 17, representando as FPs 1 e 2.

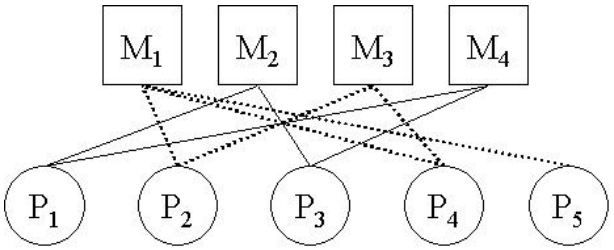


Figura 16: Grafo Bipartido representando a matriz da figura 12(a). Os vértices M_i representam as máquinas e os vértices P_i representam as partes. Os arcos desenhados com linhas contínuas representam a $FP(1)$, os com linhas tracejadas a $FP(2)$.

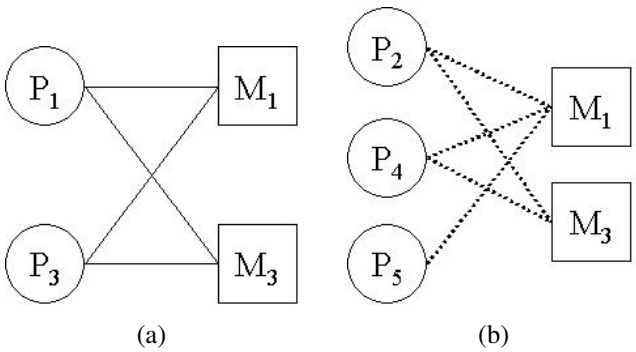


Figura 17: Dois grafos bipartidos disjuntos obtidos a partir do grafo da figura 16. Os vértices M_i representam as máquinas e os vértices P_i representam as partes. O grafo em (a) representa a $FP(1)$ e o grafo em (b) a $FP(2)$.

4 SISTEMA DE MANUFATURA FLEXÍVEL

A aplicação de Tecnologia de Grupo proporciona uma estruturação dos dados na organização, permitindo que os produtos e recursos sejam eficientemente gerenciados. A chave para isso é a formação das Famílias de Partes, o que possibilita ganhos econômicos na produção pela implantação da Manufatura Celular.

Manufatura celular é uma aplicação de TG onde máquinas dissimilares são agrupadas em células de manufatura, cada qual com a capacidade de produzir um determinado tipo de parte, FP ou grupo de FPs diferentes. Os objetivos da manufatura celular são similares aos da TG: redução dos tempos referentes à produção das partes (*setup*, manipulação de materiais, etc), aumento da qualidade dos produtos, simplificação do escalonamento, entre outros (GROOVER, 2001).

O conceito de Sistema de Manufatura Flexível (SMF) surge com o desenvolvimento dos sistemas de informação, aliado aos avanços tecnológicos no ambiente produtivo e à aplicação dos conceitos de TG e manufatura celular. Tanto na manufatura celular quanto nos SMFs, o elemento principal é a flexibilidade operacional, a qual tem reflexo direto na capacidade produtiva.

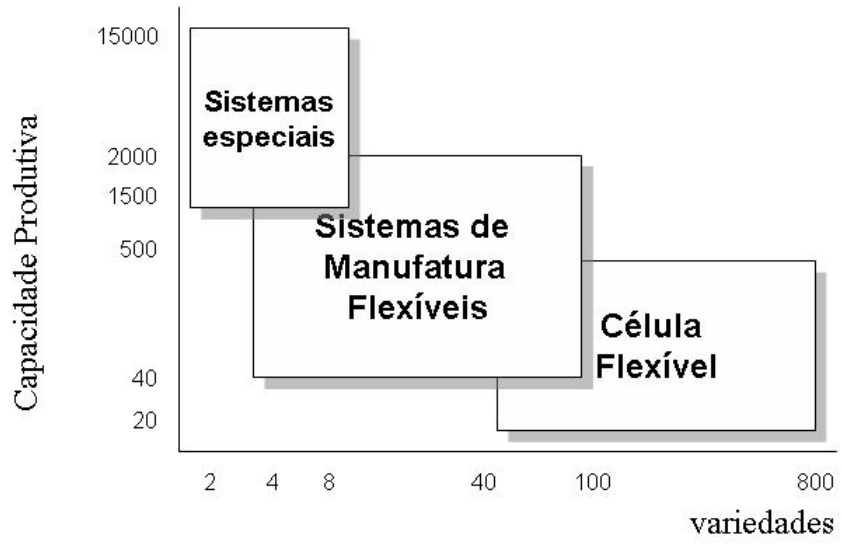


Figura 18: Tipos de Sistema de Manufatura Celular em relação à capacidade produtiva Fonte: (LORINI, 1993)

Neste capítulo, será abordado o conceito de SMF, que inicia com a definição das Células de Manufatura Flexíveis.

4.1 Célula de Manufatura Flexível

Quando se tem definidas as FPs, seja pelo método de Inspeção visual, Classificação e Codificação ou Análise do Fluxo de produção, pode-se formar Células de Manufatura Flexíveis (CMF). As CMFs possuem a capacidade de serem facilmente modificadas, de forma a atender a necessidade de produção de um novo tipo de produto, com o mínimo de tempo de *setup* das máquinas (LORINI, 1993).

Uma CMF pode ser física ou logicamente definida. Uma célula física é um conjunto de máquinas reunidas em um determinado local, sob uma determinada disposição, de modo a reunir uma série de facilidades para o processamento de um determinado produto. Uma CMF lógica (ou virtual) consiste de um seqüenciamento dos produtos na linha de produção, sem considerar a localização física das máquinas.

A flexibilidade da CMF está baseada em três componentes técnicos presentes na automação flexível (LORINI, 1993):

1. um grupo de máquinas de Controle Numérico (NC);
2. um sistema automático de transporte e manuseio de materiais;
3. um sistema computacional para o controle da CMF.

Tanto uma CMF quanto um SMF são considerados o mais alto grau de automação industrial pela presença desses três componentes técnicos. A capacidade de construir uma CMF é o ponto de partida para a implementação de um SMF na organização.

4.2 Conceito de SMF

Um Sistema de Manufatura Flexível pode ser identificado como a mais sofisticada implementação da manufatura celular, apoiado pelo intensivo uso de sistemas de informação. Esse sistema de informação extende-se ao controle dos equipamentos de produção, armazenagem e transporte de materiais, permitindo a integração do setor produtivo com os diversos setores de toda a organização

A definição mais tradicional de um SMF é a que envolve um conjunto de estações de processamento (geralmente do tipo CNC (*Computer Numerical Control*)), um sistema automático de transporte e manuseio de materiais que interconecta essas máquinas e um sistema computacional que faz o controle das máquinas e do sistema de transporte (GROOVER, 2001) (LORINI, 1993) (KUSIAK, 1986) (KAIGHOBADI; VENKATESH, 1993) (GUNASEKARAN; MARTIKAINEN; YLI-OLLI, 1993), sendo o sistema operado por técnicos especializados. Esse sistema é dito flexível por ser capaz de processar uma variedade de diferentes tipos de partes simultaneamente nas estações de trabalho, além de tornar possível o ajuste do *mix* de produção para se adequar a mudanças na demanda.

4.3 Histórico dos Sistemas de Manufatura Flexíveis

O conceito de SMF surgiu na década de 60, quando os princípios de TG estavam sendo aplicados em diversas plantas na América, Europa (principalmente) e Ásia e, concomitantemente, os avanços tecnológicos permitiram o surgimento das máquinas NC. O conceito de SMF foi creditado a David Williamson, na época, engenheiro da Molins, a qual patenteou o conceito chamando de *System 24* (porque acreditava-se que, agrupando máquinas e ferramentas, o sistema poderia trabalhar 24hs/dia, sem interferência humana). Esse conceito incluía controle computacional, máquinas NC e magazines capazes de armazenar diferentes tipos de ferramentas (GROOVER, 2001).

Um dos primeiros SMF instalados nos EUA foi o da Ingersoll-Rand Company em Roanoke, Virginia, no final dos anos 60. Outros sistemas foram introduzidos depois, a maioria em grandes companhias, como Caterpillar, John Deere e General Electric Co., as quais possuíam poder para realizar o investimento na implantação do sistema, além de possuirem experiência com máquinas NC. Foram instalados também SMFs na República Federal da Alemanha (1969), na União Soviética (em 1972) e no Japão (início dos anos 70) (GROOVER, 2001). Nos EUA, os usuários de SMF abrangiam indústrias como a Avco Lycoming, General Eletric, Pratt & Whitney (motores para o setor aeroespacial), Boeing, FMC Corporation, General Dynamics, McDonnell Douglas (componentes aeroespaciais), G.E. Medicals Systems (partes para equipamentos médicos), Xerox (Painéis para copiadores) entre outras indústrias. Atualmente, as grandes empresas montadoras mundiais possuem instalações baseadas nos conceitos de Manufatura Flexível, como o Grupo Chrysler, o Grupo Ford General Motors. Indústrias como a Grafton (Farmacêuticas), Hitachi (aparelhos eletrônicos), Motorola (circuitos impressos e equipamentos eletrônicos) possuem SMFs instalados, utilizando um Sistema de Manufatura Integrado por Computador (CIM).

4.4 Flexibilidade

O fator que torna atrativo o investimento em um ambiente de SMF é justamente a flexibilidade do sistema em responder às mudanças no ambiente externo. Essa flexibilidade, como mencionado anteriormente, está ligada à capacidade de mudança do tipo de parte a ser produzido, rápida modificação de instruções operacionais e pequenos tempos de *setup*.

Segundo Groover, um sistema de manufatura pode ser classificado como flexível ou não-flexível se satisfizer os seguintes critérios (GROOVER, 2001):

1. variedade de partes: um sistema deve ser capaz de processar diferentes tipos de partes, mesmo sem estarem agrupadas em lotes;
2. mudança de escalonamento: o sistema deve aceitar mudanças no escalonamento da produção, além de mudanças no *mix* e nas quantidades da produção;
3. recuperação de erros: o sistema deve ser capaz de recuperar-se de erros por mal funcionamento ou quebras de máquinas, sem parar completamente a produção;
4. novas partes: o sistema deve ser capaz de aceitar a produção de novos *designs* de partes com relativa facilidade.

Esses quatro critérios são os mais básicos para definir um sistema como flexível. Groover (GROOVER, 2001) apresenta os tipos de flexibilidade que um sistema pode possuir na tabela 1.

Tipo de Flexibilidade	Definição	Depende dos fatores:
Flexibilidade de Máquina	capacidade de uma estação de trabalho de se adaptar a grande variedade de operações e tipos de partes. Quão maior for essa variedade, maior será a flexibilidade da máquina.	• tempo de *setup*; • facilidade na reprogramação; • capacidade de armazenamento de ferramentas; • habilidade dos trabalhadores no sistema.
Flexibilidade de Produção	a variedade de partes que podem ser produzidas no sistema.	• flexibilidade das estações individuais.

Tipo de Flexibilidade	Definição	Depende dos fatores:
Flexibilidade de *Mix*	habilidade de trocar o *mix* de produtos enquanto mantém a mesma quantidade total de produção.	• similaridade entre as partes no *mix*; • flexibilidade da máquina.
Flexibilidade de Produto	facilidade com a qual mudanças no *desing* podem ser realizadas.	• quão similar o *design* da nova parte é em relação às FPs existentes (preparação da programação *off-line*). • flexibilidade da máquina.
Flexibilidade de Rota	capacidade de produzir partes em seqüências alternativas nas estações de trabalho, em resposta a panes no equipamento ou outras interrupções nas estações.	• similaridade das partes no *mix*; • similaridade das máquinas; duplicação de máquinas; • compartilhamento de ferramentas; • treino dos trabalhadores.
Flexibilidade de Volume	habilidade de produzir, economicamente, partes em grandes ou pequenas quantidades, dado um determinado investimento no sistema.	• nível de trabalho manual da produção; • quantidade de investimento em equipamentos.

Tipo de Flexibilidade	Definição	Depende dos fatores:
Flexibilidade de Expansão	facilidade com a qual o sistema pode ser expandido a fim de aumentar a quantidade total da produção	• custos da compra de máquinas; • facilidade do *layout* a ser expandido; • tipo de manipuladores de partes usados; • facilidade de adição de trabalhadores treinados.

Tabela 1: Tipos de Flexibilidade. Fonte: (GROOVER, 2001)

4.5 Tipos de SMF

Os SMFs são sistemas voltados para uma determinada aplicação, ou seja, para produzir uma determinada variedade de partes ou executar um determinado tipo de processo. Apesar disso, pode-se distinguir SMFs pelo tipo de operação executada: (*i*) operações de processamento ou (*ii*) operações de montagem. Uma maneira de classificar um SMF é segundo o número de estações de trabalho (GROOVER, 2001):

- Célula com uma Máquina: consiste da combinação de uma máquina CNC e de um sistema de armazenamento autônomo. Partes com o processamento completo são descarregadas da área de trabalho e novas partes são inseridas. A célula pode trabalhar no modo de produção em lotes ou na produção flexível. Na produção em lotes a célula processa um conjunto de partes de um mesmo tipo em um determinado número (tamanho do lote). Quando essas partes terminam de ser processadas, um novo lote com novas partes é inserido. Na produção flexível, a célula atende aos quatro critérios de flexibilidade discutidos no item 4.4.

- Célula de Manufatura Flexível: consiste de duas ou três estações de trabalho (CNC) e de um sistema de manipulação de materiais. Esse último é conectado a um sistema de carga e descarga, possuindo uma capacidade limitada de armazenamento. A CMF satisfaz os quatro critérios de flexibilidade discutidos anteriormente.

- Sistema de Manufatura Flexível: é composto por mais de quatro estações de trabalho

interligadas por um sistema automático de transporte de materiais, contando com um sistema distribuído de controle. A diferença entre uma CMF e um SMF é o número de estações de trabalho, além do fato de um SMF incluir estações de suporte à produção. Estas outras estações podem ser paletes para manipulação de materiais, máquinas de controle de qualidade, entre outras. Outra diferença é que o sistema de controle de um SMF é mais sofisticado que o de uma CMF.

Outra classificação pode ser feita com relação ao nível de flexibilidade de um SMF:

- SMF dedicado: também conhecido como Sistema de Manufatura Especial, é projetado para produzir uma quantidade limitada de tipos de partes, sendo que o total de tipos de partes é conhecido previamente. Sendo o *design* dos produtos estável, a manufatura destes é feita baseada na similaridade dos tipos de partes usando máquinas com propósito mais específico, aumentando assim, a taxa de produção.
- SMF não-dedicado ou de seqüência aleatória (*random order*): esse tipo de SMF é adequado para produção de uma grande variedade de tipos de partes, sendo que a variação de um tipo para o outro é substancial. Além disso, novos tipos de partes podem ser inseridos na produção, e o escalonamento está sujeito a mudanças repentinas. Para lidar com essas mudanças são utilizadas máquinas de propósito geral, capazes de processar diferentes tipos de partes em diferentes seqüências (seqüências aleatórias).

4.6 Componentes de um SMF

Como descrito no início do item 4.2, o conceito de SMF está relacionado a alguns componentes básicos: (*i*) um conjunto de estações de trabalho versáteis, (*ii*) um sistema automático de transporte e manipulação de materiais, (*iii*) um sistema de controle computacional e (*iv*) pessoas capazes de operar o sistema. Esses componentes serão descritos nas seções a seguir.

4.6.1 Estações de Trabalho

É o componente primordial do sistema, onde as operações de processamento e montagem são realizadas. A configuração das máquinas pode variar bastante, dependendo da aplicação para a qual o SMF foi projetado. Essas estações não necessariamente precisam executar somente essas operações. Alguns tipos de estações de trabalho são descritas a seguir (LORINI, 1993) (GROOVER, 2001):

- Estações de Carga e Descarga: estações responsáveis por inserir partes não processadas e retirar partes que tiveram seu processamento completado do sistema. Esse sistema pode ter controle automático ou manual, sendo que, no último caso, existe um sistema computacional para dar apoio ao operador, informando as partes a serem carregadas, paletas necessárias para a fixação da mesma, entre outras informações. Quando o procedimento de carga está concluído, o sistema de manipulação de materiais deve inserir a parte no sistema, com o cuidado de verificar se não existem trabalhadores na área. Isso exige que ambos os sistemas (manipulação e carga) estejam se comunicando.

- Estações de usinagem: as estações de trabalho são geralmente são máquinas do tipo CNC ou DNC (*Distributed Numerical Control*). Essas máquinas podem possuir um sistema automático de armazenamento e de troca de ferramentas, que realiza as trocas de acordo com a sua programação (*part-program*). As estações de usinagem podem ser classificadas de acordo com a operação que executam, como estações de furação, estações de torneamento, estações de fresamanto e estações mistas de torno-fresa.

- Estações de Montagem: estações de montagens flexíveis são desenvolvidas para substituir mão-de-obra humana na linha de montagem. Geralmente uma célula de montagem conta com robôs industriais programados para executar diversos tipos de tarefas.

- Outros tipos de estações e equipamentos: estações de inspeção de produtos são comuns em plantas que implementam SMF. Máquinas de mensuração de coordenadas, sondas de inspeção e visão computacional são exemplos de estações de inspeção.

4.6.2 Sistema de Transporte e Armazenamento de Materiais

O Sistema de Transporte e Manuseio de Materiais (STMM) em um SMF tem as seguintes funções:

- movimentação das partes entre as estações de trabalho: as partes devem ser movidas de uma estação até qualquer outra através de diferentes rotas que contemplem possíveis quebras ou máquinas ocupadas.

- manipulação de diferentes tipos de partes: para movimentar as partes, são utilizadas paletes com elementos de fixação que permitem a manipulação de diferentes tipos de materiais.

- armazenamento temporário: algumas das partes que se encontram no sistema ficam em

espera, aguardando que uma máquina fique disponível para seu processamento. Assim, cada estação deve contar com uma fila onde as partes aguardam.

- fácil acesso para estações de carga e descarga: o sistema de manipulação deve fazer interface com as estações de carga e descarga.
- compatibilidade com o controle computacional: o sistema de manipulação deve ser capaz de se comunicar com as estações de trabalho e as estações de carga e descarga, para que seja possível sincronizar as tarefas da produção.

Para implementar essas funções, os componentes geralmente usados são (LORINI, 1993):

- Sistemas Contínuos de Transporte: são sistemas de esteiras servindo cada estação, podendo ter diversas topologias, como "ramos" ou "laços".
- Veículos especiais: podem ser de dois tipos: (*i*) vagões industriais sobre trilhos ou (*ii*) AVGs (*Automated Guided Vehicles*). Esses últimos recebem instruções via ondas de rádio, ultra-som ou através de fios elétricos distribuídos pelo chão da fábrica. Os AVGs podem ser utilizados em sistemas de transporte verticais e horizontais, manipulação entre áreas de *buffers* e execução de atividades de armazenamento, seguindo trilhas complexas no leiaute do chão de fábrica.
- Robôs: podem ser utilizados para transportar e movimentar peças e para o manuseio de ferramentas. Um robô é um manipulador mecânico programável, capaz de mover-se em várias direções e equipado com dispositivos que permiter realizar tarefas que seriam executadas por um ser humano.
- Paletes e Elementos de Fixação: as partes a serem processadas são fixadas em paletes, os quais são manipulados até as estações de trabalho. Esses paletes podem ser unitários ou capazes de comportar lotes inteiros de partes. Dependendo do tipo de parte, elementos de fixação especiais são necessários para prendê-las ao palete. Esses elementos devem ser configuráveis, aceitando diversos tipos de partes.

O STMM pode ser dividido em duas partes: sistema primário e sistema secundário. O sistema primário é responsável por movimentar as partes por entre as máquinas da fábrica. O sistema secundário geralmente se localiza nas estações de trabalho de carga e descarga, sendo responsável por manipular a parte do sistema primário para essas estações. O sistema secundário pode ter função de reorientar a parte e armazenar partes que aguardem processamento.

Um STMM pode ter as seguintes configurações de *layout*:

- *layout* em linha: as máquinas e o sistema de manipulação são dispostos em linha reta. As partes se movimentam em uma única direção, obedecendo uma determinada seqüência, geralmente sem fluxo reverso. Para dar maior flexibilidade de roteamento às partes, pode ser instalado nesse sistema um vagão que se movimente em ambos os sentidos da linha do sistema.

- *layout* em laço (*loop*): as máquinas são organizadas em laço, sendo interligadas por um STMM de mesmo formato. As partes geralmente circulam em um determinado sentido no laço, parando e sendo carregadas nas máquinas definidas por um STMM secundário. As estações de carga e descarga geralmente estão localizadas nas extremidades do laço. Este *layout* pode ser implementado na forma retangular.

- *layout* em escada: este tipo de *layout* consiste de um laço retangular em torno das máquinas. Para cada máquina, tem-se uma linha que as une aos lados do retângulo. Esse tipo de *layout* permite uma movimentação em vários sentidos, reduzindo o congestionamento na célula.

- *layout* misto (*open field*): consiste de uma mistura de *layouts* de laços e escadas, geralmente usado para processar grandes FPs. O número de máquinas diferentes pode ser limitado, dado que partes são roteadas para a máquina que estiver disponível com maior facilidade.

- célula centrada em robô: conta com um ou mais robôs para realizar a manipulação das partes nas máquinas. A utilização de robôs permite que vários tipos de partes diferentes possam ser manipulados com facilidade.

4.6.3 Sistema Computacional

O sistema computacional é um dos mais importantes componentes do SMF, pois realiza a integração e o monitoramento das estações de trabalho do STMM e de outros componentes de hardware que possam existir na planta. Um sistema computacional de um SMF consiste de um computador central que se comunica com microcomputadores que controlam as máquinas e os outros componentes. As funções atribuídas ao sistema computacional em um SMF figuram entre (GROOVER, 2001) (LORINI, 1993):

- Controle das Estações de Trabalho: em SMF completamente automatizados, cada estação de trabalho possui um controle computadorizado próprio. Para sistema de usinagem, geralmente são usadas máquinas CNC.

- Distribuição das informações de controle às estações: para que as estações de trabalho operem individualmente, é necessário que exista um sistema central que distribua as ordens. Podem existir computadores centrais de cada célula, responsáveis por receber as informações, fazendo o carregamento dos *part-programs* e repassando as ordens às máquinas definidas.

- Controle da Produção: realização do controle do *mix* e do fluxo das partes que são postas no sistema. O controle da produção envolve dados como quantidade de partes a ser produzida por dia, quantidades de partes não processadas e de paletes disponíveis, disponibilidade de máquinas, entre outras.

- Controle de Tráfego: gerenciamento dos STMMs primário e secundário, implementado através de interruptores em pontos chaves do STMM (como pontos de junção de fluxo de partes ou entroncamento de ramos), juntamente com controle do sistema de carga e descarga.

- Controle de carros (*shuttle*): refere-se ao controle do STMM secundário e sua sincronização com a operação das máquinas e do STMM primário.

- Monitoramento de partes: monitoramento do estado de cada palete e de cada tipo de parte que está sendo produzido.

- Controle das ferramentas: em sistemas de usinagem, ferramentas de corte são utilizadas nas estações de trabalho. O controle de ferramentas deve ter a relação do estado das ferramentas de cada máquina. Caso alguma ferramenta necessária para o processamento de uma parte não esteja presente na estação de trabalho para a qual a parte foi roteada, o sistema deve reorientar a parte para uma estação com essas ferramentas disponíveis ou avisar o operador de que essas devem ser carregadas. O sistema deve monitorar também a vida útil de cada ferramenta, baseado em dados especificados pelo usuário. Quando a utilização de uma determinada ferramenta atingir o fim da vida útil, o operador deve ser comunicado para fazer a troca.

- Monitoramento e relatório de desempenho: o sistema computacional deve monitorar a operação do SMF, coletando os dados e produzindo relatórios de desempenho para o gerenciamento da planta.

- Diagnósticos: o sistema pode indicar a causa (ou possíveis causas) de uma falha no sistema, para que o operador possa tomar as medidas corretivas. O objetivo maior desse sistema é reduzir o tempo ocioso devido à quebra de maquinário.

4.7 Níveis de SMF

Com o apoio dos sistemas computacionais, é possível fazer o gerenciamento de um SMF considerando o modelo conceitual de Stecke (Stecke *apud* (GÓMEZ, 1996)), o qual é dividido em 4 fases:

1. Projeto (Planejamento estratégico): nesta etapa são abordados problemas organizacionais do tipo seleção dos produtos a serem produzidos, escolha do maquinário, do leiaute das máquinas, sistema de transporte, entre outros.
2. Planejamento do Processo (Planejamento Tático): organização da produção a fim de obter um uso eficiente dos recursos do sistema (máquinas, ferramentas, instalações, etc).
3. Escalonamento da Produção: definição de uma seqüência de operações para as máquinas de forma a atingir um determinado objetivo.
4. Controle e Monitoramento: gerenciamento da produção em tempo real.

O modelo de Stecke foi utilizado por Sodhi, Askin e Sen (Sodhi *apud* (GÓMEZ, 1996)) para o controle de um SMF, propondo uma divisão do modelo em duas fases:

- Fase Pré-operacional: ligada a atividades anteriores ao início da produção, ou seja, *setup*, carregamento de ferramentas etc.
- fase operacional : ligada a atividades características posteriores ao início da produção: tempo real, controle ”on-line”, movimento de partes etc.

A figura 19 mostra o Modelo de Stecke utilizado por Sodhi *et al*.

4.8 Vantagens da implantação de um SMF

A implementação de um SMF é uma tarefa onerosa, pois demanda: (*i*) significante investimento monetário, dado os equipamentos de produção e sistemas de informação necessários e (*ii*) pessoal capacitado a operar o novo sistema, bem mais sofisticado que sistemas manuais tradicionais. Mesmo requerendo tantos investimentos, segundo Groover (GROOVER, 2001), os benefícios da implantação de um SMF incluem:

- aumento da utilização das máquinas: SMF possibilita uma utilização maior das máquinas dado: (*i*) 24 horas/dia de operação; (*ii*) troca automática de ferramentas; (*iii*) manipulação

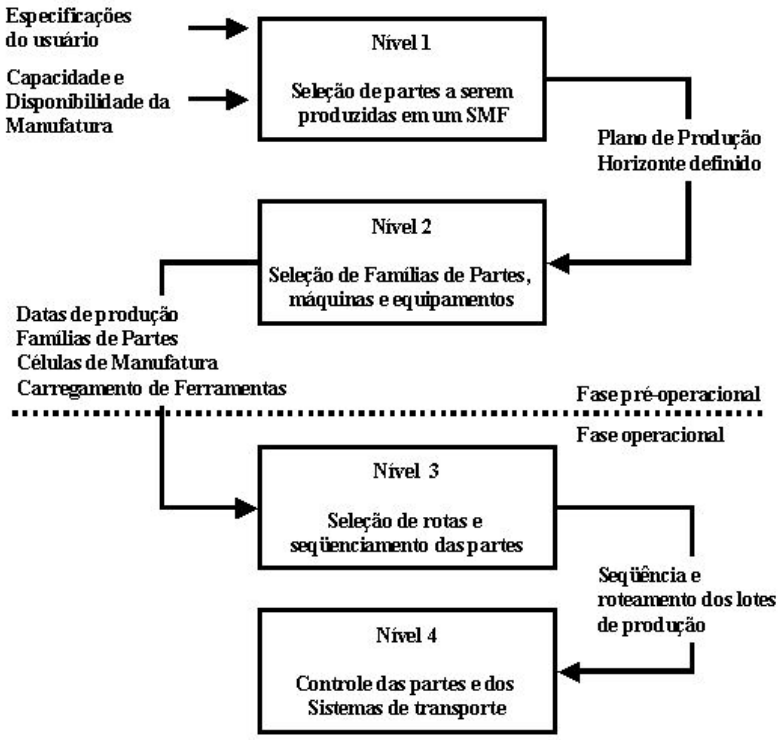


Figura 19: Modelo de Stecke utilizado por Sodhi *et al.* Fonte: (GÓMEZ, 1996)

automática de materiais; (*iv*) escalonamento dinâmico das partes, que leva em conta irregularidades da produção.

- a maior utilização das máquinas incorre na redução do número das mesmas, o que também reduz o espaço físico requerido na planta para a fabricação dos produtos.

- grande capacidade de resposta às mudanças como: (*i*) mudanças de *design* das partes; (*ii*) mudanças no escalonamento da produção; (*iii*) quebra de máquinas e (*iv*) ajuste no escalonamento da produção para atender clientes preferenciais.

- dado que diferentes partes são processadas juntas, o *work-in-process* é menor e o inventário de partes no início e no fim da produção podem ser reduzidos.

- como o nível de automação de um SMF permite que ele opere por maiores períodos de tempo sem intervenção humana, possibilitando a produção em períodos antes não atendidos;

5 PROBLEMAS ABORDADOS

5.1 Problema de Seleção de Partes

O Problema de Seleção de Partes (PSP) é um problema clássico da TG, onde se busca agrupar partes segundo algum critério de similaridade. Para tanto, existem várias ferramentas utilizadas, como apresentado no capítulo 3.

Uma das representações bastante utilizadas é a formulação matricial, onde as partes e seus atributos são descritos em uma matriz incidente do tipo máquina $\times$ parte. O PSP pode ser definido da seguinte forma:

Dado uma matriz inicial do tipo parte-ferramenta $A = [a_{ij}]$, com dimensões $N \times M$, onde N é o número de partes e M é o número de máquinas, define-se

$$a_{ij} = \begin{cases} 1 & \text{caso a máquina } i \text{ seja usada para processar a parte } j, \\ 0 & \text{caso contrário.} \end{cases}$$

Deve-se decompor A em $A_1, \ldots, A_k$ submatrizes, onde cada submatriz é composta por um conjunto de máquinas (Célula de Manufatura) e um conjunto de partes (Família de Partes). É desejável que os agrupamentos encontrados promovam o mínimo de movimentação intercelular, ou seja, que essa decomposição resulte em submatrizes mutuamente disjuntas. Pode ser considerada a restrição de número máximo de máquinas em cada célula, o que afeta a formação das FPs.

O PSP pode ser visto também pela ótica de geração de FPs e Família de Ferramentas (FF). Considera-se a matriz $A = [a_{ij}]$, onde $a_{ij} = 1$ indica que a parte i necessita da ferramenta j para seu processamento. A restrição a ser considerada, para este problema, é a quantidade máxima de ferramentas que uma máquina pode carregar no seu magazine (GÓMEZ, 1996) (HWANG; SHOGAN, 1987).

Dado o número de possíveis resultados para uma instância deste tipo de problema, o PSP é considerado um problema complexo de ser computacionalmente resolvido, sendo definido

como um problema *NP Completo* (KUSIAK; CHOW, 1987).

As técnicas mais utilizadas no tratamento desse problema são apresentadas no capítulo 3, destacando-se o Algoritmo de Agrupamento, Programação Matemática e modelagem por grafos.

5.2 Problema de Escalonamento

O Problema de Escalonamento (PE) de uma maneira geral visa alocar recursos limitados a tarefas que ocorrem no tempo (PINEDO, 1995). Na área de produção esses recursos podem ser máquinas, ferramentas, paletes, entre outros; as tarefas podem ser operações requeridas por uma parte. Existem vários tipos de PEs considerados na literatura, podendo ser classificados de acordo com o ambiente de manufatura, os recursos disponíveis e as restrições consideradas. Nos PEs considerados em manufatura, existe um número finito de partes que deve ser direcionado a um conjunto finito de estações de trabalho de forma a atingir um determinado objetivo. Associadas às partes, pode-se considerar informações como tempo de processamento, data de entrega, data de liberação e um peso (que pode representar um fator de prioridade).

Quanto ao ambiente, podem ser considerados ambientes com uma única estação de trabalho, estações idênticas em paralelo, estações diferentes em paralelo, entre outras. Os tipos mais conhecidos de ambiente podem ser definidos, em linhas gerais, como segue:

- *Flow Shop*: existem *m* estações em série; cada parte deve ser processada em cada uma das máquinas; todas as partes possuem a mesma rota; após o processamento por uma máquina, uma parte entra na fila de espera pela próxima máquina; uma parte não pode passar a frente da outra enquanto espera na fila. Um *flow shop* flexível é uma generalização do *flow shop* convencional aplicado a um ambiente com máquinas paralelas.

- *Open Shop*: consideram-se *m* máquinas, sendo que todas as partes devem ser processadas em cada uma das máquinas; algumas partes podem possuir tempo de processamento zero; não existem restrições quanto a rota das partes nas máquinas; o sequenciador pode definir a rota das partes e diferentes partes possuem diferentes rotas.

- *Job Shop*: existem *m* máquinas e cada parte possui sua própria rota; as partes não necessariamente necessitam ser processadas por todas as máquinas.

Podem ser consideradas restrições de processamento como *setup* dependente de seqüência,

preempção, precedência entre partes, reentrada na linha de produção e outras mais, dependendo do ambiente modelado (PINEDO, 1995).

Neste trabalho será abordado o Problema de Escalonamento em *Job Shop*, descrito no item a seguir.

5.2.1 Definição do Problema de Escalonamento em um *Job Shop*

O Problema de Escalonamento em um Job Shop (PEJS), pode ser descrito da seguinte forma (NOWICKI; SMUTNICKI, 1996): dado um conjunto de partes (*jobs*) $J = \{1, \ldots, n\}$ que deve ser processado em um conjunto M de máquinas, sendo que as partes requerem um conjunto de operações $O = \{1, \ldots, o\}$. A parte j consiste de uma seqüência de operações indexadas por $(l_{j-1} + 1, \ldots, l_{j-1} + o_j)$, as quais devem ser processadas nessa ordem, onde $l_j = \sum_{i=1}^{j} o_i$ é o número total de operações das primeiras j partes, onde $j = 1, \ldots, n$ $(l_0 = 0)$ e $\sum_{i=1}^{n} o_i = o$. A operação i deve ser processada na máquina $\mu_i \in M$ durante um tempo de processamento ininterrupto $p_i > 0$, $i \in O$. O conjunto O de operações pode ser dividido em subconjuntos M_k das operações processadas na máquina k. Assume-se que qualquer operação sucessiva de uma mesma parte deve ser processada em máquinas diferentes. Cada máquina pode processar no máximo uma parte por vez. Um escalonamento viável é definido por tempos de início $S_i \geq 0$, $i \in O$, tal que as restrições citadas sejam satisfeitas. O problema é encontrar um escalonamento viável que minimize o tempo total de produção (*makespan*) definido como $max_{i \in O}(S_i + p_i)$.

O primeiro método de modelagem de escalonamento proposto foram os Gráficos de Gantt, criados por Henry L. Gantt (1861-1919), sendo muito utilizados para a representação da execução de tarefas no tempo. Roy e Sussmann (JAIN; MEERAN, 1999) (BŁAŻEWICZ; DOMSCHKE; PESCH, 1996) propuseram o modelo de grafos disjuntos $\mathscr{G} = \{\mathscr{N}, \mathscr{A}, \mathscr{E}\}$, que atualmente é considerada a modelagem mais tradicional. O PEJS é representado por um grafo com pesos nos nós, existindo um vértice para cada operação onde $\mathscr{N}$ é o conjunto de nós representando operações a serem processadas no conjunto de máquinas $\mathscr{M}$. São adicionados dois nós fictícios 0 e n ao conjunto $\mathscr{N}$, os quais correspondem ao início e ao final das operações, conhecidos como fonte e destino: $\mathscr{N} = \{0, 1, 2, \ldots, n\}$. O peso positivo em cada um dos j nós é referente ao tempo de processamento p_j da operação correspondente, onde $p_0 = p_n = 0$. Os tempos de início e de final de processamento dos nós são os tempos de início e fim do PEJS. O nó 0 é conectado à operação inicial de cada parte e a operação final de cada uma é conectada ao nó n. O conjunto $\mathscr{A}$ de arcos conjuntivos dirigidos representa a precedência entre as operações, tal que $(i, j) \in \mathscr{A}$ indica que a operação i é predecessora da operação j $(i \prec j)$ na cadeia de operações. Restrições de capacidade asseguram que duas operações que necessitem de uma

mesma máquina não sejam executadas simultaneamente. Tais operações pertencem ao conjunto $\mathscr{E}$, onde cada elemento de $\mathscr{E}$ é associado com um par de arcos disjuntivos, requerendo uma máquina comum, tal que $[i,j] = \{(i \prec j),(j \prec i)\}$ e $\{i,j \in O\}$. Um exemplo de um grafo disjunto 4×3 é apresentado na figura 20, que representa o problema mostrado na tabela 2. Os arcos conjuntivos (membros de $\mathscr{A}$) são mostrados em linhas completas, enquanto que os arcos disjuntivos (membros de $\mathscr{E}$) são mostrados em linhas tracejadas.

Tabela 2: Tempos de Processamento e ordem das operações para um problema de produção de 4 produtos em 3 máquinas. Fonte: (JAIN; MEERAN, 1999)

número da operação	parte							
	J_1		J_2		J_3		J_4	
	$\mathscr{M}_i$	p_i	$\mathscr{M}_i$	p_i	$\mathscr{M}_i$	τ_i	$\mathscr{M}_i$	p_i
1-4	1	5	3	7	1	1	2	4
5-8	2	8	1	3	3	7	3	11
9-12	3	2	2	9	2	10	1	7

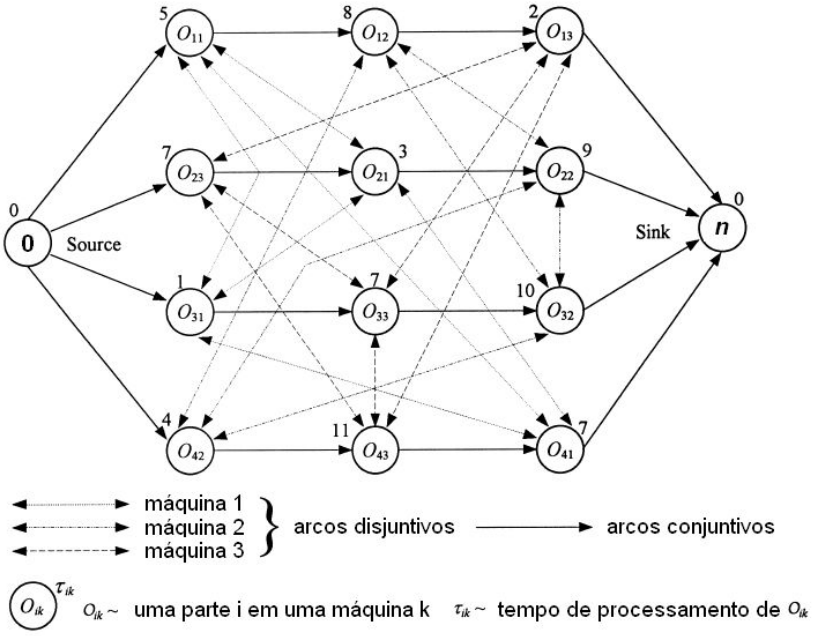


Figura 20: Grafo disjuntivo representando a um problema de produção de 4 produtos em 3 máquinas, descrito na tabela 2

O Problema de Escalonamento em um *Job Shop* também pode ser representado por um modelo de permutação em grafo proposto por Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 1996). Neste modelo o escalonamento é representado pela ordem de processamento das operações nas máquinas, através de uma m-upla $\pi = (\pi_1, \ldots, \pi_m)$, onde $\pi_k = (\pi_k(1), \ldots, \pi_k(m_k))$ é a permutação em M_k, $k \in M$; $\pi_k(i)$ representa um elemento de M_k, o qual se encontra na posição i em π_k. Seja Π_k o conjunto de todas as permutações em M_k, então

$\pi \in \Pi = \Pi_1 \times \Pi_2 \times \ldots \times \Pi_m$. Para a ordem de processamento π cria-se um grafo dirigido $G(\pi) = (O, R \cup E(\pi))$ com um conjunto O de nós e um conjunto de arcos $R \cup E(\pi)$, onde

$$R = \bigcup_{j=1}^{n} \bigcup_{i=1}^{O_j - 1} \{(l_{j-1} + i,\ l_{j-1} + i + 1)\} \text{ e}$$

$$E(\pi) = \bigcup_{k=1}^{m} \bigcup_{i=1}^{m_k - 1} \{(\pi_k(i), \pi_k(i+1))\}.$$

Os arcos do conjunto R representam a precedência das operações de cada parte, enquanto que o conjunto $E(\pi)$ representa a ordem de processamento das operações nas máquinas. Cada nó $i \in O$ tem um peso p_i referente ao tempo de processamento da operação i. Uma ordem de processamento π é viável se o grafo $G(\pi)$ não contém um ciclo. Para uma ordem de processamento $\pi \in \Pi$ viável, $r_\pi(i)$ representa o caminho mais longo no grafo $G(\pi)$ até o vértice i, incluindo o peso no vértice. É sabido que o escalonamento $S_i = r_\pi(i) - p_i,\ i \in O$ é um escalonamento viável. Nowicki e Smutnicki consideram que o *makespan* para um escalonamento viável é igual ao tamanho do caminho mais longo em $G(\pi)$. Define-se caminho crítico como o conjunto de operações que mais contribuem para o aumento do *makespan*, no grafo $G(\pi)$. O tamanho do caminho crítico é igual ao valor do *makespan*. O objetivo do PEJS é encontrar um escalonamento viável que minimize o tamanho do caminho crítico.

Um caminho crítico em $G(\pi)$ pode ser representado por $u^\pi = (u_1^\pi, \ldots, u_s^\pi)$ onde $u_i^\pi \in O$, sendo s o número de vértices no caminho crítico (NOWICKI; SMUTNICKI, 1996), tal que $1 \leq i \leq s$. O caminho crítico pode ser decomposto em subseqüências $B_1, \ldots, B_s$, chamadas blocos de u^π. Esses blocos podem ser definidos como a máxima subseqüência de u^π a qual contém operações processadas em uma mesma máquina.

Para exemplificar essa modelagem, é apresentado um exemplo de pequeno porte a seguir. Consideram-se 3 partes, 12 operações e 2 máquinas, como descrito na tabela 3. Considerando a ordem de processamento viável $\pi = (\pi_1, \pi_2)$, onde $\pi_1 = (1,8,3,11,5,10,7)$ e $\pi_2 = (2,4,9,12,6)$. O grafo dirigido correspondente é mostrado na figura 21 e o gráfico de Gantt na figura 22. $G(\pi)$ contém um único caminho crítico $u^\pi = (1,8,3,4,9,12,6,7)$, sendo decomposto em $s = 3$ blocos: $B_1 = (1,8,3)$, $B_2 = (4,9,12,6)$, $B_3 = (7)$.

O PEJS é um problema de difícil solução, considerando apenas o objetivo de minimizar o *makespan*, uma instância de $n \times m$ possui um limite superior de $(n!)^m$, ou seja, para um problema de 20×10, deveriam ser analisadas 7.2651×10^{183} possíveis soluções, para então se descobrir o ótimo do problema. Dado essa explosão fatorial, o PEJS é caracterizado como um

Tabela 3: Exemplo de modelagem por grafos dirigidos, considerando a produção de oito operações em duas máquinas.

partes	operação						
1	1	2	3	4	5	6	7
μ_i	1	2	1	2	1	2	1
p_i	2	1	2	2	1	1	1
2	8	9	10				
μ_i	1	2	1				
p_i	2	2	1				
3	11	12					
μ_i	1	2					
p_i	2	2					

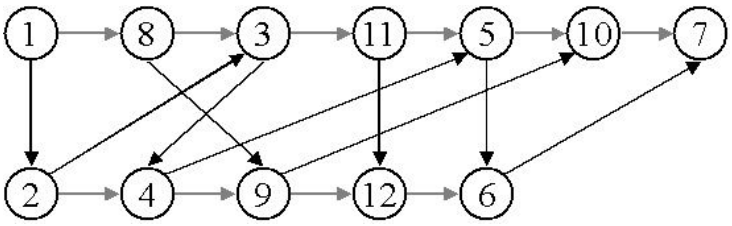


Figura 21: Grafo dirigido $G(\pi)$ para uma instância de PEJS com 3 partes, 2 máquinas e 12 operações. As linhas escuras representam os arcos do conjunto *R* e as linhas claras representam os arcos do conjunto $E(\pi)$.

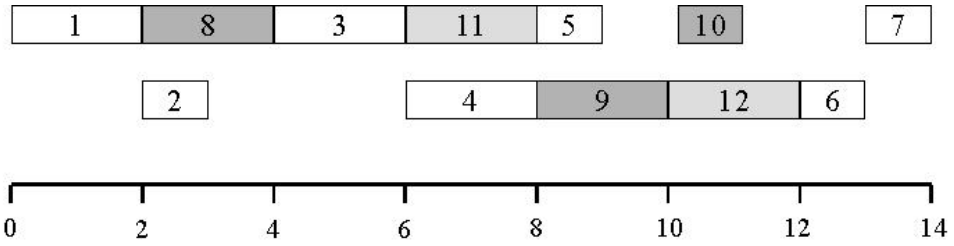


Figura 22: Gráfico de Gantt para uma instância de PEJS com 3 partes, 2 máquinas e 12 operações.

problema NP-Difícil. Poucas instâncias de PEJS podem ser resolvidas eficientemente, como por exemplo, PEJSs onde tem-se 2 máquinas e todas as partes possuem no máximo 2 operações (BŁAŻEWICZ; DOMSCHKE; PESCH, 1996).

O estudo do PEJS levou à utilização de métodos heurísticos e meta-heurísticos, como será visto nas seções a seguir. Os métodos heurísticos levam em conta a modelagem por grafos disjuntos do PEJS para definir os operadores de geração de vizinhanças de busca. Błażewicz, Domschke e Pesch (BŁAŻEWICZ; DOMSCHKE; PESCH, 1996) nomeia os principais tipos de operadores de vizinhança na literatura:

- $N1$: A transição da solução atual para a nova solução é gerada através da troca, na

representação por grafo disjuntivo da solução inicial, do arco disjuntivo (i, j) de um caminho crítico por um arco oposto (j, i).

- $N2$: Considere uma solução viável e um arco crítico (i, j) definindo a ordem de processamento das operações i e j na mesma máquina m. Define-se $pred(i)$ e $suc(i)$ como sendo o predecessor e o sucessor de i na máquina m, respectivamente. Restringir a escolha dos arcos (i, j) para aqueles vértices tal que pelo menos um dos arcos $(pred(i), i)$ ou $(j, suc(j))$ não fazem parte do caminho mais longo. Reverter o arco (i, j) e, a seguir, os arcos $(pred(h), h)$ e $(k, suc(k))$ - dado que eles existam - onde h precede diretamente i em sua parte e k é o sucessor imediato de j em sua parte. Esses dois últimos devem ser revertidos se o *makespan* for reduzido.

- $N3$: Seja (i, j) um arco disjuntivo. Considere todas as permutações das três operações $\{pred(i), i, j\}$ e $\{i, j, suc(j)\}$ no qual (i, j) é invertido.

- $N4$: Considerando que um caminho crítico pode ser dividido em blocos, como apresentado por Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 1996), para todas as operações i em um bloco, mover i para o início ou para o final do mesmo bloco.

- $N5$: um movimento é definido pela troca de duas operações sucessivas i e j, onde i ou j é a primeira ou a última operação em um bloco que pertence a um caminho crítico. No primeiro bloco somente as duas últimas operações (e simetricamente no último bloco, somente as duas primeiras) são trocadas.

- $N6$: O vizinho x' de uma seqüência x é obtido pela troca de duas operações i e j em um bloco do caminho crítico. A operação j é a última no bloco e não existe caminho direto em x conectando a parte sucessora de i com j ou, a operação i é a primeira do bloco e não existe caminho direto em x conectando i com a parte predecessora de j.

5.2.2 Estado da Arte

Uma das primeiras proposições de PEJS foi feita por Roy e Sussmann (1964) (*apud* (JAIN; MEERAN, 1999)) através da representação por grafos disjuntivos. Na década de 60 os algoritmos enumerativos, apoiados por elaboradas análises matemáticas foram muito utilizados. A principal técnica enumerativa da época era o *Branch-and-Bound*, onde uma árvore representando as possíveis seqüências é construída dinamicamente (JAIN; MEERAN, 1999). Outra técnica que começou a ser utilizada nessa época é chamada de Regra de Prioridade de Despacho, dadas as limitações das técnicas enumerativas.

Durante os anos 70, os trabalhos se concentraram na área de complexidade do PEJS, mostrando que poucas instâncias desse problema poderiam ser tratadas de maneira eficiente. Os estudos a respeito de complexidade foram muito importantes pois demostraram a intratabilidade do problema, reforçando as pesquisas sobre os métodos aproximativos.

Nos anos 80, o foco estava na criação de técnicas de otimização para suprir deficiências dos métodos aproximativos que estavam sendo utilizados (como as Regras de Despacho). Os estudos sobre métodos de construção de boas heurísticas de busca local se intensificaram, juntamente com a aplicação dessas não só no PEJS mas também em outros problemas NP-Difíceis. Ao final dos anos 80 surgem heurísticas baseadas em fenômenos naturais e métodos inteligentes, como Satisfação de Restrições e Redes Neurais Artificiais.

No período do final dos anos 80 e início dos anos 90, surgiram métodos aproximativos importantes, como de Otimização em Passos Largos (*Large Step Optimization*), Busca Tabu, Têmpera Simulada, Algoritmos Genéticos e *Shifting Bottleneck*. Esses métodos consistem em uma meta-estratégia que controla uma heurística de busca local, fazendo com que soluções piores sejam escolhidas para escapar de mínimos locais. Nesse período, trabalhos importantes a respeito de geração de vizinhanças foram publicados por Grabowski *et al* (1988) e Van Laarhoven *et al* (1988) (*apud* (JAIN; MEERAN, 1999)).

Segundo Jain e Meeran (JAIN; MEERAN, 1999), nos anos 90, novos estudos a respeito da complexidade computacional referente à construção de métodos aproximativos foram realizados. É significativo o número de autores que aborda o PEJS através de técnicas híbridas, concentrando-se no uso de Algoritmos Genéticos, Têmpera Simulada e Busca Tabu. Nos itens a seguir serão abordadas essas técnicas e alguns métodos exatos utilizados anteriormente.

5.2.2.1 Métodos Exatos

Os métodos exatos foram a primeira solução estudada para o PEJS. Eles consistem basicamente de enumeração dos possíveis escalonamentos e avaliação de cada um. Como a complexidade do problema é muito alta, os esforços de desenvolvimento dessa técnica concentram-se em encontrar limites inferiores para o PEJS a fim de otimizar o processo enumerativo.

O método exato mais conhecido é o *Branch-and-Bound* (BB), que constrói dinamicamente uma árvore cujos nós representam todas as possíveis soluções para o PEJS. A partir de uma solução é gerada uma série de outras soluções para o problema, através da expansão dos ramos da árvore (*Branch*). Para reduzir o número de possibilidades a serem analisadas, existe um critério de "poda" (*Bound*), que impede que o algoritmo expanda um ramo da árvore se um

determinado limite inferior for ultrapassado (HILLIER; LIEBERMAN, 1988).

Em seus trabalhos, Błażewics, Domschke e Pesh (BŁAŻEWICZ; DOMSCHKE; PESCH, 1996) e Jain e Meeran (JAIN; MEERAN, 1999) citam diversas aplicações de BB e limites inferiores ao PEJS. Para sanar as limitações provenientes da complexidade do PEJS, limites inferiores para BB são gerados por técnicas heurísticas e são citadas várias técnicas híbridas com BB. Kuroda e Wang (KURODA; WANG, 1996) utilizam BB com informações de conjuntos Fuzzy. Torres e Lopez (TORRES; LOPEZ, 2000) propõem um algoritmo BB que considera propagação de restrições para tratar o PEJS considerando restrições de *Not-First/Not-Last*. Mascis e Pacciarelli (MASCIS; PACCIARELLI, 2002) estudam o PESJ com restrições de bloqueio e não-espera (*blocking* e *no-wait*), através da formulação de grafos disjuntivos, aplicado em um algoritmo de BB.

5.2.2.2 Regras de Prioridade de Despacho

Como um dos primeiros métodos aproximativos estudados, Regras de Prioridade de Despacho (RPD) são disciplinas pré-fixadas de liberação de partes no sistema produtivo. RPD associa prioridades à cada operação e escalona as partes de acordo com as prioridades mais altas. Błażewicz, Domschke e Pesh (BŁAŻEWICZ; DOMSCHKE; PESCH, 1996) apresentam as mais utilizadas RPDs. Engell, Herrmann e Moser comparam várias regras de despacho (JOSHI; SMITH, 1994).

Pinedo (PINEDO, 1995) classifica as RPDs em estáticas (que levam em consideração apenas os dados da parte e das máquinas) e dinâmicas (que levam em consideração o tempo em que as partes iniciam seu processamento). Outra classificação usada por Pinedo baseia-se no tipo de informação que a RPD utiliza: regras ditas locais usam informação apenas da máquina para a qual a parte é designada, enquanto que regras globais levam em conta todas as máquinas do sistema.

Kusiak e Ahn (KUSIAK, 1992) apresentam um sistema baseado em regras de liberação para gerenciar o escalonamento. Uma das regras mais importantes é a Regra de Recursos mais Dissimilares (*Most Dissimilar Resources* - MDR), onde as partes que requerem recursos diferentes são liberadas primeiro, podendo assim ocorrer um processamento paralelo dessas.

Su, Chang e Lee (SU; CHANG, 1998) apresentam regras de dominância que surgem da combinação de várias RPDs. Uma heurística utiliza as regras de dominância dividindo o problema em estágios. Holthaus e Rajendran (HOLTHAUS; RAJENDRAN, 1997) propõem 5 regras de despacho considerando objetivos de minimização de tempo médio, máximo e variância do

fluxo das partes, proporção de partes atrasadas, média, máximo e variância do tempo de atraso. Cheng e Jiang (CHENG; J.JIANG, 1998) abordam o não cumprimento das datas de entrega através de uma heurística que combina as técnicas de Associação Dinâmica de Datas e RPDs. Kher (KHER, 2000) apresenta um trabalho onde analisa em conjunto regras de direcionamento de trabalhadores em conjunto com RPDs. Li *et al* (LI et al., 2000) propõem um sistema especialista que combina técnicas de Redes Neurais, conhecimento especialista e RPDs para tratar o problema de re-escalonamento de produção. Shoman *et al* (SHOMAN et al., 2000) faz uma análise de 16 RPDs e coeficientes de performance aplicados a um ambiente de SMF, onde, diferente de um *job shop* convencional, as partes podem ser processadas por qualquer uma das máquinas. Veral (VERAL, 2001) propõe RPDs estáticas baseadas em datas de entrega onde tenta predizer o tempo total de processamento das partes. Essa predição é feita através da análise do tempo de fluxo das partes no sistema em vários cenários. Mohanasudaram *et al* (MOHANASUDARAM et al., 2002) propõem RPDs considerando objetivos de minimizar tempo de fluxo de produção e minimizar atraso. Essas RPDs visam minimizar o tempo de fluxo de produção máximo e o seu desvio-padrão, assim como o tempo máximo de atraso e o seu desvio-padrão. Jayamoham e Rajendran (JAYAMOHAN; REJENDRAN, 2004) desenvolvem RPDs que associam às partes diferentes custos (pesos de penalidades) relativos ao atraso e ao tempo do fluxo de produção. Da mesma forma, outras RPDs considerando diferentes custos para partes adiantadas, atrasadas e para o fluxo de produção são apresentadas por Thiagarajan e Rajendaran (THIAGARAJAN; RAJENDRAN, 2005). Tanev, Uozumi e Morotomi (TANEV; UOZOMI; MOROTOME, 2004) aplicam conceitos de Algoritmos Genéticos e RPDs em um caso real de uma firma com máquinas de injeção de plástico. O objetivo do sistema proposto é escalonar de maneira eficiente os pedidos dos clientes (feitos via *web service*) nas máquinas de injeção de plástico.

5.2.2.3 Têmpera Simulada

A meta-heurística Têmpera Simulada ou *Simulated Annealing* (SA) é baseada no processo físico de têmpera de metais, que consiste em submetê-los à altas temperaturas e reduzi-la gradualmente até atingir o equilíbrio térmico (VIANA, 1998) (JAIN; MEERAN, 1999). No estado de calor inicial, os átomos encontram-se fora de suas posições iniciais e movem-se por entre níveis de energia. A medida que ocorre o resfriamento, os átomos se acomodam em configurações com nível de energia menor que o da configuração inicial.

Cada solução s de otimização combinatória é equivalentes ao estado físico do sistema e a função objetivo $E(s)$ a ser minimizada é o estado interno de energia. O objetivo da busca é levar o sistema de um estado inicial arbitrário até um estado onde a energia seja mínima.

A cada iteração é gerada uma vizinhança $V(s)$ de possíveis soluções s' e analisada a probabilidade de mudança para o estado representado pelos vizinhos. Durante a busca, o algoritmo pode aceitar soluções piores (com níveis de energia superiores ao do estado atual), baseado em uma função P de transição estocástica. P faz a transição de s para um s' baseado em um parâmetro T de temperatura e na diferença entre os níveis de energia desses dois estados (WANG; WU, 1999) (VIANA, 1998).

Geralmente o algoritmo de SA segue os passos descritos na figura 23 (WANG; WU, 1999).

Gerar a configuração inicial s;
Inicializar temperatura $T > 0$;
enquanto critério de parada não for satisfeito **faça**
  para cada $i = 1$ até L **faça**
    escolha um vizinho aleatório s'_i de s;
    faça $\Delta \leftarrow E(s'_i) - E(s)$;
    se $\Delta \leq 0$ **então**
      $s \leftarrow s'$;
    fim se
    se $\Delta > 0$ **então**
      $s \leftarrow s'$ com probabilidade $P = exp(-\Delta/T)$;
    fim se
  fim para cada
  $T \leftarrow rT$;
fim enquanto
Apresente a solução s;

Figura 23: Têmpera Simulada.

Mamalis e Malagardis (MAMALIS; MALAGARDIS, 1996) propõem duas funções de energia alternativas para modelar o atraso máximo: a primeira considera o atraso total de uma determinada seqüência; a segunda estima o tempo de término da parte. He, Yang e Tiger (HE; YANG; TIGER, 1996) apresentam uma heurística que combina SA e o método de geração de vizinhança baseado em Inserção, visando reduzir o atraso. Kolonko (KOLONKO, 1999) demonstra que um algoritmo de SA convencional possui propensão a aderir a ótimos locais quando T é reduzido. Além disso Kolonko apresenta um híbrido de algoritmo genético e SA, onde a população gerada pelo algoritmo genético é executada pelo SA (a qual utiliza controle adaptativo de temperatura). Satake *et al* (SATAKE et al., 1999) utilizam SA combinado com procedimentos de re-escalonamento da produção para minimizar o *makespan*. Wang e Wu (WANG; WU, 1999) apresentam um trabalho onde utilizam a metodologia de superfície de resposta para encontrar um conjunto de parâmetros para o SA, que tornem a busca mais eficiente (temperatura inicial,

razão do decréscimo da temperatura e tamanho da vizinhança gerada). Steinhöfel, Albrecht e Wong (STEINHÖFEL; ALBRECHT; WONG, 1999) propõem um método de geração de vizinhança baseado na modelagem por grafos disjuntivos. Nesse método, os vizinhos são gerados pela troca de arcos de um caminho crítico na seqüência considerada, sendo que o arco é escolhido de acordo com a quantidade de caminhos da seqüência. Três anos depois, os mesmos autores apresentam uma versão paralelizada do modelo de SA (STEINHÖFEL; WONG, 2002). Haaymakers e Hoogeveen (HAAYMAKERS; HOOGEVEEN, 2000) usam SA para tratar o PEJS considerando máquinas em paralelo, restrições de não-espera e sobreposição de operações. Os autores fazem uma adaptação no método de geração de vizinhança, utilizando movimentos de realocação de operações e reinserção de partes. Wang e Zeng (WANG; ZHENG, 2001) combinam técnicas de Algoritmos Genéticos e SA: primeiro o algoritmo genético gera uma solução baseada nos operadores de cruzamento e mutação; depois a solução é otimizada pelo processo de SA. Zhou, Feng e Han (ZHOU; FENG; HAN, 2001) também constroem um híbrido de algoritmos genéticos e SA, considerando regras de despacho no processo evolucionário. Azizi e Zolfaghari (AZIZI; ZOLFAGHARI, 2004) propõem dois modelos de SA para tratar o PEJS minimizando o *makespan*: um algoritmo de SA com controle adaptativo de temperatura (que altera a temperatura baseado na quantidade de movimentos aceitos); o outro modelo implementa uma lista de movimentos proibidos (lista tabu) de forma a evitar que soluções sejam visitadas novamente, sendo que essa última técnica obteve melhores resultados. Xia e Wu (XIA; WU, 2005) tratam o PE em um *job shop* flexível (onde uma operação pode ser processada em qualquer máquina de um determinado conjunto de máquinas), com multiplos objetivos (minimizar o *makespan*, o carregamento total das máquinas e o carregamento da máquina gargalo). Os autores utilizam um híbrido de *particle swarm optimization* e SA para tratar esse problema.

5.2.2.4 Algoritmos Genéticos

A técnica de Algoritmo Genético (AG) foi apresentada por Holland (1975) e baseia-se no processo de evolução que ocorre na biologia. A idéia básica reside na teoria Darwinista de que indivíduos mais adaptáveis ao meio ambiente sobrevivem e se reproduzem.

Analogamente, o AG parte de uma população inicial (configurações iniciais de um problema) gerada aleatoriamente, faz a avaliação de cada indivíduo (através de uma função objetivo f ou função de *fitness*), escolhendo os melhores entre eles,(melhores resultados de f) e realiza manipulações genéticas (através de operadores genéticos) criando uma nova população (VIANA, 1998) (GONÇALVES; RESENDE, 2004). A terminologia utilizada em AGs faz referência à biologia, valendo a seguinte correspondência (GOLDBERG, 1989): *strings*, que fazem a representação

da configuração da solução de um problema, nos AGs, fazem analogia aos cromossomos do sistema biológico. Em sistemas naturais, um ou mais cromossomos combinam-se para formar a descrição genética que permite a construção e operação de algum organismo (pacote genético). Em sistemas naturais, o pacote genético é chamado de genótipo, sendo representado por estruturas de *strings*. Nos sistemas naturais, o organismo formado pela interação do seu pacote genético com o meio ambiente é chamado de fenótipo. Nos AGs a estrutura forma um conjunto de parâmetros e uma solução alternativa (ou ponto no espaço de busca). Na terminologia biológica é dito que os cromossomos são compostos por genes, que podem assumir determinados valores chamados alelos. Nos AGs, diz-se que uma *string* é composta por características, as quais podem assumir diferentes valores. Os operadores genéticos utilizados para gerar novos indivíduos podem ser baseados em qualquer critério de geração de vizinhança. Os operadores tradicionais utilizados em AGs são os seguintes:

- Cópia ou Duplicação: com base nos valores de f os indivíduos são selecionados e seu código genético é duplicado na nova população.
- Cruzamento (*crossover*): operação que possibilita a recombinação do código genético de dois indivíduos selecionados. Esse operador promove uma diversificação no espaço de busca.
- Mutação: consiste em realizar uma pequena perturbação no código genético dos indivíduos selecionados. Geralmente os indivíduos escolhidos para sofrer mutações são indivíduos com padrão genético pobre (valor não-atrativo de f).

A figura 24 descreve o pseudo-código de um AG.

Gerar população inicial P_t
Avaliar População Pt
enquanto critério de parada não é satisfeito **faça**
  Cruzar alguns elementos de P_t e colocar em P_{t+1}
  Copiar alguns elementos de P_t e colocar em P_{t+1}
  Mutar alguns elementos de P_t e colocar em P_{t+1}
  Avaliar a população P_{t+1}
fim enquanto

Figura 24: Algoritmo Genético.

A maioria dos trabalhos que utilizam AGs para tratar o PEJS e suas restrições tem foco na representação do problema. Podem ser vistas diversas técnicas de representar soluções

em forma de cromossomos e várias maneiras de gerar populações através de operadores genéticos. Observa-se que uma das formas de modelagem mais utilizadas é a de grafos disjuntos (BŁAŻEWICZ; DOMSCHKE; PESCH, 1996). Existem várias propostas de heurísticas híbridas que tentam superar algumas limitações de convergência (JAIN; MEERAN, 1999).

Cheng, Gen e Tsujimura (CHENG; GEN; TSUJIMURA, 1996) (CHENG; GEN; TSUJIMURA, 1999) apresentam um tutorial de trabalhos envolvendo AGs no tratamento do PEJS até 1999. No primeiro trabalho os autores focam o conceito básico de AG, as representações mais utilizadas no PEJS (representação baseada em operações, em partes, em listas, em relações, em regras, em grafos, em tempo, baseadas em máquinas e baseadas em chaves aleatórias) e promovem uma discussão sobre complexidade e requisitos de memória. No segundo trabalho os autores focam a aplicação de AGs ao PEJS, dissertando sobre operadores genéticos, modelos híbridos e modelos gerados a partir de AGs. Kumar e Srinivasan (KUMAR; SRINIVASAN, 1996) utilizam AGs combinados com RPDs para tratar o PEJS. Sakawa e Mori (SAKAWA; MORI, 1999) (SAKAWA; KUBOTA, 2000) consideram o PEJS com tempos de processamento e datas de entrega definidas por conjuntos *fuzzy* na modelagem do AG. Na geração de novos indivíduos é proposta uma função de similaridade para o AG, definida a partir dos conjuntos *fuzzy*. Ghrayed (GHRAYEB, 2003) também modela os tempos de processamentos e o *makespan* por conjuntos *fuzzy*, porém utiliza AGs com uma função objetivo bi-critério, onde visa minimizar o *makespan* integral e incerto (relacionado aos conjuntos *fuzzy*).

Ghedjati (GHEDJATI, 1999) trata o PEJS com um híbrido de AG, RPDs e regras heurísticas, onde as duas primeiras técnicas são responsáveis pela seleção das partes a serem processadas e a última é responsável pela escolha das máquinas. Yu e Liang (YU; LIANG, 2001) tratam o PE em um *job shop* expandido utilizando um híbrido de Redes Neurais Restritas (cujos neurônios representam restrições de processamento) e AGs. Yun (YUN, 2002) utiliza Programação por Restrições, Lógica Fuzzy e AGs para tratar o PEJS clássico e uma versão de PEJS, onde o processamento das operações pode ser interrompido para a entrada de outra operação. A razão de mutações e cruzamentos é definida por um controlador *fuzzy*. Varela (VARELA et al., 2003) *et al* tratam o PEJS com gargalos propondo uma técnica heurística para a geração de populações iniciais. Essa heurística associa probabilidades de demanda pelos recursos, gerenciando gargalos. Kim, Park e Ko (KIM; PARK; KO, 2003) tratam o PEJS integrado com Planejamento de Processos, aplicando Algoritmos Coevolucionários (técnica que possui um comportamento semelhante ao AG, onde cada população representa parte do problema modelado; através de um método cooperativo de troca de informações, a solução é encontrada).

Park, Choi e Kim (PARK; CHOI; KIM, 2003) apresentam AGs paralelisados e híbridos de

AGs e Busca Tabu para tratar o PEJS. A solução inicial é gerada pelo híbrido AG-Tabu, que serve de entrada para o AG paralelo. Kis (KIS, 2003) trata o PEJS onde as partes podem possuir várias rotas alternativas. É feita uma representação do problema utilizando grafos do tipo "OU". São propostas duas heurísticas: uma baseada em AGs e outra baseada em Busca Tabu, ambas com métodos de geração de vizinhanças baseados em movimentos de inserção de um conjunto de operações em um escalonamento e de otimização de escalonamento pela troca de operações. Mattfeld e Bierwirth (MATTFELD; BIERWIRTH, 2004) trata a minimização do atraso e do fluxo de produção em um *job shop* utilizando técnicas de redução do espaço de busca aplicadas a AGs. Essas técnicas são baseadas em RPDs e ajudam a decidir conflitos entre operações que requerem os mesmos recursos. Cavory, Dupas e Gonçalves (CAVORY; DUPAS; GONCALVES, 2005) abordam o PEJS cíclico utilizando um modelo composto de um escalonador (que gera seqüências viáveis, resolvendo conflitos de recursos) e um AG que tenta melhorar as seqüências geradas. Watanabe, Ida e Gen (WATANABE; IDA; GEN, 2005) propõem um modelo de AG com adaptação à area de busca. Neste processo a geração de novas populações se dá da seguinte forma: uma fase de seleção baseada em cruzamento e uma fase de seleção baseada em mutação. Em cada uma destas fases é feita uma busca pelo indivíduo com melhor código genético. Gonçalves, Mendes e Resende (GONÇALVES; MENDES; RESENDE, 2005) apresentam uma heurística evolucionária, a qual conta com um módulo de decodificação e seleção de escalonamentos viáveis, um módulo de melhoramento do escalonamento (através de busca local) e um módulo de AG que recebe como entrada o escalonamento melhorado e devolve uma população para o primeiro módulo.

5.2.2.5 Busca Tabu

Busca Tabu (BT) é uma técnica de resolução de problemas de otimização combinatória criada por Glover (GLOVER, 1989). Considera-se uma função f a ser otimizada e uma solução inicial s. É gerada uma vizinhança $V(s)$ de possíveis soluções a partir de s através de um critério de movimento. O melhor vizinho s' gerado é selecionado e caso seja o melhor movimento até então realizado, este é armazenado. O s' selecionado serve de ponto de partida para a geração da vizinhança na iteração seguinte. Para escapar de ótimos locais é implementada uma lista de movimentos proibidos (lista tabu LT). Sempre que um movimento s' é selecionado, verifica-se se este não se encontra em LT; caso se encontre, o movimento é descartado e um outro movimento (pior) em $V(s)$ é escolhido. Dado que o armazenamento apenas dos movimentos pode não ser suficiente para determinar se a busca está realmente voltando a um ponto já visitado, implementa-se um Critério de Aspiração $A(s)$. Esse critério verifica se $f(s')$ proibido é melhor que $f(s*)$; caso seja, o movimento é aceito e o estado tabu esquecido. Existem três critérios de parada tradicionais para a busca: (i) caso seja feito um determinado número *nbmax* de iterações

sem que seja encontrada melhora em $f(s*)$, onde $s*$ representa o melhor resultado encontrado pela busca; (*ii*) caso a busca alcance o resultado ótimo conhecido para o problema abordado; (*iii*) caso a vizinhança a ser percorrida seja inexistente. O pseudo-código da Busca Tabu é descrito na figura 25.

---

contador de iteração $niter \leftarrow 0$;
contador de melhor iteração $melhiter \leftarrow 0$;
s =solução inicial;
lista tabu $LT \leftarrow \varnothing$;
melhor solução $s* \leftarrow s$;
enquanto $(niter - melhiter < nbmax)$ ou $f(s) \leq f(optimum)$ ou $V(s) - LT = \varnothing$ **faça**
  $niter \leftarrow niter + 1$;
  gerar um conjunto $V(s)$ que não seja tabu ou $f(s') < A(f(s*))$;
  escolher a melhor solução s' em $V(s)$;
  atualizar LT e a função de aspiração A;
  se $f(s') < f(s*)$ **então**
    $s* \leftarrow s$
    $melhiter \leftarrow niter$;
  fim se
  $s \leftarrow s'$;
fim enquanto

Figura 25: Busca Tabu

Glover e Laguna detalham outras características possíveis de serem exploradas (GLOVER; LAGUNA, 1997), como as memórias baseadas na freqüência em que um determinado valor é encontrado, lista de soluções de elite (onde as melhores soluções encontradas em uma vizinhança são armazenadas e utilizadas posteriormente, para intensificar a busca em uma determinada região).

Busca Tabu trouxe bons resultados no tratamento do PEJS, sendo que os trabalhos que utilizam essa técnica exploram mecanismos de geração de vizinhança e utilização de memórias de curto e longo prazos (GLOVER; LAGUNA, 1997). A flexibilidade de geração de vizinhança facilita o uso de heurísticas de busca local como critérios de geração de vizinhança.

Nowicki e Smutnick (NOWICKI; SMUTNICKI, 1996) propõem o algoritmo TSAB (*Tabu Search Algorithm with Back Jump Tracking*). A geração de vizinhança de busca é feita através do operador $N5$. O movimento executado, a vizinhança de movimentos e o estado da lista tabu são armazenados em uma lista que funciona como histórico para o TSAB, o que permite que a busca explore melhor o espaço de resultados. Essa técnica teve um bom desempenho frente a algoritmos *Shifting Bottleneck* e Têmpera Simulada e Busca Tabu clássica, para problemas de

pequena, média e larga escalas.

Hertz e Widmer (HERTZ; WIDMER, 1996) tratam o PEJS em um SMF considerando restrição de troca de ferramentas nas máquinas pela mudança de operações. Utilizando a modelagem de grafos disjuntos, os autores propõem uma ferramenta baseada em BT em três fases: (*i*) todas as máquinas são tratadas independentemente e os *setups* são escalonados nelas; (*ii*) encontra-se o número mínimo de trocas de ferramentas para os *setups* escalonados na fase 1; (*iii*) tratando todas as máquinas simultaneamente, encontra-se o caminho mais longo no grafo disjunto considerando os *setups* como operações adicionais.

Gómez (GÓMEZ, 1996) trata o PEJS considerando datas de entrega, processamento em lotes e turnos de produção, visando minimizar o número de instantes de parada, o número de trocas de ferramentas, o tempo ocioso, o atraso e o *makespan*. São consideradas restrições de turnos de produção e fabricação em lotes. A heurística utilizada é a BT com dois critérios de geração de vizinhança: trocas entre lotes de partes (visando redução do tempo de *setup*) e retirada e inserção de uma parte, visando redução do atraso. Os experimentos demonstram um conflito entre os objetivos a serem otimizados (atraso e trocas de ferramentas).

Song e Lee (SONG; LEE, 1996) utilizam o método de geração de vizinhanças baseado em Van Laarhoven para tratar o PEJS onde os tipos de produtos se repetem dentro de um período de tempo. Os autores usam movimento de troca de arcos em um caminho crítico. Agnetis *et al* (AGNETIS et al., 1997) abordam o PEJS em uma CM com duas máquinas, considerando o carregamento das partes e das ferramentas nas máquinas. Para tratar esse problema são utilizadas técnicas de BT (para buscar escalonamentos eficientes) e RPDs (para escalonar as ferramentas nas máquinas).

Valls, Pérez e Quintanilla (VALLS; PEREZ; QUINTANILLA, 1998) consideram um PEJS com produção em lotes e tempos de *setup* associados às máquinas. O critério de geração de vizinhança é composto por três módulos: um módulo para solução viável, um módulo de minimização do atraso e um módulo de diversificação da busca. Verhoeven (VERHOEVEN, 1998) trata o PEJS onde máquinas podem processar várias operações ao mesmo tempo e as partes requerem múltiplas máquinas, possuindo tempos de processamento diferentes em cada uma. Com o objetivo de minimizar o tempo de término de cada parte, o autor utiliza BT com um operador de vizinhança baseado em $N1$. Armentano e Scrich (ARMENTANO; SCRICH, 2000) utilizam RPDs como solução inicial para BT na minimização do *makespan*. Os autores implementam estratégias de intensificação (dentro de um determinado número de iterações a lista tabu é zerada e a busca reinicia do melhor resultado encontrado) e diversificação (proibindo movimentos que levam uma operação ser atribuída a uma mesma máquina um número demasiado

de vezes).

Pezzella e Merelli (PEZZELLA; MERELLI, 2000) utilizam um híbrido de *Shifting Bottleneck* e BT (TSSB), onde a primeira técnica é utilizada para gerar a solução inicial e reotimizar a solução gerada pela BT.

Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 2002) propõem melhorias para o algoritmo TSAB, implementando aceleradores e restringindo a busca no espaço a áreas promissoras. Essa proposta é o algoritmo chamado de *i*-TSAB, ou *iterative-TSAB*. Neste trabalho são apresentadas novas propriedades das vizinhanças geradas pelo operador $N5$. Essas novas propriedades permitem acelerar a busca, reduzindo o tempo computacional de checagem de uma vizinhança significativamente. O *i*-TSAB utiliza iterativamente o TSAB para realizar uma busca local entre duas soluções de uma lista de soluções de elite. As novas soluções encontradas vão substituindo a lista de elite durante a execução do *i*-TSAB. Como as soluções de elite são as melhores encontradas, o algoritmo proposto tenta convergir para a solução ótima rapidamente. Os experimentos realizados comparam *i*-TSAB com TSAB, *Shifting Bottleneck* de Balas e Vazacompoulos (BALAS; VAZACOMPOLOUS, 1998) e TSSB de Pezzela e Merelli (PEZZELLA; MERELLI, 2000).

Zhang *et al* (ZHANG et al., 2006a) propõem um método de movimento onde a primeira e a última partes de um bloco do caminho crítico têm suas posições trocadas com elementos internos do mesmo bloco. Neste trabalho os autores fazem uma comparação entre o método de geração de vizinhança proposto e outros métodos propostos na literatura. Em outro trabalho, Zhang *et al* (ZHANG et al., 2006b) propõem um método híbrido de BT e SA. BT implementa a estratégia de intensificação da busca (através de uma lista de soluções de elite) e SA a diversificação (aceitando soluções (atrativas ou não) de acordo com a função estocástica de aceitação P).

Watson, Howe e Whitley (WATSON; HOWE; WHITLEY, 2006) realizam experimentos para verificar os fatores que mais influenciam o desempenho do algoritmo *i*-TSAB de Nowicki e Smutinicki. Neste trabalho os autores demonstram que, ampliando o operador de vizinhança $N5$ nas técnicas de Monte Carlo, Busca Local Iterativa e Busca Tabu simples, obtém-se resultados competitivos com algoritmo *i*-TSAB. Os experimentos levam os autores a inferir que o operador $N5$ aliado a estruturas de memória de longo prazo tem um impacto significantemente positivo em qualquer heurística em que são aplicados.

passo 0. $M_0 = \varnothing$;
passo 1. Identificar a máquina-gargalo m entre as M/M_0 máquinas e calcular a seqüência ótima, dado o escalonamento parcial das máquinas em $M0$. Fazer $M_0 \leftarrow M_0 + m$.
passo 2. Re-otimizar sucessivamente a seqüência de cada máquina $k \in M_0$, dado o escalonamento parcial das máquinas em $M_0/\{k\}$.
se $M_0 = M$ **então**
  parar.
senão
  ir para o passo 1.
fim se

Figura 26: *Shifting Bottleneck*

5.2.2.6 *Shifting Bottleneck*

Shifting Bottleneck (SB) é uma heurística proposta por Addams em 1988 (JAIN; MEERAN, 1999) e foi a primeira a resolver a instância FT10 do PEJS. Essa técnica consiste em relaxar o PEJS em m PEs de uma só máquina e resolvê-los um a cada vez. As soluções de cada máquina são comparadas e a máquina que tiver o maior limite superior é identificada como máquina-gargalo. SB seqüencia a máquina-gargalo primeiro, deixando as máquinas já seqüenciadas fixas e ignorando as não seqüenciadas. Cada vez que uma máquina-gargalo é seqüenciada, as demais máquinas já visitadas ficam suscetíveis a uma reotimização. Sendo M o conjunto de máquinas e M_0 o conjunto de máquinas escalonadas pelo SB, a figura 26 mostra em passos o procedimento do SB (IVES; LAMBRECHT, 1996).

Jain e Meeran citam diversos trabalhos que utilizam SB para tratar o PEJS até 1999, com destaque para o trabalho de Balas e Vazacopoulos (JAIN; MEERAN, 1999). Singer (SINGER, 2001) apresenta uma heurística de *rolling horizon* que divide o escalonamento em janelas de tempo e otimiza cada janela independentemente. O procedimento de SB controla a heurística proposta. Wenqui e Aihua (WENQI; AIHUA, 2004) propõem um procedimento de SB melhorado (*Improved SB*), o qual resolve o PEJS para uma máquina considerando regras de prioridade entre operações. Mönch e Brießel (MÖNCH; BRIESSEL, 2005) consideram o PEJS com máquinas paralelas e partes re-entrantes.

5.3 Problema Proposto

No estudo do estado da arte realizado neste trabalho, puderam ser observados PEJS com várias particularidades. No entanto, não foi observado um trabalho que abordasse simultaneamente restrições de turnos de produção, ferramental necessário para o processamento das operações e datas de entrega dos produtos. Neste trabalho propõe-se tratar o PEJS considerando restrições de tempo de *setup*, datas de entrega e turnos de produção. Considera-se uma função objetivo cujo valor representa a soma ponderada entre os seguintes objetivos de minimização: tempo total de produção (referido aqui como *Makespan*), tempo dos instantes de parada para troca de ferramentas (Tempo de *Setup*) e tempo de atraso na entrega das partes. Neste trabalho o problema proposto será referido pela sigla PEM. A formulação do problema, restrições e hipóteses são apresentadas a seguir.

5.3.1 Formulação e hipóteses do PEM

Considera-se um conjunto de partes $J = \{1,\ldots,n\}$ a serem produzidas, um conjunto $M = \{1,\ldots,m\}$ de máquinas que processam essas partes, um conjunto $O = \{1,\ldots,o\}$ de operações requeridas pelas partes e um conjunto $T = \{1,\ldots,t\}$ de ferramentas necessárias. O conjunto O pode ser decomposto em subconjuntos $O_j,\ j \in J$. A parte j consiste de uma seqüência O_j de operações que devem ser processadas em uma determinada ordem, sendo que a última operação deve ser realizada antes da data de entrega d_j. Uma operação i pode ser definida pela máquina $\mu_i \in M$ na qual deve ser processada, pelo seu tempo $p_i \geq 0$ e pelas ferramentas T_{O_j} requeridas. O conjunto de operações pode ser decomposto em subconjuntos $M_k = \{i \in O : \mu_i = k\}$, cada um correspondente às operações que devem ser processadas na máquina k, sendo $m_k = |M_k|, k \in M$. Um turno de produção é um período de tempo η em que se admite processo produtivo. As principais hipóteses são as seguintes:

- uma máquina pode processar apenas uma parte por vez;
- cada máquina possui um magazine com capacidade limitada ζ;
- cada operação precisa de um número de ferramentas que nunca excede a capacidade ζ do magazine da máquina na qual será processada;
- não existem operações que possam ser interrompidas;
- duas operações sucessivas de uma mesma parte são processadas em máquinas diferentes;
- nenhuma parte pode ser processada depois do final do turno de produção;

- nenhuma operação pode ficar com seu processamento inacabado ao final de um turno.

Uma vez que o magazine das máquinas tem capacidade limitada, paradas para trocas de ferramentas serão necessárias. O momento em que uma máquina pára seu processamento para troca de ferramentas é chamado de *setup*. Cada *setup* dura $\alpha + \beta t$ unidades de tempo, onde α é o tempo fixo referente à remoção das ferramentas e limpeza da área de trabalho, β é o tempo fixo gasto na colocação de uma ferramenta no magazine e t é o número de ferramentas trocadas em um *setup* (HERTZ; WIDMER, 1996).

Um escalonamento viável é definido por uma seqüência de tempos de início de operações $S_i \geq 0, i \in O$, tal que as restrições acima sejam satisfeitas. O problema proposto é encontrar um escalonamento viável que minimize os seguintes tempos:

- *Makespan*: o tempo máximo necessário para o processamento de todas as operações, considerando apenas as horas trabalhadas dentro dos turnos de produção.

$$Makespan = \max_{i \in O}(S_i + p_i). \tag{5.1}$$

- Atraso: diferença positiva entre a data de término e a data de entrega prevista para cada uma das partes a serem produzidas. Diferenças negativas são consideradas 0.

$$Atraso = \sum_{j=1}^{J} (\max_{i \in O_j}(S_i + p_i) - d_j), \ \forall \max_{i \in O_j}(S_i + p_i) - d_j \geq 0. \tag{5.2}$$

- Tempo de *Setup*: o tempo total gasto em paradas para a troca de ferramentas durante a produção das partes.

$$Setup = \sum_{k=1}^{M} \sum_{o=1}^{M_k} x_{ko} \cdot (\alpha + \beta t), \tag{5.3}$$

onde x_{ko} é 1 se ocorre *setup* entre operações o na máquina k, 0 caso contrário.

Cada um destes tempos pode ser considerado como uma parcela independente a ser minimizada. A função objetivo f_π proposta considera a minimização desses tempos simultaneamente, sendo cada um deles uma variável de decisão da função. De forma a realizar o gerenciamento da importância de cada variável de decisão, associam-se pesos W_i a elas.

$$\text{minimizar } f_\pi = W_1 \cdot Makespan + W_2 \cdot Atraso + W_3 \cdot Setup, \tag{5.4}$$

sendo

$Makespan$, $Atraso$, $Setup \geq 0$,

$W_i \geq 0,\ i = 1,\ldots,3$

A revisão bibliográfica realizada destaca o bom desempenho dos trabalhos de Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 1996) (NOWICKI; SMUTNICKI, 2002) no tratamento do PEJS com o objetivo de minimizar o *makespan*. Os autores propuseram um modelo de grafos dirigidos para representar o problema e apresentaram uma técnica baseada em Busca Tabu, que obteve bons resultados.

No tratamento do PEJS com restrições de ferramentas, dois trabalhos têm destaque: (*i*) Hertz e Widmer (HERTZ; WIDMER, 1996), que fazem a modelagem dos instantes de parada através da formulação de grafos disjuntos de Roy e Sussmann (BŁAŻEWICZ; DOMSCHKE; PESCH, 1996), aplicando uma técnica baseada em Busca Tabu e (*ii*) Gómez (GÓMEZ, 1996) aborda uma versão do PEJS considerando uma função multi-objetivo, onde procura minimizar os tempos de troca de ferramentas, *makespan* e o tempo ocioso. O problema é aplicado a um SMF composto de uma máquina versátil, utilizando um modelo baseado em Busca Tabu.

Os trabalhos citados acima contribuiram para a modelagem do PEM, na qual se utiliza a formulação de grafos proposta por Nowicki e Smutnicki, adaptando as trocas de ferramentas, como no trabalho de Hertz e Widmer. O tratamento de vários objetivos ao mesmo tempo é feito com base nos trabalhos de Gómez. Nos itens a seguir são apresentados os detalhes da modelagem.

5.3.2 Modelagem do Problema

Para modelar o PEM escolheu-se utilizar o modelo de grafos dirigidos de Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 1996), apresentada na seção 5.2. Sendo $\pi = (\pi_1,\ldots,\pi_m)$ uma permutação das $i \in O$ operações nas $m \in M$ máquinas, $G(\pi) = (O, R \cup E(\pi))$ o grafo dirigido que representa essa permutação, $r_\pi(i)$ o caminho mais longo em $G(\pi)$ até o vértice i incluindo p_i, então as equações (5.1) e (5.2) podem ser reescritas da seguinte forma:

- *Makespan*:

$$Makespan = \max_{i \in O}(r_\pi(i)). \tag{5.5}$$

- Atraso:

$$Atraso = \sum_{j=1}^{J} (\max_{i \in O_j}(r_\pi(i)) - d_j),\ \forall \max_{i \in O_j}(r_\pi(i)) - d_j \geq 0. \tag{5.6}$$

De forma a adaptar a modelagem proposta por Nowicki e Smutnicki para o PEM, as restrições de tempo de *setup*, turnos de produção e datas de entrega são adicionadas. Para o cálculo de f_π, que é resultado da ponderação das 3 variáveis de decisão, a restrição de trocas de ferramentas é adicionada, e as operações de *setup* adicionadas. A seguir é feita a divisão da produção em turnos de produção e o *makespan* é calculado. Com as datas de término de cada parte, pode-se calcular o atraso.

5.3.2.1 Inclusão do Tempo de Setup

Em seu trabalho, Hertz e Widmer (HERTZ; WIDMER, 1996) demonstram que uma solução ótima para o PEJS pode não ser ótima para o PEJS com restrições de *setup*, dado que o tempo de parada para a troca de ferramentas e o compartilhamento dessas entre as operações deve ser considerado. Em sua modelagem, os autores supõem que, dado uma ordem de processamento e o conteúdo do magazine das máquinas, o escalonamento com o mínimo *makespan* pode ser encontrado pela identificação do caminho crítico de custo mínimo. Neste caso os *setups* são considerados operações adicionais que tem uma duração conhecida, sendo possível construir o grafo $G(\pi)$.

Hertz e Widmer sugerem duas etapas anteriores ao cálculo do caminho crítico. Estas etapas consistem em: (*i*) identificar em cada máquina a ocorrência de *setups* e o conjunto de ferramentas requeridas pelas operações de cada máquina e (*ii*) determinar o conteúdo do magazine em cada *setup*.

No tratamento do PEM, é proposto o agrupamento das operações em Famílias de Operações (FO), da mesma forma que é realizado o agrupamento de partes em FPs, como apresentado no capítulo 3. As operações são agrupadas em $FO = (FO(1), \ldots FO(Q))$, sendo Q o número de FOs. Considera-se uma matriz bidimensional H de dimensões $Q \times Q$, onde cada elemento a_{ij} contém o número de ferramentas trocadas entre as FOs i e j, sendo que $a_{ij} = 0$ para $i = j$.

Estando as operações agrupadas em Famílias de Operações, pode-se realizar uma etapa de definição da ocorrência de *setups*, do conjunto de ferramentas envolvido em cada um e no conteúdo do magazine durante o período de atividade das máquinas. Considerando FOs, a

equação 5.3 pode ser reescrita da seguinte forma:

$$Setup = \sum_{i=1}^{O-1} (\alpha + \beta \cdot H_{FO(i),FO(i+1)}). \tag{5.7}$$

O exemplo apresentado na tabela 3 é modificado para incluir o tempo de *setup*. Considera-se o conjunto de ferramentas requeridas pelas 12 operações como $T = (x,x,y,x,y,z,z,z,y,z,x,y)$, o *setup* com $\alpha = 0$ e $\beta = 1$, a capacidade do magazine $\zeta = 2$. São geradas duas famílias de operações: $FO = (xy,xz)$, sendo que cada uma das operações é atribuída a uma FO. A matriz que identifica a troca de ferramentas entre FOs é $H = (1,1)$. O gráfico de Gantt considerando as trocas de ferramentas é apresentado na figura 27.

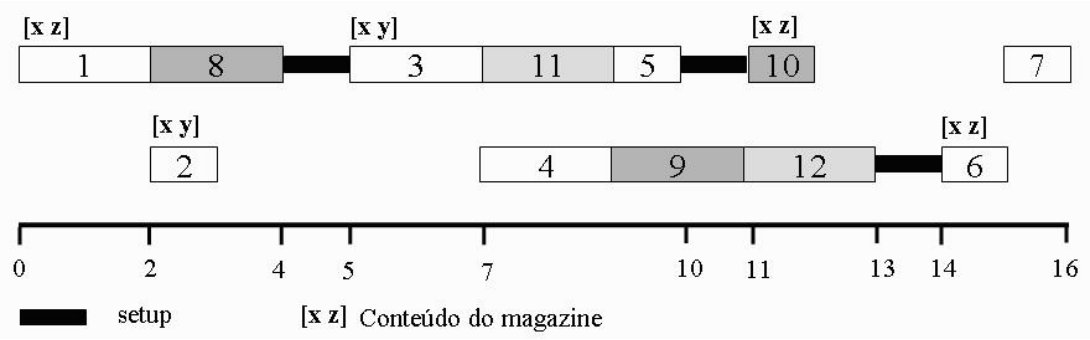


Figura 27: Gráfico de Gantt para uma instância do PEJS considerando tempo de *setup* entre operações, com $\alpha = 0$ e $\beta = 1$.

Na figura 27 pode-se notar a ocorrência de 3 *setups*. Usando a equação 5.7, calcula-se o valor do tempo de troca de ferramentas:

$FO(8) \neq FO(3)$, $FO(5) \neq FO(10)$, $FO(12) \neq FO(6)$,

$1 + 1 + 1 = 3$.

Com a identificação dos *setups*, podem ser inseridos no grafo dirigido $G(\pi)$ operações que os representam. A figura 28 representa o problema apresentado na tabela 3 com a adição das operações representando os *setups*.

5.3.2.2 Inclusão dos turnos de produção e cálculo do *Makespan*

Com a adição do tempo de cada *setup*, a próxima etapa é o cálculo do *makespan* levando em consideração os turnos de produção. Para melhor ilustrar a modelagem dos turnos de produção, foi feita uma modificação na figura 27 para que fossem considerados turnos de $\eta = 8$ unidades de tempo.

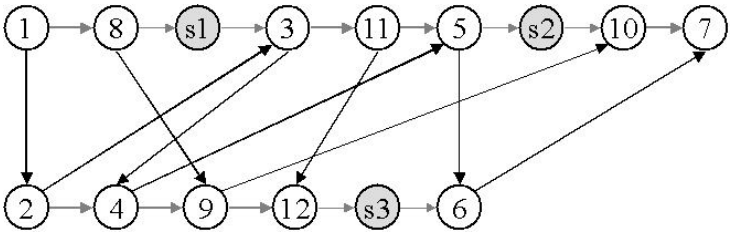


Figura 28: Grafo dirigido para uma instância do PEJS considerando tempo de *setup* entre operações. As operações s1, s2 e s3 são referentes aos *setups*.

A inclusão dessa restrição impede que uma operação encerre um turno com seu processamento inacabado. Assim, podem existir períodos de ociosidade das máquinas ao final de cada turno, que devem ser levados em consideração para o cálculo do *makespan*. Na figura 29 pode-se observar o tempo ocioso ao final de um turno de produção. O tempo de produção das partes é crescente e os turnos são considerados apenas uma divisão desse tempo, não sendo considerados o tempo entre-turnos.

5.3.2.3 Inclusão de Datas de Entrega

A última etapa para o cálculo do valor de f_π é o cálculo do valor do atraso da produção. As datas de entrega são consideradas um atributo de cada parte na modelagem do PEM. Logo, a formulação de grafos dirigidos de Nowicki e Smutnicki é modificada para considerar esse atributo pela inclusão de mais um peso aos vértices de $G(\pi)$. Esse peso representa o atraso da operação i em relação à data d_j de entrega da parte. Continuando o exemplo anterior, considera-se o gráfico de Gantt da figura 30. As datas de entrega das partes são $d_1 = 14$, $d_2 = 10$ e $d_3 = 12$.

Utilizando a equação (5.6), pode-se calcular o tempo de atraso do exemplo anterior.

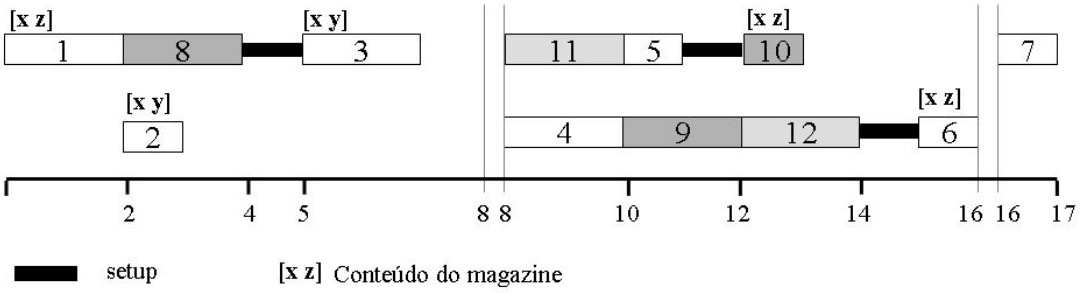


Figura 29: Gráfico de Gantt para uma instância do PEM com 3 partes, 12 operações, 3 tipos de ferramentas, $\zeta = 2$, $\eta = 8$. Nota-se que o tempo de produção é crescente e os turnos apenas dividem esse tempo.

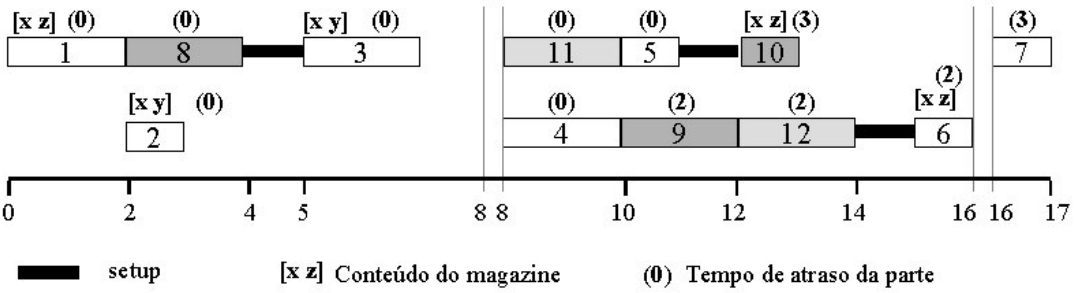


Figura 30: Gráfico de Gantt para uma instância do PEM com $J = 3$, $M = 2$, $O = 12$, $T = 2$, $\zeta = 2$, $\eta = 8$, $D = (14, 10, 12)$.

Assim, para as partes 1, 2 e 3 e suas últimas operações 7, 10 e 12:

$$r_\pi(7) = 17,\ r_\pi(10) = 13,\ r_\pi(12) = 14,$$
$$d_1 = 14,\ d_2 = 10,\ d_3 = 12,$$
$$(r_\pi(7) - d_1) + (r_\pi(10) - d_2) + (r_\pi(12) - d_3),$$
$$(17 - 14) + (13 - 10) + (14 - 12) = 8.$$

Com a inclusão das datas de entrega é possível fazer o cálculo do valor de f_π. Consideram-se os pesos $W_1 = 1$, $W_2 = 1$ e $W_3 = 1$ para *makespan*, atraso e trocas de ferramentas, respectivamente. Para o escalonamento visualizado na figura 30, o valor de f_π é o seguinte:

$$f_\pi = W_1 \cdot Makespan + W_2 \cdot Atraso + W_3 \cdot Setup$$
$$f_\pi = 1 \cdot 17 + 1 \cdot 8 + 1 \cdot 3$$
$$f_\pi = 28$$

Uma vez definido o método de cálculo de cada uma das parcelas da função objetivo, é feita a definição do método de geração de vizinhança, estruturas de memória e técnica de busca no espaço a serem utilizadas para a minimização de f_π. No capítulo 6 serão descritas essas definições.

6 TÉCNICAS UTILIZADAS

O Problema de Escalonamento em *Job Shop*, clássico na área de otimização combinatória, é de difícil solução, possuindo complexidade *NP*-Difícil (VIANA, 1998). A revisão do estado da arte mostrou que os métodos exatos (como *branch-and-bound*) não são eficientes no tratamento desse tipo de problema. Resultados considerados mais promissores (BŁAŻEWICZ; DOMSCHKE; PESCH, 1996) começaram a ser atingidos com o uso de heurísticas e, posteriormente, meta-heurísticas, como Têmpera Simulada, Algoritmos Genéticos e Busca Tabu. Os trabalhos que utilizaram BT e híbridos de BT alcançaram os resultados mais promissores, a exemplo, os já citados trabalhos de Hertz e Widmer (HERTZ; WIDMER, 1996), Pezzella e Merelli (PEZZELLA; MERELLI, 2000) e Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 2002).

Em seu trabalho, Watson, Howe e Whitley (WATSON; HOWE; WHITLEY, 2006) investigam se o bom desempenho do método de Nowicki e Smutnicki (*i*-TSAB) dá-se pelo fato de ser baseado em BT ou por empregar um operador de vizinhança $N5$ (citado no capítulo anterior), mecanismos de detecção de ciclos a longo prazo e estratégias de intensificação e diversificação. Os autores demonstram que, utilizando essas estratégias auxiliares, outras meta-heurísticas (como Monte Carlo) apresentam resultados competitivos com *i*-TSAB.

O trabalho acima citado motiva o uso dessas estratégias auxiliares junto a meta-heurística escolhida para minimizar o valor da função objetivo f_π descrita no capítulo 5. Na seção a seguir são descritas as estratégias auxiliares de geração de vizinhança com o operador $N5$, detecção de ciclos a longo prazo e estratégias de intensificação e diversificação. A meta-heurística inicialmente escolhida para tratar o PEM é a Busca Tabu, que é detalhada na seção seguinte.

6.1 Estratégias auxiliares

As estratégias auxiliares consideradas neste capítulo são técnicas que, inseridas em uma determinada meta-heurística, melhoram seu desempenho. Para o PEJS, Watson, Howe

e Whitley consideram em seu trabalho estratégias de geração de vizinhança, intensificação, diversificação e detecção de ciclos usadas nos trabalhos de Nowicki e Smutnicki.

A estratégia de geração de vizinhança utilizada com sucesso nos trabalhos de Nowicki e Smutnicki e Watson, Howe e Whitley é baseada no operador de geração de vizinhança $N5$, sigla escolhida por Błażewicks, Domschke e Pesch (BŁAŻEWICZ; DOMSCHKE; PESCH, 1996). Esse operador foi proposto por Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 1996), sendo baseada no operador $N1$ de Van Laarhoven *et al* (LAARHOVEN; AARTS; LENSTRA, 1992) e Taillard (TAILLARD, 1994) e $N4$. No operador $N1$, um movimento $v = (x,y)$ é realizado sendo x e y operações adjacentes em uma mesma máquina ou em um caminho crítico, respectivamente. No operador $N4$, o movimento v é realizado somente nas primeiras e nas últimas partes que compõem um bloco de um caminho crítico em π. Assim, a vizinhança é definida como as ordens de processamento obtidas da ordem π através do movimento v. Para esses operadores, o tamanho da vizinhança depende do número de caminhos críticos encontrados em π. O operador $N5$ considera apenas um único caminho crítico u^{π} (arbitrariamente escolhido) que possui os blocos $B_1,\ldots,B_s$. As duas primeiras (e simetricamente as duas últimas) operações dos blocos $B_2,\ldots,B_{s-1}$ são trocadas de posição, sendo que cada bloco tem pelo menos duas operações. No primeiro bloco B_1 são trocadas somente as duas últimas operações, e por simetria, no último bloco B_s são trocadas as duas primeiras.

Formalmente, Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 1996) definem o operador $N5$ da forma apresentada a seguir. Considera-se: para o escalonamento π um caminho crítico u que contém r operações; o conjunto de movimentos de π como sendo $V(\pi) = \bigcup_{j=1}^{s} V_j(\pi)$. Os movimentos realizados para gerar a vizinhança são os seguintes:

$$V_1(\pi) = \begin{cases} \{(u_{b_1-1}, u_{b_1})\} & \text{se } a_1 < b_1 \text{ e } s > 1, \\ \varnothing & \text{caso contrário.} \end{cases}$$

$$V_j(\pi) = \begin{cases} \{(u_{a_j}, u_{a_j+1}), (u_{b_j-1}, u_{b_j})\} & \text{se } a_j < b_j,\ j = 2,\ldots,r-1 \\ \varnothing & \text{caso contrário.} \end{cases}$$

$$V_s(\pi) = \begin{cases} \{(u_{a_s}, u_{a_s+1})\} & \text{se } a_s < b_s < 1, \\ \varnothing & \text{caso contrário.} \end{cases}$$

Para o exemplo visto no capítulo anterior, mostrado na figura 31, onde existe um caminho crítico u formado por 3 blocos $B_1 = (1,8,3)$, $B_2 = (4,9,12,6)$ e $B_3 = (7)$. Considera-se π_v : $v \in \Pi$ as ordens de processamento conseguidas a partir de π aplicando-se o movimento v. A vizinhança $N(\pi)$ é composta por todas as ordens de processamento conseguidas pela aplicação

de movimentos de $V(\pi)$, sendo $N(\pi) = \{\pi_v \ : \ v \in V(\pi)\}$. Segundo a definição do operador $N5$, podem ser realizados um movimento no bloco B_1, dois movimentos no bloco B_2 e nenhum movimento no bloco B_3. Assim, $V_1(\pi) = \{(8,3)\}$, $V_2(\pi) = \{(4,9),(12,6)\}$ e $V_3(\pi) = \varnothing$. Neste caso a vizinhança gerada possui 3 vizinhos, que são os seguintes:

$$\pi = ((1,8,3,11,5,10,7),(2,4,9,12,6))$$
$$N(\pi) = \{Q(\pi,(8,3)),Q(\pi,(4,9)),Q(\pi,(12,6))\}$$
onde
$$\pi_{(8,3)} = ((1,3,8,11,5,10,7),(2,4,9,12,6))$$
$$\pi_{(4,9)} = ((1,8,3,11,5,10,7),(2,9,4,12,6))$$
$$\pi_{(12,6)} = ((1,8,3,11,5,10,7),(2,4,9,6,12))$$

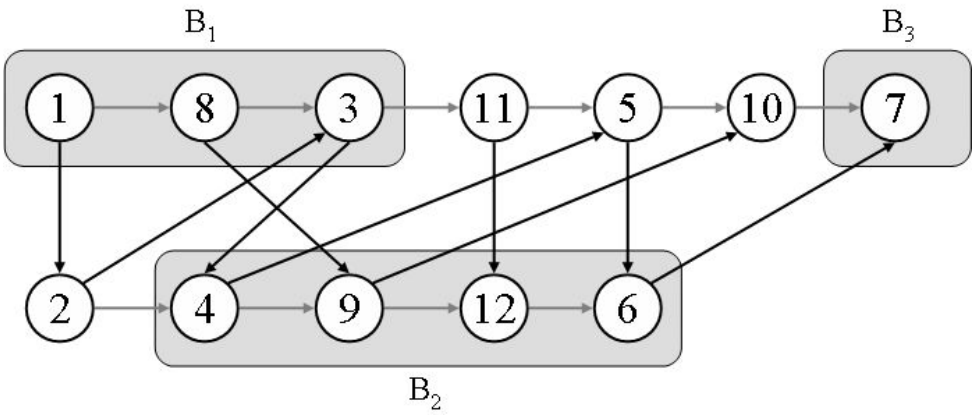


Figura 31: Instância do PEJS considerando o caminho crítico u^{π} e os blocos B_1, B_2, B_3.

Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 1996), provam que uma vizinhança obtida através do operador $N5$ possui duas propriedades: todas as ordens de produção são viáveis e promissoras (com *makespan* menor), dado que os movimentos gerados por $N5$ são um sub-conjunto dos movimentos gerados por $N1$; caso $V(\pi) = \varnothing$, a ordem de processamento π é ótima. Outras duas propriedades de $N(\pi)$ são apresentadas em um trabalho posterior (NOWICKI; SMUTNICKI, 2002), onde os autores provam que é possível reduzir o tempo de cálculo do *makespan* dos vizinhos em $N(\pi)$. A vizinhança gerada pelo operador $N5$ tem um número consideravelmente menor de vizinhos que a vizinhança de $N1$. O fato do operador $N5$ ignorar movimentos que não oferecem melhora na função objetivo leva a geração de um espaço de busca desconexo, onde nem sempre existe um caminho entre uma solução arbitrária e o resultado ótimo. Assim, uma meta-heurística que está baseada somente no operador $N5$ não pode garantir que uma solução ótima venha a ser localizada (WATSON; HOWE; WHITLEY, 2006) (NOWICKI; SMUTNICKI, 2002).

Watson, Howe e Whitley demonstram que algumas soluções geradas pelo operador $N5$

podem fazer com que a busca faça ciclos em torno de um grupo de resultados. Neste caso, mecanismos de detecção de ciclos devem ser utilizados, possibilitando que a busca consiga sair do ciclo detectado. A detecção de ciclo utilizada no algoritmo TSAB é feita pela checagem explícita dos valores de *makespan* a cada iteração. Se o valor se repete dentro de um número $max\delta$ de iterações, TSAB reinicia a busca da melhor solução até então encontrada. Watson, Howe e Whitley propõem um *framework* para detecção de ciclo chamado CMF (*Core Metaheuristic Framework*). Esse framework executa $MaxIters$ iterações de uma meta-heurística T e retorna o melhor resultado encontrado. É utilizado o conceito de mobilidade (distância de uma solução a outra em número de iterações); caso a mobilidade da busca seja inferior a um limite M_{thrs}, a busca executa MWL_{nom} iterações utilizando o operador $N1$.

Além do operador $N5$ e da detecção de ciclos, estratégias de intensificação e diversificação melhoram o desempenho de uma meta-heurística. Intensificação é a estratégia de explorar soluções próximas a uma determinada solução promissora. Na diversificação são geradas soluções com atributos diferentes de forma a tentar encontrar regiões no espaço de busca ainda não exploradas. Os mecanismos de intensificação e diversificação utilizados por Watson, Howe e Whitley em seu *framework* IDMF (*Intensification - Diversification Metaheuristic Framework*) são semelhantes aos empregados por Nowicki e Smutnicki no algoritmo *i*-TSAB. Para implementar essas estratégias, considera-se uma lista de soluções de elite LE, que armazena as melhores soluções encontradas em um determinado número de iterações de uma meta-heurística.

O procedimento descrito pelos autores consiste basicamente de duas fases: inicial e iterativa. Na fase inicial, a partir de uma solução aleatória (ou gerada por uma heurística), os autores utilizam uma meta-heurística para encontrar as soluções de elite que vão compor LE.

No algoritmo *i*-TSAB é implementada uma função $\phi = NIS(\alpha, \beta, Makespan^r)$, onde α e β são soluções de elite, $Makespan^r = \max\{Makespan(\alpha), Makespan(\beta)\}$ é o *makespan* de referência e ϕ é uma solução eqüidistante a α e β, obtida via *path relinking*. Na fase inicial, a lista LE contém apenas a solução π^0 gerada pela heurística externa. Seja π^{i-1} , $\forall\, 2 \leq i <$ $maxE$ a última solução introduzida na lista LE de tamanho $maxE$; π^i deriva da exploração local de ϕ através do algoritmo TSAB, sendo $\phi = NIS(\pi^{i-1}, \pi^0, Makespan^*)$, sendo $Makespan^*$ atualizado a cada iteração da fase inicial.

O *framework* IDMF conta na fase inicial com uma função CMF', similar ao CMF, para construir a lista LE. Diferente da fase inicial de *i*-TSAB, IDMF gera os elementos de LE independentemente, aplicando CMF' em soluções aleatoriamente geradas por um método externo.

Uma vez construída a lista LE, a segunda fase consiste na aplicação de estratégias de

intensificação e diversificação nos elementos de *LE* utilizando uma meta-heurística.

O algoritmo *i*-TSAB vai atualizando *LE* da seguinte maneira: seja π^k a solução em *LE* com menor *makespan*; identifica-se a solução π^l mais distante de π^k em termos de distância em grafos disjuntos; encontra-se a solução $\phi = NIS(\pi^k, \pi^l, Makespan^*)$; inicia-se uma exploração local utilizando TSAB em ϕ, a qual provê uma nova solução elite que substitui π^l. O processo é repetido até que a distância entre π^k e π^l não ultrapasse um limite *maxD* especificado.

Na segunda fase, o *framework* IDMF alterna séries de intensificação e diversificação, que são aplicadas aos elementos de *LE* de acordo com probabilidades $prob_i$ e $prob_d$ a cada iteração. A intensificação consiste em aplicar o procedimento CMF' em uma solução x aleatoriamente escolhida em *LE*. Se $Makespan(s^*) < Makespan(x)$ sendo s^* a solução retornada por CMF', s^* substitui x. Na diversificação, duas soluções $x, y \in LE$ são escolhidas aleatoriamente; a função *NIS* do algoritmo *i*-TSAB é então utilizada para gerar uma solução ϕ entre x e y; é aplicado o procedimento CMF' em ϕ; se $Makespan(s^*) < Makespan(x)$, então s^* substitui x em *LE*. IDMF termina assim que CMF' e *NIS* excedam um determinado número $MaxIters$ de iterações.

Os trabalhos citados nesta seção apresentam bons resultados frente aos pesquisados no estado da arte. Watson, Howe e Whitley (WATSON; HOWE; WHITLEY, 2006) demonstram que o uso de estratégias de intensificação e diversificação, apresentadas sob a forma do *framework* IDMF, melhoram o desempenho de uma meta-heurística, tornando-a competitiva frente ao trabalho de Nowicki e Smutnicki. Baseado nesses trabalhos é proposta a técnica para o tratamento do PEM, apresentada a seguir.

6.2 Técnica para abordar o Problema

Com base no levantamento do estado da arte, com ênfase nos trabalhos de Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 1996) (NOWICKI; SMUTNICKI, 2002), Watson, Howe e Whitley (WATSON; HOWE; WHITLEY, 2006) e Hertz e Widmer (HERTZ; WIDMER, 1996), propõe-se uma técnica para o tratamento do PEM, a qual conta com os operadores $N5$ e $N1$ para a geração de vizinhança, lista de soluções de elite, procedimento baseado em *path relinking* que permite uma estratégia de intensificação e diversificação da busca. A técnica proposta é derivada de adaptações no algoritmo *i*-TSAB, sendo chamada de ei-TSAB (*expanded i-TSAB*). As adaptações realizadas são referentes aos turnos de produção, datas de entrega e *setups*, como descrito no capítulo 5.

Dada a restrição de ferramentas, necessita-se fazer o agrupamento das operações em

Família de Operações. Para tanto, utiliza-se o Algoritmo de Identificação de Grupos Modificado (AIGM), proposto por Gómez (GÓMEZ, 1996). Esse algoritmo é semelhante ao de Kusiak e Chow, descrito no capítulo 3, leva em consideração o compartilhamento de ferramentas entre operações. O AIGM possui os seguintes passos:

Passo 1. Atualizar a iteração $k = 0$ e limite de ferramentas.

Passo 2. Selecionar uma linha i da matriz incidente A^k que possua a menor quantidade de $a_{ij} = 1$ e traçar uma linha horizontal h_i sobre ela.

Passo 3. Para cada j tal que $a_{ij} = 1$ cruzado pela linha horizontal h_i, traçar uma linha vertical v_j.

Passo 4. Verificar se o número de linhas verticais traçadas supera o limite estipulado do número de ferramentas. Se superar, selecionar de maneira crescente em relação ao número de linhas verticais v_j traçadas tal que o limite do número de ferramentas seja respeitado (número de linhas verticais traçadas). Ir para o passo5.

Passo 5. Para cada i tal que $a_{ij} = 1$, cruzado pela linha vertical v_j, trace uma linha horizontal h_i. Volte ao passo 2.

Passo 6. Transformar a matriz incidente A^k em A^{k+1} removendo as linhas e colunas traçadas nos passos 2 a 5. As colunas da matriz incidente A^k onde $a_{ij} = 1$ devem continuar aparecendo na matriz A^{k+1}.

Passo 7. Se a matriz $A^{k+1} = \varnothing$, parar; caso contrário, fazer $k = k + 1$ e voltar ao passo 2.

A geração das FOs se dá antes da geração da solução inicial para o ei-TSAB. Propõe-se utilizar inicialmente soluções inicias geradas aleatoriamente, segundo a abordagem de Watson, Howe e Whitley (WATSON; HOWE; WHITLEY, 2006). Estudos já realizados com soluções iniciais baseadas em Regras de Despacho simples podem ser consideradas (RODRIGUES; GóMEZ, 2005). Essas Regras de Despacho consistem de procedimentos simples, como: ordenação das partes segundo o tempo total de processamento, ordenação segundo datas de entrega e ordenação segundo tipos de FOs. Nowicki e Smutnicki utilizam uma técnica de solução inicial baseada em inserção.

O algoritmo ei-TSAB opera em duas fases: inicial e iterativa, como descrito na seção anterior. Considera-se a modelagem do PEM apresentada no capítulo 5. Define-se uma função *mSwitch* que identifica a ocorrência dos *setups* e calcula o tempo de cada um, conforme as ferramentas trocadas, inserindo-os como vértices em $G(\pi)$. Define-se uma função *mPath* que

retorna, para uma determinada ordem de processamento π, o valor da função objetivo f_π e o caminho crítico identificado.

Define-se uma função $\phi = NIS(\alpha, \beta, f_\pi^r)$ que retorna uma solução ϕ que se encontra entre as soluções α e β, considerando o valor f_π de referência f_π^r, de forma similar ao algoritmo *i*-TSAB. Essa função utiliza o operador $N1$ para a geração da solução ϕ. A figura 32 apresenta o pseudocódigo da função *NIS* do ei-TSAB. Utiliza-se uma medida de distância $D(\alpha, \beta)$ que retorna a distância (em número de trocas de arcos) entre as soluções α e β. A função executa $maxV \cdot D(\alpha, \beta)$ iterações, sendo que se um movimento possuir um f_π menor que f_π^r, a função pára, retornando esse movimento.

Fazer $\pi \leftarrow \alpha$, $iter \leftarrow 0$;
Encontrar Distância D entre α e β;
Construir $G(\pi)$ e calcular o tempo de cada *setup* com *mSwitch*;
Encontrar caminho crítico e f_π com *mPath*;
repetir
  $iter \leftarrow iter + 1$; Encontrar $N(\pi)$ com operador $N1$;
  Para cada $v \in N(\pi)$ calcular f_{π_v};
  Selecionar um subconjunto $K \leftarrow N^+ \in N(\pi)$ de soluções que tem a menor distância de α;
  se $N^+ = \varnothing$ **então**
    $K \leftarrow N(\pi)$;
  fim se
  Escolher o vizinho w tal que $f_{\pi_w} = \min_{v \in K}\{f_{\pi_v}\}$;
  Modificar: (1) $G(\pi)$ para $G(\pi_w)$; (2) caminho crítico;
  Fazer $\pi = \pi_w$ e $\phi = \pi$;
  se $f_\pi < f_\pi^r$ **então**
    Fazer $f_\pi^r \leftarrow f_\pi$ e Parar.
  fim se
até $iter \geq maxV \cdot D(\alpha, \beta)$

Figura 32: Função *NIS* para o algoritmo ei-TSAB.

Define-se a meta-heurística baseada no algoritmo TSAB, expandida para considerar as restrições do PEM (diferenciada aqui pela sigla *e*TSAB). Considerando; π^B e f_π^B como a melhor ordem de processamento encontrada e o seu valor da função objetivo, respectivamente; a lista tabu LT inicia vazia; TSAB inicia com uma ordem de processamento ϕ e retorna π^B e f_π^B, executando $MaxIter$ iterações sem encontrar melhora em f_π^B. A figura 33 ilustra o pseudocódigo do algoritmo.

O algoritmo ei-TSAB é apresentado na figura 34. Considera-se uma solução inicial π^0 gerada por um método externo. A técnica retorna a ordem de processamento π^* e seu *makespan*

Fazer $\pi = \phi$ e $iter = 0$
Construir o grafo $G(\pi)$ e calcular o tempo de cada *setup* com *mSwitch*;
Encontrar caminho crítico e f_π com *mPath*;
Fazer $C \leftarrow f_\pi$ e $(\pi^B, f_\pi^B) \leftarrow (\pi, C)$;
repetir
  $iter \leftarrow iter + 1$;
  Encontrar $N(\pi)$ com operador $N5$;
  Para cada $v \in N(\pi)$ calcular f_{π_v};
  Encontrar o movimento $w \in N(\pi)$ com base em f_{π_v} e na lista tabu LT;
  Modificar: (1) a lista tabu LT; (2) $G(\pi)$ para $G(\pi_w$; (3) caminho crítico;
  Fazer $C \leftarrow f_{\pi_w}$ e $\pi \leftarrow \pi_w$;
  se $C < f_\pi^B$ **então**
    Fazer $(\pi^B, f_\pi^B) \leftarrow (\pi, C)$ e $iter = 0$
  fim se
até $iter \geq MaxIter$

Figura 33: Função *e*TSAB.

f_π^*.

Fazer $(\pi^1, f_\pi^1) \leftarrow eTSAB(\pi^0)$ e $f_\pi^* \leftarrow f_\pi^1$;
para cada cada $i = 2, \ldots, maxE$ **faça**
  Fazer $\phi = NIS(\pi^{i-1}, \pi^0, f_\pi^*)$;
  Fazer $(\pi^i, f_\pi^i) \leftarrow eTSAB(\phi)$ e $f_\pi^* = \min\{f_\pi^*, f_\pi^i\}$;
fim para cada
Encontrar $1 \leq k \leq maxE$ tal que $f_\pi^k = f_\pi^*$; $\pi^* = \pi^k$;
repetir
  Encontrar $1 \leq l \leq maxE$ tal que $D(\pi^k, \pi^l) = \max\{D(\pi_k, \pi^i) \ : \ 1 \leq i \leq maxE\}$;
  Fazer $phi \leftarrow NIS(\pi^k, \pi^l, f_\pi^*)$ e $(\pi^l, f_\pi^l) \leftarrow eTSAB(phi)$;
  se $f_\pi^l < f_{pi}^k$ **então**
    $(\pi^*, f_\pi^*) \leftarrow (\pi^l, f_\pi^l)$ e $k \leftarrow l$;
  fim se
até $\max\{D(\pi^k, \pi^i) \ : \ 1 \leq i \leq maxE\} < maxD$.

Figura 34: Algoritmo ei-TSAB.

Nota-se que o *framework* proposto por Watson, Howe e Whitley tem uma estrutura semelhante ao algoritmo descrito na figura 34, com a diferença que, no lugar da função *e*TSAB tem-se uma outra meta-heurística. Considerando essa idéia, o algoritmo proposto pode considerar outros métodos heurísticos, como os já vistos no capítulo 5.

Espera-se que o algoritmo ei-TSAB tenha um desempenho semelhante ao do *i*-TSAB,

uma vez que ele é uma adaptação desse último para o PEM. As propriedades da vizinhança gerada através do operador $N5$ permitem cálculos mais rápidos dos valores de *makespan*, atraso e trocas de ferramentas. No capítulo a seguir será descrita a arquitetura do modelo computacional proposto.

7 MODELO PROPOSTO

Para tratar o problema de Escalonamento em um *Job Shop* considerando restrições de datas de entrega, turnos de produção e trocas de ferramentas, propõe-se a utilização das técnicas de Análise de Agrupamentos e Busca Tabu, como descrito no capítulo 6.

A implementação dessas duas técnicas é feita na forma de um modelo computacional, composto de quatro módulos, que é descrito nesse capítulo. Essa implementação torna possível que sejam realizados experimentos onde o comportamento das variáveis da função objetivo apresentada no capítulo 5 são analisadas.

7.1 Arquitetura do Modelo

O modelo proposto é composto de quatro módulos: (*i*) adaptação de problemas-teste ao PEM, (*ii*) geração de Famílias de Operações, (*iii*) geração da solução inicial e (*iv*) escalonamento da produção. A execução desses módulos é sequencial, iniciando com a adaptação de um problema-testes já existente, ao PEM e finalizando com a apresentação do escalonamento que considera as restrições impostas pelo problema. Nos itens a seguir esses módulos são detalhados.

7.1.1 Adaptação de Problemas-teste ao PEM

O primeiro módulo do modelo proposto tem a função de adaptar problemas-teste (ou estudo de casos de PEJS) existentes na literatura, acrescendo os dados referentes ao PEM. Os problemas-teste adaptados possuem as seguintes informações:

- Número de partes e máquinas;
- Precedência das operações;
- Tempo de processamento das operações;

- Data de entrega das partes;
- Ferramentas requeridas por cada operação.

Os problemas-teste obedecem o formato apresentado na figura 35. A maioria dos autores citados no estado da arte apresentado no capítulo 5 utilizam problemas-teste de *benchmark* para o PEJS. Em seu trabalho, Jain (JAIN; MEERAN, 1999) faz uma relação desses problemas-teste, cujas dimensões estão no formato partes $\times$ máquinas:

- Fisher e Thompson: três problemas-teste de tamanhos 6×6, 10×10 e 20×20 propostas em 1963. Esses problemas-teste são referidas pela sigla FT3, FT10 e FT20 e receberam grande atenção da comunidade acadêmica. O problema-teste FT10 é uma das mais utilizadas pelos autores, sendo que a solução ótima conhecida para esse problema-teste foi encontrada somente em 1987, enquanto que os problemas-teste FT6 e FT20 foram resolvidas em 1975.
- Lawrence: são propostos 40 problemas-teste de 8 tamanhos diferentes (10×5, 15×5, 20×5, 10×10, 15×10, 20×10, 30×10 e 15×15), sendo referidas pela sigla LA.
- Adams *et al*: cinco problemas-teste de dois diferentes tamanhos propostas em 1988 (10×10 e 20×15) referenciadas pela sigla ABZ.
- Applegate e Cook: os autores propuseram 10 problemas-teste de tamanho 10×10, referidas pela sigla ORB.
- Storer *et al*: apresentam vinte problemas-teste de três diferentes tamanhos (20×10, 20×15, 50×10), sendo referidas pela sigla SWV.
- Yamada e Nakano: oito problemas-teste com dimensões 20×20, referidas pela sigla YN.
- Taillard: são propostos oitenta problemas-teste de oito dimensões diferentes (15×15, 20×15, 20×20, 30×15, 30×20, 50×15, 50×20 e 100×20). Esses problemas-teste são comumente referidas pela sigla TD.
- Demirkol *et al*: oitenta problemas de oito diferentes dimensões (20×15, 20×20, 30×15, 30×20, 40×15, 40×20, 50×15 e 50×20), referidas pela sigla DMU.

Os problemas-teste descritos acima podem ser utilizadas no modelo proposto mediante a adição das informações de datas de entrega de cada parte e ferramentas de cada operação. A modificação desses problemas-teste é feita no módulo 1 do modelo proposto. A partir de

um arquivo que contenha as informações de número de partes e máquinas, precedência das operações de cada parte e tempos de processamento, são geradas datas de entrega para cada parte e ferramentas para cada operação.

Figura 35: Formato dos problemas-teste do PEM .

As datas de entrega dos produtos e as ferramentas das operações são geradas segundo uma distribuição uniforme, considerando os parâmetros definidos pelo usuário. Os dados de entrada do módulo 1 são os seguintes:

- Número de partes, máquinas e operações;
- Precedência das operações e tempos de processamento;
- Quantidade total de ferramentas das operações;
- Limite do magazine da máquina versátil;
- Média para as datas de entrega das partes.

Assim, os problemas-teste podem ser utilizadas na validação parcial do modelo proposto e nos experimentos com políticas de minimização, que são descritas no próximo capítulo. A validação é uma etapa que envolve os módulos 2 (geração de FOs) e 4 (geração do escalonamento considerando f_π.

7.1.2 Geração de Famílias de Operações

O módulo 2 tem por função gerar a informação de Famílias de Operações utilizando o Algoritmo de Identificação de Grupos Modificado, apresentado no capítulo 6. Este módulo recebe um arquivo de entrada gerado pelo módulo 1. A saída deste módulo é uma estrutura de dados que define a quantidade de ferramentas trocadas entre FOs e a FO de cada uma das operações.

7.1.3 Geração da Solução Inicial

Uma vez geradas as FOs, a solução inicial pode ser construída pelo módulo 3. Essa solução inicial é um escalonamento viável gerado por uma regra simples, onde são seqüenciadas as operações da parte 1 até as operações da parte n. A saída deste módulo é a seqüência inicial de operações que iniciará o processo de escalonamento realizado no módulo 4.

7.1.4 Geração do Escalonamento da Produção

A execução dos três primeiros módulos gera os dados necessários para que o problema-teste possa ser tratada, no módulo 4. Este módulo implementa do algoritmo ei-TSAB descrito no capítulo 6 e possui os duas categorias de parâmetros: parâmetros do PEM e parâmetros da heurística.

Os parâmetros do PEM referem-se ao gerenciamento das variáveis de decisão e das restrições do problema abordado, sendo os seguintes:

- W_1, W_2 e W_3: são os pesos de cada uma das variáveis de decisão da função objetivo f_π, definida no capítulo 5. Através desses pesos pode-se fazer o gerenciamento das políticas de otimização.
- Tempo do turno de produção;
- Tempos α e β referentes ao *setup*: referentes ao tempo fixo para a parada da máquina e limpeza da área de trabalho e o tempo gasto na troca de uma ferramenta.

Os parâmetros da heurística referem-se a técnica ei-TSAB previamente descrita, sendo os seguintes:

- Número de iterações sem encontrar melhora na função objetivo (*nbmax*);

- Tamanho da lista de movimentos proibidos (*LT*);
- Tamanho da lista de regiões a serem intensificadas (*maxL*);
- Tamanho da lista de soluções de elite a serem intensificadas (*maxE*);
- Distância entre soluções para o procedimento de *path-relinking* (*maxV*);
- Distância mínima entre as soluções na lista de elite (*maxD*);
- Tamanho da lista de detecção de ciclo (*maxC*).

A figura 36 mostra a representação do fluxo de dados entre os módulos e dos parâmetros de entrada de cada módulo. Nesta figura está representada a etapa de validação do modelo, ligada aos módulos 2 e 4.

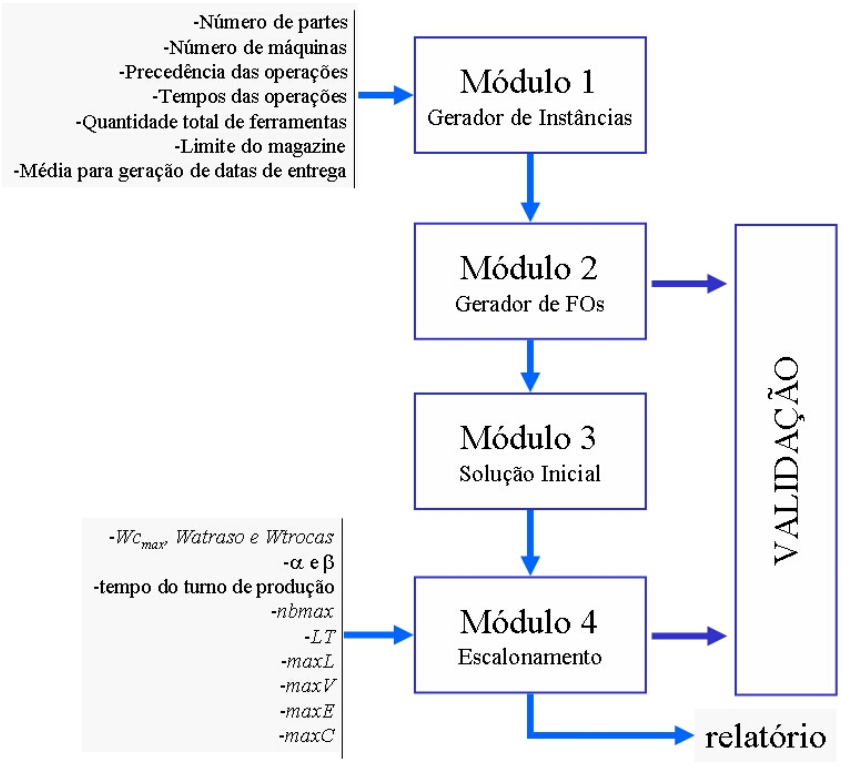


Figura 36: Arquitetura do Modelo Proposto.

Considerando a arquitetura do modelo proposto, são apresentados no capítulo a seguir a validação e os resultados dos experimentos realizados com o modelo implementado.

8 EXPERIMENTOS

Para tratar o Problema de Escalonamento em um *Job Shop* considerando restrições de datas de entrega, turnos de produção e tempo de *setup*, foi desenvolvido um modelo computacional que implementa as técnicas de Análise de Agrupamentos e Busca Tabu. Esse modelo é composto de quatro módulos, como apresentado no capítulo anterior.

Neste capítulo é apresentada a implementação, a validação e a análise do comportamento do modelo frente a variações nos pesos de f_π. Os testes e a validação foram feitos utilizando problemas-teste de *benchmark* adaptados à estrutura do PEM, e a análise do modelo é feita frente diferentes políticas de minimização dos tempos de produção.

8.1 Implementação e Validação

A implementação do modelo computacional descrito no capítulo 7 foi feita em linguagem C e C++ e os experimentos foram rodados em uma máquina com dois processadores Pentium III 800Mhz, 512Mb de memória RAM.

A validação do modelo computacional foi feita em duas etapas: validação da geração de Famílias de Operações e validação do escalonamento. Na primeira etapa, o módulo de geração de FOs utilizou um problema-teste de 10 partes $\times$ 10 ferramentas proposto por Tang e Denardo, também utilizada por Gómez em seu trabalho (GÓMEZ, 1996). Na segunda etapa, é feita a validação parcial do módulo de escalonamento, em duas sub-etapas. Na primeira, compara-se o resultado obtido com o modelo implementado com o trabalho de Hertz e Widmer (HERTZ; WIDMER, 1996), na minimização da variável $Setup$. Na segunda sub-etapa, utilizam-se problemas-teste de *benchmark* para validar a minimização da variável $Makespan$.

Na figura 37 é apresentada a matriz do tipo partes *versus* ferramentas, utilizada na primeira etapa da validação. O número ótimo de FPs para este problema-teste é 5, resultado encontrado pelo modelo de Gómez (GÓMEZ, 1996). O módulo 2 do modelo computacional gera o mesmo número de FPs, mostrado na figura 38.

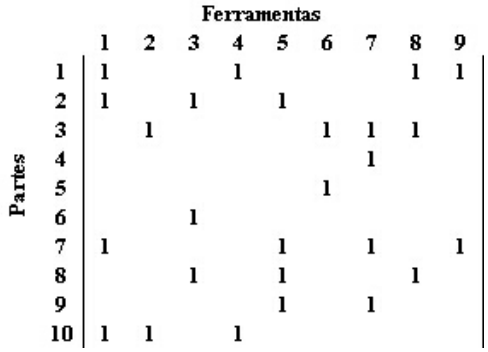


Figura 37: Problema-teste com estrutura tipo $partes \times ferramentas$ proposto por Tang e Denardo, utilizado por Gómez (GÓMEZ, 1996).

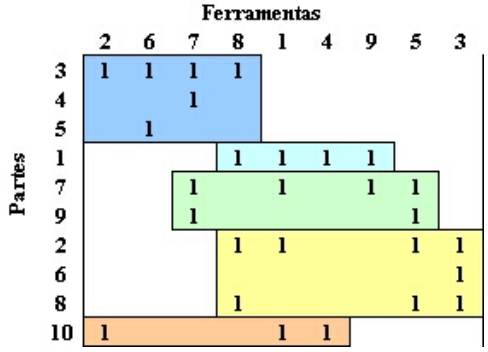


Figura 38: Resultado obtido pelo modelo para o problema-teste proposto por Tang e Denardo.

Para a etapa de avaliação do Módulo de Escalonamento, foram feitos experimentos onde foi validada a modelagem da variável *Setup*. Esta validação é feita pela comparação dos resultados obtidos por Hertz e Widmer em seu trabalho (HERTZ; WIDMER, 1996), onde os autores modificaram os problemas-teste de *benchmark*, inserindo a restrição de tempo de *setup*.

No experimento realizado pelos autores, cerca de 45 problemas-teste foram modificados, sendo 40 problemas-teste de Lawrence e 5 de Adams *et al* (JAIN; MEERAN, 1999). Os autores trabalham com as seguintes suposições:

- nenhuma ferramenta é requerida por mais de uma operação;
- Seja $\{o_1, \ldots, o_r\}$ o conjunto ordenado de operações de uma parte p, cujo número de ferramentas requeridas pela operação $o_i (1 \leq i \leq r)$ é i;
- a capacidade do magazine de uma máquina é igual ao número de máquinas;
- Seja T o tempo de processamento médio de uma operação, definido como

$$T = \sum_{i=1}^{O} p_i / O, \tag{8.1}$$

e assume-se que $\alpha = T/10$ e $\beta = 0$.

As ferramentas foram geradas sob uma distribuição uniforme. Foram realizados 10 execuções do modelo proposto pelos autores para cada problema-teste. No seu trabalho, Hertz e Widmer apresentam os melhores resultados obtidos nas execuções do modelo e um percentual de quantas vezes esses resultados foram obtidos nas execuções realizadas.

Para validar o modelo computacional implementado neste trabalho, escolheu-se 11 dos problemas-teste utilizados por Hertz e Widmer. Foram geradas ferramentas para cada operação, segundo as restrições acima citadas. Os parâmetros utilizados para validar o modelo foram escolhidos segundo o trabalho de Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 1996) (NOWICKI; SMUTNICKI, 2002), sendo os seguintes:

- $nbmax = 5000$;
- $LT = 10$;
- $maxL = 5$;
- $maxE = 5$;
- $maxV = 0.5$;
- $maxD = 5$.

A variável atraso foi desconsiderada, sendo atribuído valor zero ao seu peso. Para a variável *Makespan* o peso atribuído foi 2 e para a variável *Setup* o peso atribuído foi 1, como no trabalho de Hertz e Widmer. Nos experimentos realizados, o modelo ei-TSAB apresentou resultado no mínimo similar ao modelo proposto por Hertz e Widmer. Na tabela 4 são apresentados os melhores resultados dos experimentos realizados, encontrados por Hertz e Widmer e obtidos com a execução do ei-TSAB.

Na segunda etapa de validação do módulo de escalonamento, foram escolhidas os três problemas-teste propostos por Fisher e Thompson (JAIN; MEERAN, 1999): FT6, FT10 e FT20. O problema-teste FT6 é composto por 6 partes em 6 máquinas, totalizando 36 operações; o problema-teste FT10 é composto por 10 partes e 10 máquinas, totalizando 100 operações e o problema-teste FT20 é composto de 20 partes e 5 máquinas, totalizando 100 operações Esses problemas-teste possuem o tempo total de produção ótimo conhecido, apresentado na tabela 5.

Esses problemas-teste de *benchmark* são referentes ao PEJS, cujo objetivo é minimizar o tempo total de produção, sem considerar outros fatores, como as restrições do PEM. Assim

Tabela 4: Valores da função objetivo (*makespan*) obtidos por Hertz e Widmer (HERTZ; WIDMER, 1996) e pela técnica ei-TSAB

Problema-teste	Hertz e Widmer	ei-TSAB
LA16	963	961
LA17	793	789
LA18	876	863
LA19	870	859
LA21	1097	1091
LA22	971	971
ABZ5	1271	1261
ABZ6	970	963
ABZ7	691	685
ABZ8	701	697
ABZ9	717	706

Tabela 5: Valores ótimos conhecidos para os problemas-teste propostos por Fisher e Thompson, em unidades de tempo

Problema-teste	*makespan*
FT6	55
FT10	930
FT20	1165

sendo, foi realizada uma validação parcial do modelo, referente somente ao comportamento observado na variável *Makespan*. Os problemas-teste foram modificadas pela inclusão das restrições do PEM através da execução do módulo 1. Para cada problema-teste foram definidos um conjunto de 10 ferramentas, um limite de magazine de 4 ferramentas e datas de entrega geradas segundo uma distribuição uniforme, segundo os seguintes intervalos: 10 a 60 (FT6), 10 a 1000 (FT10) e 10 a 2000 (FT20). Os parâmetros da busca que foram utilizados na validação são os mesmos da etapa de validação anterior.

Desconsiderando a restrição de turnos de produção e atribuindo valor zero aos pesos das variáveis de decisão *Atraso* e *Trocas*, o modelo proposto alcança o resultado ótimo conhecido para os três problemas-teste utilizados na validação.

Na implementação do modelo computacional, notou-se que a consideração da restrição de turnos de produção, do atraso das partes e do tempo de *setup* introduzem maior complexidade ao modelo, o que impede que se consiga a mesma eficiência computacional obtida por Nowicki e Smutnicki em sua técnica, o *i*-TSAB. Tal eficiência é baseada nas propriedades da vizinhança gerada através da consideração de um caminho crítico e da utilização do operador de geração de vizinhança N5.

O aumento da complexidade computacional devido a consideração das restrições do

PEM pôde ser observada nos testes realizados com os problemas-teste de *benchmark*. A implementação dos aceleradores descritos no trabalho de Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 2002) foi difícil, e dada a complexidade adicional das restrições do PEM, os tempos computacionais observados nos testes variaram de um minuto (para problemas-teste com 225 operações) até 15 minutos (para problemas-teste de 600 operações). Outro fator que contribui para o aumento do tempo computacional são as operações de *setup*, que fazem com que o número de operações a serem verificadas aumente quase duas vezes. Dado esse fator, nos experimentos apresentados a seguir, não foram utilizados os mesmos parâmetros descritos por Nowicki e Smutnicki em seu trabalho. O número de áreas de re-intensificação foi o fator que sofreu maior redução, dado que aumenta sensivelmente o número de interações da busca. Os valores foram reduzidos para que o tempo de execução fosse menor. Experimentos adicionais foram feitos, onde para problemas-teste de menor porte os parâmetros foram aumentados.

Nas seções a seguir são discutidos os experimentos que têm por objetivo analisar o comportamento do modelo implementado frente a três políticas de minimização dos tempos de produção. Essas políticas de minimização são definidas pelos pesos das variáveis de decisão da função objetivo F, apresentada no capítulo 5.

8.2 Políticas de Otimização

A função objetivo definida para classificar o escalonamento obtido pelo modelo permite que sejam definidas três políticas de minimização: (*i*) do tempo total de produção, (*ii*) do atraso das partes e (*iii*) do tempo dos *setups*. O gerenciamento dessas políticas se dá pela atribuição de valores para os pesos das variáveis de decisão da função objetivo f_π.

Para analisar como a adoção de uma determinada política influencia o comportamento das variáveis de decisão de f_π, são definidos três tipos de experimentos, um para cada tipo de política. Nestes experimentos são observados como o aumento do valor dos pesos de uma determinada variável influencia no comportamento do modelo. É analisado também o impacto das alterações feitas nos parâmetros da técnica utilizada para tratar o problema. Inicialmente são definidas os problemas-teste utilizados e os parâmetros com os quais os experimentos serão realizados, considerando as restrições do problema apresentado no capítulo 5.

Para realizar uma análise qualitativa das políticas de minimização, define-se uma solução de referência obtida pelo modelo proposto. Os resultados obtidos com os valores dos pesos dessa solução serão comparados com os resultados obtidos em cada uma das políticas de otimização definidas.

8.2.1 Problemas-teste e Parâmetros Utilizados nos Experimentos

Para a realização dos experimentos, foram escolhidas seis dos problemas-teste propostos por Taillard (TAILLARD, 2006), de três dimensões diferentes, modificadas de forma a se adequarem ao PEM. Na tabela 6 são apresentados os códigos de cada problema-teste e suas dimensões. O sub-índice desse código refere-se ao número do problema-teste disponibilizado por Taillard. Neste caso, foram utilizadas a primeira e a segundo problema-teste de cada dimensão escolhida.

Tabela 6: Código e dimensões das problemas-teste utilizados nos experimentos.

Código	Dimensões
$ta1515_1$ e $ta1515_2$	15 partes, 15 máquinas, 225 operações
$ta3020_1$ e $ta3020_2$	30 partes, 20 máquinas, 600 operações
$ta5015_1$ e $ta5015_2$	50 partes, 15 máquinas, 750 operações

Foram definidos os seguintes parâmetros referentes às restrições do problema proposto:

- Turno de produção $\eta = 480$ minutos (8 horas);
- Tempos referentes ao *setup*: tempo de parada para o reinício da máquina e limpeza da área de trabalho $\alpha = 5$ minutos; tempo de troca de uma ferramenta $\beta = 4$ minutos;
- Limite do magazine das máquinas versáteis $\zeta = 4$;
- Número total de ferramentas necessárias para o processamento das partes é 10;

Os valores utilizados nos parâmetros do modelo computacional são os seguintes:

- $nbmax = 5000$;
- $LT = 5$;
- $maxL = 1$;
- $maxC = 100$;
- $maxE = 4$;
- $maxV = 0,5$;
- $maxD = 5$;

8.2.2 Solução Não-Tendenciosa

Para realizar comparações entre os resultados obtidos nas diferentes políticas de minimização dos tempos de produção do PEM, é definida uma solução na qual as variáveis de decisão têm a mesma contribuição no valor de f_π, denominada Solução Não-Tendenciosa (SNT). Essa solução consiste em uma configuração de pesos, para as variáveis de f_π, na qual nenhuma das parcelas da função objetivo é privilegiada.

Para obter essa solução, foi realizado um experimento com seis problemas-teste de três diferentes dimensões fornecidas por Taillard (TAILLARD, 2006) e modificadas para se adequarem ao PEM. Nestes experimentos foram considerados os parâmetros apresentados no item anterior.

Para cada problema-teste foram realizados 100 experimentos onde os valores dos pesos das variáveis de decisão de f_π foram variados segundo uma distribuição uniforme em um intervalo de 0 a 100. Para cada diferente dimensão é feita a média dos valores assumidos pelas variáveis de decisão nas diferentes execuções do modelo proposto. Nos experimentos realizados, a variável *Atraso* assumiu os maiores valores, e sendo assim, são atribuídos pesos proporcionais para as variáveis *Makespan* e *Trocas*. Na tabela 7 estão os valores dos pesos da SNT para as diferentes dimensões dos problemas-teste analisados.

Tabela 7: Valor dos pesos das variáveis de decisão de f_π para a SNT.

Dimensão	*Makespan*	*Atraso*	*Trocas*
15×15	5	1	14
30×20	15	1	29
50×15	21	1	46

Com base nesses pesos, foram geradas SNTs para os 6 problemas-teste utilizados nos experimentos, utilizando o modelo computacional proposto. Nas tabelas 8 e 9, são apresentados os valores de f_π e de suas parcelas, obtidos pela técnica de solução inicial e pela ei-TSAB.

Problemas-teste	f_π	*Makespan*	*Atraso*	*Setup*
$ta1515_1$	169926	10767	73041	2870
$ta1515_2$	171669	10216	78214	2825
$ta3020_1$	951239	27250	393643	7834
$ta3020_2$	1041034	27990	390054	7970
$ta5015_1$	1946139	33637	777600	10047
$ta5015_2$	1855412	31586	734498	9948

Tabela 8: Valores de f_π obtidos pelo método de solução incial, considerando pesos da SNT.

Nos experimentos acima descritos, o escalonamento gerado pela técnica ei-TSAB, uti-

lizando os pesos da SNT, reduz o valor da função objetivo em média 65,5% (com um desvio padrão médio de 2%). As variáveis de decisão *Makespan*, *Atraso* e *Setup*, em média, sofrem reduções de 83,4%, 82,3% e 6.5%, respectivamente, com desvios médios semelhantes ao de f_π.

Nos experimentos apresentados a seguir, é analisado o comportamento das variáveis de decisão frente a políticas de minimização. Esses experimentos são feitos da seguinte forma: o valor do peso da variável a ser privilegiada é aumentando, enquanto que os valores dos pesos das outras duas variáveis são mantidos constantes. Os valores nos quais os pesos das variáveis ficam fixados são definidos na tabela 7.

Para cada uma dos seis Problemas-teste apresentados, foram realizados experimentos onde o peso da variável a ser privilegiada assume os seguintes valores: 100, 500 e 1000. Nos itens a seguirão apresentados os resultados obtidos para o problema-teste $ta1515_1$. As tabelas que descrevem os resultados obtidos com as outros problemas-teste se encontram nos apêndices A, B e C.

8.2.3 Minimização do *Makespan*

Na tabela 10 são apresentados os resultados obtidos no experimento com a política de minimização do tempo total de produção.

Nos experimentos realizados onde a variável *Makespan* é privilegiada, nota-se que ocorre uma redução média de 87,4% em relação à Solução Inicial na variável *Makespan* (com desvio de 2%), 87,7% na variável *Atraso* (com desvio de 3%) e 3,2% na variável *Setup* (com

Tabela 9: Valores de F obtidos pelo método ei-TSAB, considerando pesos da SNT.

Problema-teste	f_π	*Makespan*	*Atraso*	*Trocas*
$ta1515_1$	52877	1580	6532	2563
$ta1515_2$	57095	1563	10640	2576
$ta3020_1$	343502	5957	112293	7466
$ta3020_2$	360606	4466	72520	7624
$ta5015_1$	669888	5061	129275	9442
$ta5015_2$	689885	5226	145025	9459

Tabela 10: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta1515_1$, considerando a política de minimização do *Makespan*.

Valor de W_1	*Makespan*	*Atraso*	*Trocas*
100	1529	6400	2730
500	1523	6090	2714
1000	1523	6771	2726

desvio de 0.9%). Comparando os resultados desta política com os obtidos com os pesos da SNT, as variações são mais modestas: redução média de 3,6% (desvio de 19%) e 6,7% (desvio de 22%) para as variáveis *Makespan* e *Atraso*, respectivamente, e um aumento de 5,8% (com desvio de 2%) para a variável *Setup*. O desvio-padrão mais acentuado observado na comparação feita entre essa política de minimização e a SNT se deve ao fato de que, para alguns problemas-teste, a redução chega a 50%.

O gráfico apresentado na figura 39 ilustra o comportamento das variáveis de decisão de f_π, considerando os experimentos da tabela 10 e os valores da Solução Inicial e da SNT, para o problema-teste $ta1515_1$.

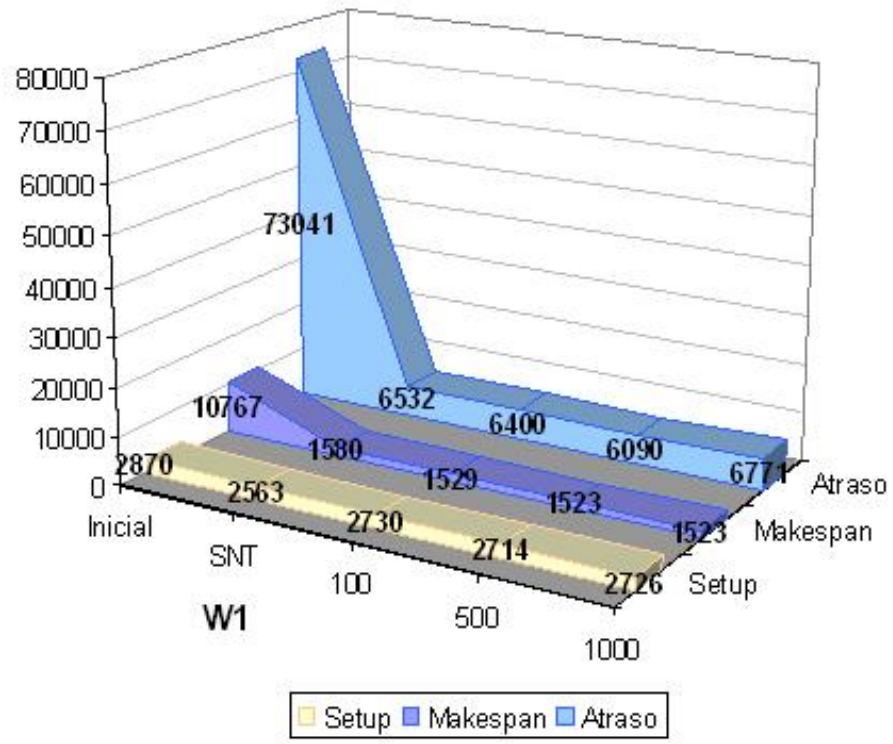


Figura 39: Comparação entre os valores das variáveis de decisão de f_π, a Solução Inicial e a SNT, considerando a política de minimização do *Makespan*, para o problema-teste $ta1515_1$.

Nos resultados apresentados na tabela 10, nota-se uma sensível redução na variável *Makespan* com o aumento do peso dessa variável de 100 para 500. Para um valor mais alto, porém, não se observa melhora no comportamento da variável. Para outros problemas-teste, porém, o mesmo comportamento não é observado. Por exemplo, para o problema-teste $ta3020_1$, cuja tabela é apresentada no apendice A, o melhor valor para a variável *Makespan* é obtido com a atribuição do valor 500 para o peso W_1, enquanto que um valor pior é obtido ao se atribuir o valor 1000 para este peso.

Outro fator observado na maioria dos problemas-teste, nos experimentos, é que a redução do tempo total de produção contribui para a redução do atraso das partes. Ou seja, no momento em que o tempo necessário para completar o processamento de todas as partes é menor, o atraso da produção é reduzido também.

O tempo de *setup*, por outro lado, sofre um aumento dado essa política de minimização. Isso se deve ao fato de que, o tempo total de *setup* não depende apenas do valor dos tempos das operações do caminho crítico, que é a base para a geração da vizinhança da técnica escolhida. O tempo de *setup* depende do tipo de FO de cada operação, e dos tempos ociosos entre as operações de todo o escalonamento.

8.2.4 Minimização do Atraso

Nestes experimentos o valor do peso da variável *Atraso* é aumentado enquanto que os valores dos pesos das variáveis *Makespan* e *Setup* são mantidos fixos nos valores da SNT. a tabela 11 mostra o comportamento das variáveis de f_π observado nestes experimentos.

Tabela 11: Valores da variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta1515_1$, considerando a política de minimização do Atraso.

Valor de W_2	*Makespan*	*Atraso*	*Trocas*
100	1558	5481	2782
500	1549	5643	2766
1000	1561	5443	2713

Fazendo uma comparação com o método de Solução Inicial, o aumento do valor do peso W_2 melhora o valor da variável *Atraso* em média 81,3% (com um desvio de 31,5%, em sua grande maioria, para reduções maiores, chegando a 92%). A variável *Makespan* sofre uma redução média de 82% (com desvio padrão semelhante ao da variável *Atraso*), e a variável *Setup* sofre uma redução média de 2,1% (com desvio de 2%).

O gráfico apresentado na figura 40 mostra os resultados obtidos pela Solução Inicial, pelo modelo utilizando os pesos da SNT e utilizando os pesos da política de minimização do Atraso.

Comparado-se os resultados obtidos nesta política com a SNT, observa-se o seguinte: para problemas-teste de 225 e 600 operações, o atraso sofre pequena redução (no máximo 30%). Para os problemas-teste com 750 operações, a variável *Atraso* assume valores até 65% maiores que o valor obtido com os pesos da SNT. Deve ser levado em consideração que, para os experimentos apresentados aqui, os valores dos parâmetros da técnica utilizada são relativamente

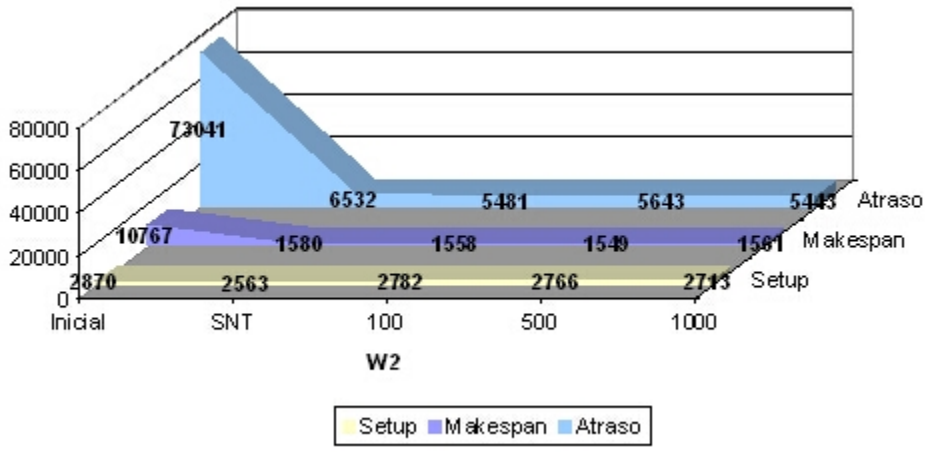


Figura 40: Comparação entre os valores das variáveis de decisão de f_π, a Solução Inicial e a SNT, considerando a política de minimização do Atraso, para o problema-teste $ta1515_1$.

menores que os utilizados originalmente pelos autores (NOWICKI; SMUTNICKI, 2002). Esse fator tem um impacto na obtenção de melhores resultados para f_π.

Outro fator observado em alguns problemas-teste é a definição de valores menores para W_2 resulta na redução da variável que está sendo privilegiada. Esse comportamento se deve ao fato de que a variável *Atraso* depende do valor de atraso de todas as últimas operações de todos os produtos, estando vinculada ao caminho crítico apenas através das operações que estiverem nele. A redução do caminho crítico pode acarretar na redução do atraso das partes se, neste caminho, estiverem operações finais de partes que tenham atraso alto. As partes que possuem operações finais fora do caminho crítico não serão movimentadas pelo critério de geração de vizinhança, independente do valor atribuído ao peso da variável *Atraso*. A redução significativa do tamanho do caminho crítico, que promove a diminuição do atraso das partes com últimas operações nele, pode ser notada quando se observam a variação dos valores da variável *Atraso* nesta política de minimização e na política de minimização do *Makespan*. A variável em questão sofre redução média de 87,4% considerando a minimizaçao do *Makespan*, contra 81,3% na minimização do *Atraso*. Mesmo o desvio padrão da redução média nesta última política seja alto (cerca de 30%), somente para os problemas-teste de 225 operações os valores da variável *Atraso* são melhores que os obtidos na política de minimização do *Makespan*.

8.2.5 Minimização do Tempo de *Setup*

Da mesma forma que nos experimentos anteriores, os valores do peso da variável *Setup* são aumentados, de modo a implementar a política de minimização que privilegie a redução do tempo de *setup*. A aplicação desta política, no entanto, não contribuiu para melhoras significa-

tivas no comportamento do modelo, se comparado com a SNT e as demais políticas. Na tabela 12 são apresentados os resultados observados nos experimentos.

Tabela 12: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta1515_1$, considerando a política de minimização do Tempo de *Setup*.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
100	1651	11452	2585
500	2517	17598	2424
1000	3222	19769	2471

Observou-se nos experimentos realizados que as variáveis *Makespan*, *Atraso* e *Setup* sofrem reduções de 79,8%, 78,9% e 14,4%, respectivamente (com respectivos desvios de 1,6%, 2,9% e 3,9%) em relação à Solução Inicial. Comparando-se com a SNT, a variável *Setup* sofre uma redução de 8,4%. A variável *Makespan* sofre um acréscimo médio de 24,2%, salvo no problema-teste $ta3020_1$, onde é reduzido em 7%. A variável *Atraso* sofre redução de até 28,7% nos problemas-teste de 600 operações, sendo que nos problemas-teste restantes sofre acréscimo de 100,3%. O gráfico apresentado na figura 41 é referente aos resultados obtidos com a política de minimização do tempo *setup* para o problema-teste $ta1515_1$.

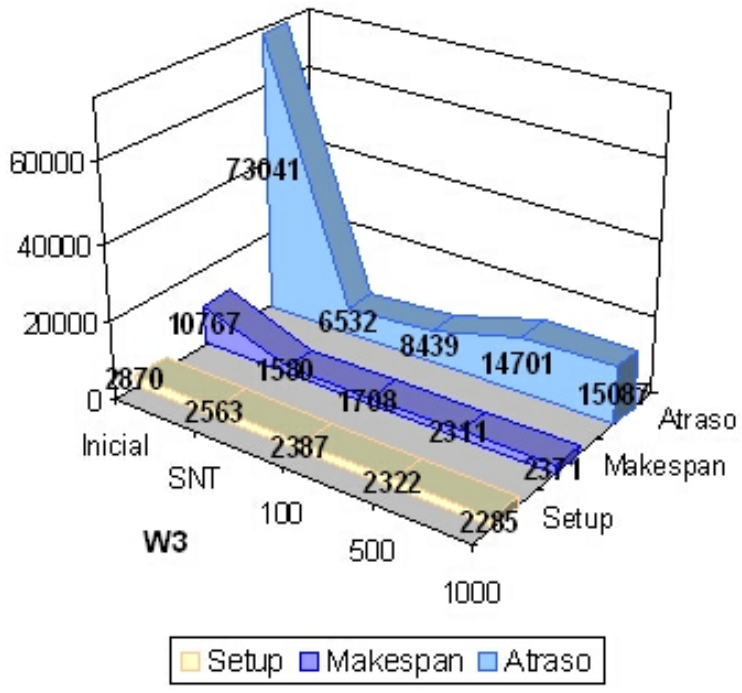


Figura 41: Comparação entre os valores das variáveis de decisão de f_π, a Solução Inicial e a SNT, considerando a política de minimização do Tempo de *Setup*, para o problema-teste $ta1515_1$.

O tempo de *setup* sofre acréscimo sempre que duas operações de FOs diferentes são processadas na mesma máquina, em seqüência, fazendo com que a máquina pare para trocar

ferramentas. Essa troca pode ocorrer em tempos ociosos, quando uma máquina libera uma parte e fica esperando a chegada de outra, ou seja, não se pode afirmar que toda a ocorrência de *Setup* aumente o tempo total de produção.

A ocorrência de um *setup* é representada por uma operação cujo tempo de processamento é definido pela fórmula $\alpha + \beta r$, apresentada no capítulo 6. O tempo total de *Setup* é a soma dos tempos de todas as operações de *setup* da produção, sendo que somente algumas delas pertencerão ao caminho crítico. A redução do tempo de *setup* ocorre quando é realizada uma troca entre duas operações do caminho crítico de forma que operações de mesma FO fiquem em seqüência. Independente do valor atribuído ao peso da variável *Setup*, as operações de *setup* que não fazem parte do caminho crítico não serão eliminadas.

Para exemplificar essa situação, considera-se um determinado escalonamento para o problema-teste FT6, devidamente alterada para adaptar-se ao PEM, representada no grafo apresentado na figura 42. As operações em vermelho representam a ocorrência de *setup*. As operações em negrito representam o caminho crítico, onde a vizinhança é gerada, no método ei-TSAB. Observa-se que a troca de operações somente no caminho crítico não se mostra adequada para reduzir o número de *setups* significativamente.

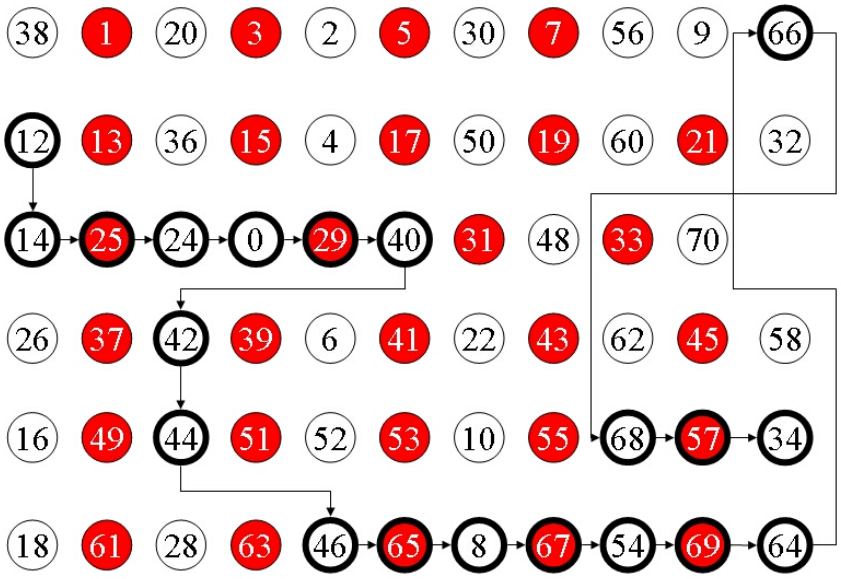


Figura 42: Grafo representando o problema-teste FT6.

A análise do comportamento do modelo frente a política de minimização do Tempo de *Setup* demonstra que o conceito de caminho crítico e o operador de vizinhança *N*5 utilizado na construção do modelo não são adequados para a minimização significativa desse tempo de produção. A geração de vizinhança do modelo é focada no conceito de caminho crítico, que possui pouca influência no Tempo de *Setup*. Foram realizados experimentos onde o operador de

geração de vizinhança $N5$ foi substituído pelo operador $N1$, que realiza trocas de duas operações em todo o caminho crítico, mas os resultados não proporcionaram melhoras significativas no comportamento da variável $setup$.

8.3 Variação dos Parâmetros do Modelo

Foram realizados experimentos onde os parâmetros da técnica ei-TSAB foram variados, nos quais o objetivo é verificar o impacto da variação desses parâmetros do método de busca no comportamento das variáveis de decisão de f_π.

Nesta análise o método ei-TSAB é utilizado inicialmente considerando os parâmetros descritos por Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 2002):

- $nbmax = 20.000$;
- LT assumindo os valores 8 e 15;
- $maxL$ assumindo os valores 5 e 7;
- $maxE$ assumindo os valores 5 e 8.

Foram feitos experimentos que refletem as três políticas de minimização, onde o peso utilizado para privilegiar cada variável é 1000, mantendo as outras duas variáveis com seus pesos fixos no valor dos pesos da SNT. Para fazer a análise das políticas de minimização, foi utilizada o problema-teste $ta1515_1$. As tabelas 13 e 14 mostram os resultados obtidos na execução do modelo computacional para os parâmetros acima definidos.

Tabela 13: Variação dos parâmetros da técnica utilizada:$nbmax = 20.000$,$LT = 8$, $maxL = 5$ e $maxE = 5$ frente as três políticas de minimização.

Política	*Makespan*	*Atraso*	*Trocas*
Makespan	1470	5737	2776
Atraso	1549	4972	2823
Setup	2832	18006	72294

Tabela 14: Variação dos parâmetros da técnica utilizada:$nbmax = 20.000$,$LT = 15$, $maxL = 7$ e $maxE = 8$ frente as três políticas de minimização.

Política	*Makespan*	*Atraso*	*Trocas*
Makespan	1490	5839	2766
Atraso	1565	4794	2784
Setup	2491	16481	2196

O aumento do valor dos parâmetros da técnica utilizada tem impacto positivo no comportamento das variáveis de decisão. Nas três políticas utilizadas, o valor da variável privilegiada sofreu redução tanto em relação à SNT quanto em relação ao melhor resultado encontrado nos experimentos com os parâmetros da técnica reduzidos.

Utilizando os parâmetros aumentados ($nbmax = 20.000$, $LT = 8$,$maxL = 5$ e $maxE = 5$), foram feitos experimentos onde cada uma das variáveis é minimizada, enquanto que a influência das outras á anulada. O objetivo desses experimentos é verificar qual o comportamento do modelo minimizando apenas uma variável por vez. Na tabela 15 são apresentados os resultados desses experimentos. Considera-se que a variável que está sendo minimizada tem o valor do peso igual a 1, enquanto que as outras tem ao seu peso atribuído o valor 0.

Tabela 15: Minimização das variáveis de F separadamente

Variável	*Makespan*	*Atraso*	*Trocas*
Makespan	1289	3732	2806
Atraso	1319	2733	2868
Setup	3012	22299	2324

Comparando os resultados da tabela 15 com os resultados da SNT, a consideração de apenas uma variável na busca proporciona uma redução de 18,4% para o *Makespan*, 58,1% para o *Atraso* e 11% para o *Setup*. Comparado com a Solução Inicial, as variáveis *Makespan*, *Atraso* e *Setup*, quando minimizadas separadamente, assumem valores 88%, 96,2% e 20,5% menores, respectivamente, para o problema-teste $ta1515_1$. O gráfico apresentado na figura 43 mostra o comportamento das variáveis de decidão considerando os experimentos com os parâmetros aumentados, a SNT e os resultados obtidos nas três políticas de minimização analisadas. Considera-se na figura: (1) os parâmetros $LT = 8$, $maxL = 5$ e $maxE = 5$, (2) $LT = 15$, $maxL = 7$ e $maxE = 8$, (3) variáveis minimizadas independentemente.

Nos experimentos realizados com as três políticas de minimização, nota-se que o modelo implementado privilegia a redução do tempo total de produção. O conceito de caminho crítico e o operador $N5$ são ferramentas eficientes para tratar o PEJS, reduzindo consideravelmente o número de vizinhos a serem visitados, propiciando uma convergência rápida. A redução da variável *Makespan* acarreta na redução da variável *Atraso*, dado que ela tem ligação com o tempo total de produção. A variável *Setup* não sofre variações significativas considerando as políticas de minimização, dado o conceito de caminho crítico.

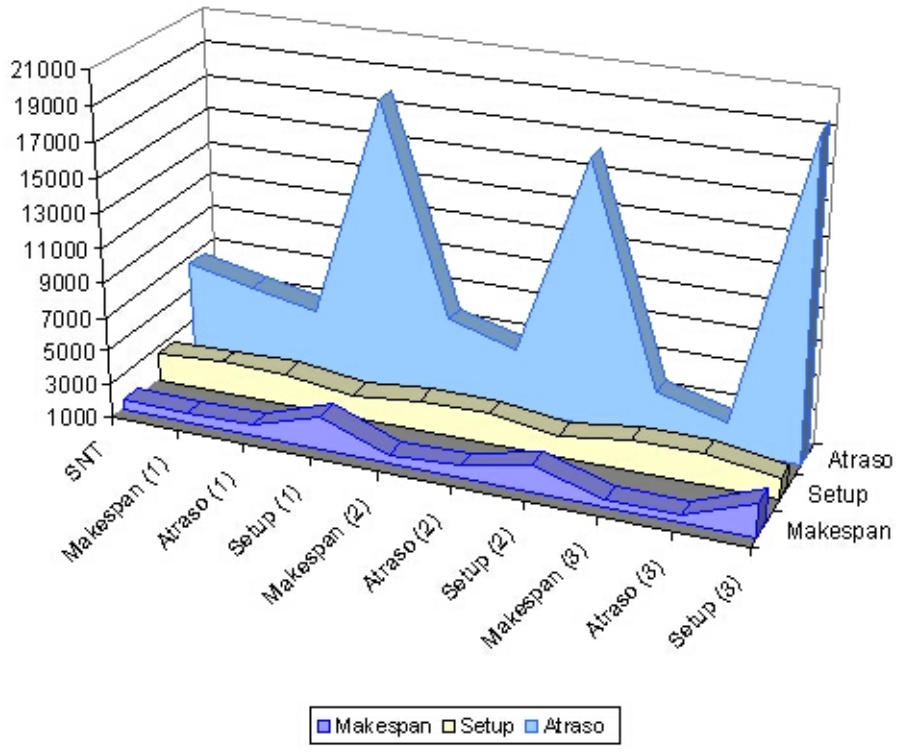


Figura 43: Gráfico comparativo entre políticas de minimização, SNT, considerando novos parâmetros.

9 CONCLUSÃO

O Problema de Escalonamento em um *Job Shop* é um problema clássico na área de Otimização Combinatória, surgindo de estudos realizados no ambiente produtivo. Em linhas gerais, esse problema considera um ambiente produtivo composto de partes a serem processadas em máquinas, dispostas em uma planta fabril onde o objetivo é minimizar o tempo total de produção dessas partes.

A complexidade computacional deste problema é alta, motivando o uso de heurísticas e meta-heurísticas na sua abordagem. Na revisão bibliográfica apresentada, pode-se observar que o uso de heurísticas como Algoritmos Genéticos, Têmpera Simulada e Busca Tabu propiciam bons resultados para o PEJS e suas variações. Os trabalhos mais recentes pesquisados focam o uso de métodos que combinam duas heurísticas, dispondo de estratégias de intensificação e diversificação.

Para o desenvolvimento deste trabalho foi considerada uma versão do PEJS com turnos de produção, com a necessidade de diferentes tipos de ferramentas para o processamento das partes e as datas de entrega dessas partes. As restrições consideradas nessa versão do PEJS fazem com que a dificuldade do problema aumente, porém deixam ele mais próximo do ambiente produtivo real.

O ambiente produtivo considerado é o Sistema de Manufatura Flexível, onde existe um conjunto de máquinas de propósito geral, um Sistema de Transporte e Armazenamento de Materiais e um Sistema Computacional para a gestão da produção. As máquinas são capazes de realizar qualquer operação e tem um *magazine* de capacidade limitada, inferior ao número total de ferramentas necessárias para o processamento de todas as partes. Uma vez que partes que necessitam de diferentes ferramentas estão em seqüência, trocas de ferramentas se fazem necessárias (*setup*).

Foi estudado o Problema de Seleção de Partes para modelar a troca de ferramentas entre os diferentes tipos de operações. Neste problema, as partes são agrupadas em Famílias de Operações de acordo com alguma similaridade de processo produtivo. Várias ferramentas de

Tecnologia de Grupo foram pesquisadas, optando-se pelo Algoritmo de Seleção de Partes, que utiliza conceitos de Análise de Agrupamentos. Foi implementada a versão proposta por Gómez (GÓMEZ, 1996) desse algoritmo, a qual considera o compartilhamento de ferramentas entre partes de diferentes famílias. Essa implementação permitiu que fosse gerada uma estrutura de dados que representa as Famílias de Operações, que facilita o cálculo do tempo de *setup* entre operações.

Considerando as restrições adicionais ao PEJS, foi gerada uma função objetivo f_π que quantifica os tempos de produção considerados. Essa função é composta por três variáveis de decisão: (*i*) Tempo Total de Produção, ou *Makespan*, (*ii*) Tempo de Atraso das partes e (*iii*) Tempo de *Setup*. Através de atribuição de pesos para essas variáveis, é possível realizar a gestão da importância delas.

Para gerar uma seqüência de operações nas máquinas do SMF que minimize o valor de f_π, foram estudados três trabalhos: o de Hertz e Widmer (HERTZ; WIDMER, 1996) e os de Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 1996) (NOWICKI; SMUTNICKI, 2002). O primeiro trabalho faz uma modelagem do *setup* como uma operação fictícia no grafo que representa o escalonamento. Hertz e Widmer utilizaram Busca Tabu para realizar um escalonamento que minimizasse o tempo total de produção, privilegiando a ocorrência de *setups* nos tempos ociosos da produção. O trabalho de Nowicki e Smutnicki trata o PEJS original utilizando quatro conceitos principais: um sistema de reconstituição de áreas já visitadas para diversificação da busca (*Back Jump Tracking*), um conjunto de operações que mais contribui para o *Makespan* (caminho crítico), um operador de vizinhança que considera trocas entre as primeiras e as últimas operações dos blocos de um caminho crítico (operador $N5$) e a geração de soluções através da consideração de atributos de soluções de elite existentes (*path relinking*). Esses conceitos foram implementados em uma Busca Tabu, sendo denominado esse método de *i*-TSAB.

Com base nestes trabalhos foi implementada uma versão da técnica *i*-TSAB (chamada de ei-TSAB) que considera as restrições adicionais do PEJS estudado. A busca foi modificada para considerar o valor de f_π como o fator a ser minimizado. O modelo de grafo de Nowicki e Smutniki foi adaptado para considerar as operações fictícias que representam os *setups*, os turnos de produção e o valor do atraso das operações de cada parte. Foi construída uma arquitetura para o modelo, composta de quatro módulos: (*i*) módulo de leitura de problemas-teste, (*ii*) módulo de geração das Famílias de Operações, (*iii*) módulo de geração da Solução Inicial e (*iv*) módulo de Otimização do Escalonamento.

O modelo computacional foi testado e validado utilizando problemas-teste de *benchmark* disponíveis na literatura, as quais possuem o resultado ótimo (ou menor limite inferior)

conhecido. Esses problemas-teste, no entanto, são compostas apenas dos tempos de processamento e das restrições de precedências das operações das partes, não contemplando os dados necessários para testes com as restrições de turnos, atraso e ferramental. Para que estes problemas-teste pudessem ser utilizadas, o módulo de leitura de problemas-teste do modelo gerava os dados relativos ao problema abordado. O modelo computacional implementado foi validado em três etapas: (*i*) validação da geração de Famílias de Operações, utilizando um problema-teste proposto por Tang e Denardo, utilizada por Gómez (GÓMEZ, 1996); (*ii*) validação Da variável *Setup*, comparando o ei-TSAB com resultados obtidos por Hertz e Widmer (HERTZ; WIDMER, 1996) e (*iii*) validação da variável *Makespan*, utilizando os problemas-teste FT6, FT10 e FT20 propostas por Fisher e Thompson (JAIN; MEERAN, 1999).

Dado a complexidade adicional que as restrições de *Atraso* e de Tempo de *Setup* introduzem no modelo, não foi possível a mesma eficiência computacional provida pela técnica de aceleração da busca, descrita por Nowicki e Smutnicki (NOWICKI; SMUTNICKI, 2002). Os testes com o modelo computacional implementado mostraram que a consideração das restrições adicionais ao PEJS tornam a convergência do modelo para um f_π mínimo mais lenta. Outro fator observado é que o número de operações fictícias aumenta o tamanho do problema-teste, o que eleva o tempo de execução do modelo.

Foi feita a análise do comportamento do modelo frente a definição de três políticas de: (*i*) minimização do *Makespan*, (*ii*) minimzação do *Atraso* e (*iii*) minimização do Tempo de *Setup*. Para verificar o quanto cada política minimiza cada uma das variáveis, foi definida uma solução não tendenciosa com a qual se fazem comparações. Essa solução é obtida pela aplicação da técnica implementada parametrizada com uma combinação de pesos que faz cada uma das variáveis de decisão ter a mesma contribuição para o valor de f_π. Os pesos da SNT propiciam uma melhora significativa no valor das variáveis de decisão se comparado com a Solução Inicial.

A política de minimização do tempo total de produção ocasiona uma sensível redução no valor assumido pelas variáveis *Makespan* e *Atraso*, em relação a SNT. Isso se deve ao fato de que o modelo implementado é fortemente baseado no conceito de caminho crítico, que define o valor do *Makespan*. A redução do tempo total de produção contribui para a redução do atraso das partes que possuem operações no caminho crítico, o que ocasiona uma redução no atraso total. O Tempo de *Setup* sofre acréscimo na implementação dessa política. Esse comportamento ocorre dado que a variável *Setup* é calculada pelo somatório do tempo de todas as paradas da produção para a troca de ferramentas. Como o operador de geração de vizinhança $N5$ realiza troca de operações dentro dos blocos do caminho crítico, somente os *setups* que ocorrerem entre

as operações deste é que poderão ser reduzidos.

A política de minimização do tempo de atraso das partes apresenta resultados inferiores se comparados com os obtidos pela SNT. Esse comportamento ocorre uma vez que o atraso é calculado com base no tempo de término das últimas operações de cada parte. Considerando o conceito de caminho crítico, as partes que podem ter seu atraso reduzido a cada iteração da busca, são aquelas cujas últimas operações fazem parte do caminho crítico. Um comportamento similar é observado na política de minimização do Tempo de *Setup*, onde a variável $Setup$ sofre uma sensível redução em relação a SNT, enquanto que as variáveis $Makespan$ e $Atraso$ sofrem acréscimos.

Foram realizados experimentos com o objetivo de verificar se o aumento dos parâmetros da técnica utilizada melhora o resultado das variáveis de decisão de f_π, seguindo as três políticas de minimização. Esse aumento teve impacto positivo, melhorando o comportamento das variáveis em relação a SNT. Utilizando esses parâmetros aumentados, fêz-se uma análise do comportamento do modelo quando minimiza uma variável de decisão por vez. Nestes últimos experimentos, nota-se que as três variáveis atingem o menor valor registrados nos experimentos com as políticas de minimização.

Nos experimentos realizados neste trabalho, nota-se que a técnica implementada tende a privilegiar a minimização da variável $Makespan$. Mesmo realizando as modificações para que a técnica i-TSAB considerasse as restrições adicionais ao PEJS, o conceito de caminho crítico e o operador $N5$ forçam a busca a reduzir o tempo total de produção. A variável $Atraso$ é beneficiada nesse processo, dado que o valor do atraso depende do tempo no qual as partes são terminadas. A variável $Setup$ sofre maior redução quando considerada unicamente no modelo, anulando a influência das outras duas variáveis. O fato das variáveis $Atraso$ e $Setup$ não dependerem exclusivamente do caminho crítico e sim de todo o escalonamento, enfraquece a eficiência da gestão dos tempos de produção através dos pesos dessas variáveis.

Por fim, pode-se dizer que o modelo proposto é uma abordagem para o tratamento do PEJS, considerando as restrições de turnos de produção, tempos de *setup* e atraso das partes, oferecendo resultados interessantes quando as três variáveis são consideradas separadamente. A implementação da técnica i-TSAB foi trabalhosa e os conceitos de caminho crítico e o operador de geração de vizinhança escolhidos prejudicam o desempenho da busca quando as três variáveis são consideradas simultaneamente.

Como trabalhos futuros, sugere-se a pesquisa de um conceito de caminho crítico que contemple as restrições de *setup* e atraso. A utilização desse conceito no problema original permite que o espaço de busca seja restrito a movimentos que ocasionam melhora significativa

na busca. Assim sendo, é interessante que esse conceito seja melhor estudado para ser estendido a problemas multi-objetivo.

A função objetivo que permite gerenciar os tempos de produção abordados neste trabalho pode ser modificada para trabalhar com custos de produção. Para tanto, sugere-se que sejam feitas modificações no cálculo do valor de f_π a fim de considerar custos de atraso, custos de estocagem de produtos, custos de máquinas tempo ocioso, entre outros.

Sugere-se ainda um estudo da estrutura do problema abordado neste trabalho, a fim de inserir conhecimento adicional nos métodos heurísticos utilizados atualmente. Esse conhecimento seria empregado na geração eficiente de vizinhanças.

O estado da arte mostra que as técnicas de Busca Tabu, Têmpera Simulada e Algoritmos Genéticos retornam os melhores resultados para o PEJS. Sugere-se como alternativa o uso da técnica *Scatter Search* como abordagem para o problema estudado.

REFERÊNCIAS

AGNETIS, A. et al. Joint job/tool scheduling in as flexible manufacturing cell with no-onboard tool magazine. *Computer Integrated Manufacturing Systems*, v. 10, n. 1, p. 61–68, 1997.

ARMENTANO, V. A.; SCRICH, C. R. Tabu search for minimizing total tardiness in a job shop. *International Journal of Production Economics*, v. 63, n. 2, p. 131–140, janeiro 2000. Disponível em: <www.sciencedirect.com>.

AZIZI, N.; ZOLFAGHARI, S. Adaptative temperature control for simulated annealing: a comparative study. *Computer & Operations Research*, v. 31, p. 2439–2451, 2004.

BALAS, E.; VAZACOMPOLOUS, A. *Guided local search with shift bottleneck for job shop scheduling*. [S.l.], 1998.

BEDWORTH, D. D.; HENDERSON, M. R.; WOLFE, P. M. *Computer-integrated desing and manufacturing*. [S.l.]: McGraw-Hill, 1991. 653 p. ISBN 0-07-004204-7.

BŁAŻEWICZ, J.; DOMSCHKE, W.; PESCH, E. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operations Research*, v. 93, p. 1–33, 1996.

CAVORY, G.; DUPAS, R.; GONCALVES, G. A genetic approach to solving the problem of cyclic job shop scheduilng with linear constraints. *European Journal of Operations Research*, v. 161, p. 73–85, 2005.

CHENG, R.; GEN, M.; TSUJIMURA, Y. A tutorial survey of job shop scheduling problems using genetic algorithms - I. Representation. *Computers & Industrial Engeneering*, v. 30, n. 4, p. 983–997, 1996.

CHENG, R.; GEN, M.; TSUJIMURA, Y. A tutorial survey of job shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies. *Computers & Industrial Engeneering*, v. 36, p. 343–364, 1999.

CHENG, T. C. E.; J.JIANG. Job shop scheduling for missed due-date performance. *Computers & Industrial Engeneering*, v. 34, n. 2, p. 297–307, 1998.

CORMEN, T. H. *Introduction to algorithms*. [S.l.]: MIT, 2001. 1180 p. ISBN 0-262-03293-7.

GHEDJATI, F. Genetic Algorithms for the job shop scheduling problem with unrelated parallel constraints: heuristic mixing method and machine precedence. *Computers & Industrial Engeneering*, v. 37, p. 39–42, 1999.

GHRAYEB, O. A. A bi-criteria optimization: minimizing the integral value and spread of the fuzzy makespan of job shop scheduling problems. *Applied Soft Computing*, v. 2, n. 3, p. 197–210, 2003.

GLOVER, F. Tabu Search: Part I. *ORSA, Journal of Computing*, v. 1, n. 3, p. 191–207, 1989.

GLOVER, F.; LAGUNA, M. *Tabu Search*. [S.l.]: Kluwer Academic Publishers, 1997. 382 p. ISBN 0-7923-9965-X.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. [S.l.]: Addison-Wesley, 1989. ISBN 0-201-15767-5.

GÓMEZ, A. T. *Modelo para o seqüenciamento de partes e ferramentas em um sistema de manufatura flexível com restrições às datas de vencimento e à capacidade do magazine*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos, São Paulo, Brasil, 1996.

GONÇALVES, J. F.; MENDES, J. J. de Magalhães; RESENDE, M. G. C. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, v. 167, p. 77–95, 2005.

GONÇALVES, J. F.; RESENDE, M. G. C. An evolutionary algorithm for manufacturing cell formation. *Computer & Industrial Engineering*, Elsevier, v. 47, p. 247–273, 2004.

GROOVER, M. P. *Automation, production systems and computer-integrated manufacturing*. Second. [S.l.]: Prentice-Hall, 2001. 856 p. ISBN 0-13-088978-4.

GUNASEKARAN, A.; MARTIKAINEN, T.; YLI-OLLI, P. Flexible manufacturing systems: an investigation for research and applications. *European Journal of Operational Research*, v. 66, p. 1–26, 1993.

HAAYMAKERS, W. H. M.; HOOGEVEEN, J. A. Scheduling multipurpose batch process industries with no-wait restrictions with simulated annealing. *European Journal of Operational Research*, v. 126, p. 131–151, 2000.

HE, Z.; YANG, T.; TIGER, A. An exchange heuristic imbedded with simulated annealing for due-dates job-shop scheduling. *European Journal of Operations Research*, v. 91, p. 99–117, 1996.

HERTZ, A.; WIDMER, M. An improved tabu search approach for solving the job shop scheduling problem with tooling constraints. *Discrete Applied Mathematics*, v. 65, p. 319–345, 1996.

HILLIER, F. S.; LIEBERMAN, G. J. *Introdução à Pesquisa Operacional*. [S.l.]: Campus, 1988. 805 p. ISBN 85-7001-143-1.

HOLTHAUS, O.; RAJENDRAN, C. Efficient dispatching rules for a scheduling in a job shop. *International Journal of Production Economics*, v. 48, p. 87–105, 1997.

HWANG, S. S.; SHOGAN, A. W. *Modeling and Solution of an FMS Part Selection Problem*. 1987. 56 p.

IVES, P.; LAMBRECHT, M. Extending shift bottleneck procedure to real-life applications. *European Journal of Operational Research*, v. 90, n. 2, p. 252–268, abril 1996. Disponível em: <www.sciencedirect.com>.

JAIN, A. S.; MEERAN, S. Deterministic job shop scheduling: Past, present and future. *European Journal of Operations Research*, v. 113, p. 390–434, 1999.

JAYAMOHAN, M. S.; REJENDRAN, C. Development and analysis of cost-based dispatching rules for job shop scheduling. *European Journal of Operations Research*, v. 157, p. 307–321, 2004.

JHA, N. K. *Handbook of Flexible Manufacturing Systems*. [S.l.]: Academic Press, 1991. 328 p. ISBN 0-12-385310-9.

JOSHI, S. B.; SMITH, J. S. *Computer control of flexible manufacturing systems : research and development*. [S.l.]: Chapman & Hall, 1994. 478 p. ISBN 0-412-56200-6.

KAIGHOBADI, M.; VENKATESH, K. Flexible manufacturing Systems: an Overview. *International Journal of Operations & Prodution Management*, v. 14, n. 4, p. 16–49, 1993.

KHER, H. V. Examination of work assignment and dispatching rules for managing vital customer priorities in dual resource constrained job shop environments. *Computer & Operations Research*, v. 27, p. 525–537, 2000.

KIM, Y. K.; PARK, K.; KO, J. A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *European Journal of Operational Research*, v. 30, p. 1151–1171, 2003.

KIS, T. Job shop scheduling problem with processing alternatives. *European Journal of Operational Research*, v. 151, p. 307–332, 2003.

KOLONKO, M. Some new results on simulated annealing applied to the job shop scheculing problem. *European Journal of Operations Research*, v. 113, p. 123–136, 1999.

KUMAR, N. H.; SRINIVASAN, G. A genetic algorithm for job shop scheduling - A case study. *Computers in Industry*, v. 31, p. 155–160, 1996.

KURODA, M.; WANG, Z. Fuzzy job shop scheduling. *International Journal of Production Economics*, v. 44, p. 45–51, 1996.

KUSIAK, A. Application of operational research models and techniques in flexible manufacturing systems. *European Journal of Operational Research*, v. 24, p. 336–345, 1986.

KUSIAK, A. *Intelligent Design and Manufacturing*. [S.l.]: John Wiley & Sons, 1992. 753 p. ISBN 0-471-53473-0.

KUSIAK, A.; CHOW, W. S. *Efficient Solving of the Group Techonology Problem*. 1987. 22 p.

LAARHOVEN, P. J. M. V.; AARTS, E. H. L.; LENSTRA, J. K. Job Shop Scheduling Problem by Simulated Annealing. *Operations Research*, v. 40, n. 1, p. 113–125, janeiro-fevereiro 1992.

LI, H. et al. A production rescheduling expert simulation system. *European Journal of Operational Research*, v. 124, p. 283–293, 2000.

LORINI, F. J. *Tecnologia de Grupo e Organização da Manufatura*. [S.l.]: Universidade Federal de Santa Catarina, 1993. 105 p.

MAMALIS, A. G.; MALAGARDIS, I. Determination of due dates in job shop scheduling by simulated annealing. *Computer Integrated Manufacturing Systems*, v. 9, p. 2, 1996.

MASCIS, A.; PACCIARELLI, D. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, v. 143, p. 498–517, 2002.

MATTFELD, D. C.; BIERWIRTH, C. An efficient genetic algorithm for job shop scheduling with tardiness objectives. *European Journal of Operational Research*, v. 155, p. 616–630, 2004.

MOHANASUDARAM, K. M. et al. Scheduling rules for dynamic shops that manufacture multi-leve shops. *Computers & Industrial Engeneering*, v. 44, p. 119–131, 2002.

MÖNCH, L.; BRIESSEL, R. A distributed shifting bottleneck heuristic for complex job shops. *Computers & Industrial Engeneering*, v. 49, n. 3, p. 363–380, novembro 2005. Disponível em: <www.sciencedirect.com>.

MOSIER, C. T.; YELLE, J.; WALKER, G. Survey of Similarity Coefficient Based Methods as Applied to the Group Techonology Configuration Problem. *Omega: International Journal of Sciencitif Management*, Elsevier Science, v. 25, p. 65–79, 1997.

NOWICKI, E.; SMUTNICKI, C. A fast tabu search algorithm for the permutarion flow-shop problem. *European Journal of Operational Research*, Elsevier, v. 91, p. 160–175, 1996.

NOWICKI, E.; SMUTNICKI, C. *Some new tools to solve the job shop problem*. [S.l.], outubro 2002.

OSMAN, L. H.; LAPORTE, G. Metaheuristics: A bibliography. *Annals of Operations Research*, n. 63, p. 513–628, 1996.

PARK, B. J.; CHOI, H. R.; KIM, H. S. A hybrid genetic algorithm for the job shop scheduling problems. *Computers & Industrial Engeneering*, v. 45, p. 597–613, 2003.

PEZZELLA, F.; MERELLI, E. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, v. 120, n. 2, p. 297–310, janeiro 2000. Disponível em: <www.sciencedirect.com>.

PINEDO, M. *Scheduling: Theory, algorithms and Systems*. Englewood Cliffs: Prentice-Hall, 1995. 378 p. ISBN 0-13-706757-7.

RAYWARD-SMITH, V. J. et al. *Modern heuristic search methods*. Chichester: John Wiley & Sons, 1996. ISBN: 0-471-96280-5.

RODRIGUES, A. G.; GóMEZ, A. T. Production time minimization strategies: a Tabu Search approach. In: *Annals of the 2nd ICINCO - International Conference on Informatics in Control, Automation and Robotics*. [S.l.: s.n.], 2005.

SAKAWA, M.; KUBOTA, R. Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms. *European Journal of Operational Research*, v. 120, p. 393–407, 2000.

SAKAWA, M.; MORI, T. An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate. *Computers & Industrial Engeneering*, v. 36, p. 325–341, 1999.

SATAKE, T. et al. Simulated annealing approach for minimizing makespan of ght general job shop. *International Journal of Production Economics*, v. 60-61, p. 515–522, 1999.

SHOMAN, M. A. et al. The iteractive process between some dispatching mecanisms and interrupted machine centers in FMSs. *Journal of Materials Processing Technologies*, v. 107, p. 466–477, 2000.

SINGER, M. Decomposition methods for large job shops. *Computer & Operations Research*, v. 28, n. 3, p. 193–207, março 2001. Disponível em: <www.sciencedirect.com>.

SONG, J.-S.; LEE, T.-E. A tabu search procedure for periodc job shop scheduling. *Computers & Industrial Engeneering*, v. 30, n. 3, p. 433–447, 1996.

STEINHÖFEL, A. A. K.; WONG, C. K. Fast parallel heuristics for the job shop scheduilng problem. *Computer & Operations Research*, v. 29, p. 151–169, 2002.

STEINHÖFEL, K.; ALBRECHT, A.; WONG, C. K. Two simulated annealing-based heuristics for the job shop scheduling problem. *European Journal of Operations Research*, v. 118, p. 524–548, 1999.

SU, L.-H.; CHANG, P.-C. A heuristic for scheduling general job shops to minimize maximum lateness. *Mathematical and Computing Modelling*, v. 27, n. 1, p. 1–15, 1998.

TAILLARD Éric. *Instâncias de Problemas de Escalonamento em Job Shop*. 2006. Disponível em: <http://ina2.eivd.ch/collaborateurs/etd/problemes.dir/ordonnancement.dir/ordonnancement.html>.

TAILLARD Éric D. Parallel Taboo Search Techniques for Job Shop Scheduling Problem. *ORSA Journal of Computing*, v. 6, n. 2, p. 108–117, 1994.

TANEV, I. T.; UOZOMI, T.; MOROTOME, Y. Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provider approach. *Applied Soft Computing*, v. 5, p. 87–100, 2004.

THIAGARAJAN, S.; RAJENDRAN, C. Scheduling in dynamic assembly job-shops to minimize the sum of weighted earliness, weighted tardiness and weighted flowtime of jobs. *Computers & Industrial Engeneering*, v. 49, p. 463–503, 2005.

TORRES, P.; LOPEZ, P. On Not-First/Not-Last conditions in disjunctive scheduling. *European Journal of Operational Research*, v. 127, n. 2, p. 332–343, 2000.

VALLS, V.; PEREZ, M. A.; QUINTANILLA, M. S. A tabu search approach to machine shceduling. *European Journal of Operations Research*, v. 106, p. 277–300, 1998.

VARELA, R. et al. A knowledge-based evolutionary strategy for scheduling problems with bottlenecks. *European Journal of Operational Research*, v. 145, p. 57–71, 2003.

VERAL, E. A. Computer simulation of a due-date setting in a multi-machine job shops. *Computers & Industrial Engeneering*, v. 41, p. 77–94, 2001.

VERHOEVEN, M. G. A. A tabu search for resource-constrained scheduling. *European Journal of Operations Research*, v. 106, p. 266–276, 1998.

VIANA, G. V. *Meta-heurísiticas e Programação Paralela em Otimização Combinatória*. [S.l.]: UFC Edições, 1998. ISBN 85-7282-039-6.

WANG, L.; ZHENG, D.-Z. An effective hybrid optimization strategy for job-shop scheduling problems. *Computer & Operations Research*, v. 28, p. 585–596, 2001.

WANG, T.-Y.; WU, K.-B. A parameter set design procedure for the simulated annealing algorithm under the computational time constraint. *Computer & Operations Research*, v. 26, p. 665–678, 1999.

WATANABE, M.; IDA, K.; GEN, M. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. *Computers & Industrial Engeneering*, v. 48, p. 743–752, 2005.

WATSON, J.-P.; HOWE, A. E.; WHITLEY, L. D. Deconstructing Nowicki and Smutnicki's *i*-TSAB tabu search algorithm for the job-shop scheduling problem. *Computer & Operations Research*, v. 33, p. 2623–2644, 2006.

WENQI, H.; AIHUA, Y. An improved shifting bottleneck procedure for the job shop scheduling problem. *Computer & Operations Research*, v. 31, n. 12, p. 2093–2110, outubro 2004. Disponível em: <www.sciencedirect.com>.

XIA, W.; WU, Z. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engeneering*, v. 48, p. 409–425, 2005.

YIN, Y.; YASUDA, K. Similarity coefficient methods applied to the cell formation problem: a comparative investigation. *Computers and Industrial Engineering*, v. 48, p. 471–489, 2005.

YU, H.; LIANG, W. Neural network and genetic algorithm-based hybrid approach to expanded job-shop scheduling. *Computers & Industrial Engeneering*, v. 39, p. 337–356, 2001.

YUN, Y. S. Genetic algorithm with fuzzy logic controller for preemptive and non-preemptive job-shop scheduling problems. *Computers & Industrial Engeneering*, v. 43, p. 623–644, 2002.

ZHANG, C. Y. et al. A tabu search algorithm with a new neighborhood estructure for the job shop scheduling problem. Article in press. fevereiro 2006. Disponível em: <www.sciencedirect.com>.

ZHANG, C. Y. et al. A very fast TS/SA algorithm for the job shop scheduling problem. Article in press. abril 2006. Disponível em: <www.sciencedirect.com>.

ZHOU, H.; FENG, Y.; HAN, L. The hybrid heuristic genetic algorithm for job shop scheduling. *Computers & Industrial Engeneering*, v. 40, p. 191–200, 2001.

APÊNDICE A – POLÍTICA DE MINIMIZAÇÃO DO *MAKESPAN*

São apresentadas aqui as tabelas referentes aos resultados obtidos na análise do comportamento da variável *Makespan*. Os resultados são referentes aos problemas-teste $ta1515_2$, $ta3020_1$, $ta3020_2$, $ta5015_1$ e $ta5015_2$, sendo apresentados em unidades de tempo.

Tabela 16: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta1515_2$, considerando a política de minimização do *Makespan*.

Valor de W_1	*Makespan*	*Atraso*	*Trocas*
Inicial	10216	78214	2825
SNT	1563	10640	2576
100	1497	10453	2667
500	1512	10463	2715
1000	1512	10329	2729

Tabela 17: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta3020_1$, considerando a política de minimização do *Makespan*.

Valor de W_1	*Makespan*	*Atraso*	*Trocas*
Inicial	27250	393643	7834
SNT	5957	112293	7466
100	3286	58757	7534
500	2924	51997	7574
1000	3308	57328	7491

Tabela 18: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta3020_2$, considerando a política de minimização do *Makespan*.

Valor de W_1	*Makespan*	*Atraso*	*Trocas*
inicial	27990	390054	7970
SNT	4466	72520	7624
100	2841	39383	7796
500	2842	38670	7776
1000	2824	37163	7702

Tabela 19: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta5015_1$, considerando a política de minimização do *Makespan*.

Valor de W_1	*Makespan*	*Atraso*	*Trocas*
inicial	33637	777600	10047
SNT	5061	129275	9442
100	5141	131207	9727
500	3717	85832	9787
1000	3938	91201	9773

Tabela 20: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta5015_2$, considerando a política de minimização do *Makespan*.

Valor de W_1	*Makespan*	*Atraso*	*Trocas*
inicial	31586	734498	9948
SNT	5226	145025	9459
100	4882	135168	9732
500	4712	131488	9787
1000	4649	130706	9745

APÊNDICE B – POLÍTICA DE MINIMIZAÇÃO DO ATRASO

São apresentadas aqui as tabelas referentes aos resultados obtidos na análise do comportamento da variável *Atraso*. Os resultados são referentes aos problemas-testes $ta1515_2$, $ta3020_1$, $ta3020_2$, $ta5015_1$ e $ta5015_2$, sendo apresentados em unidades de tempo.

Tabela 21: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta1515_2$, considerando a política de minimização do Atraso.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
Inicial	10216	78214	2825
SNT	1563	10640	2576
100	1530	9863	2721
500	1532	9825	2697
1000	1485	9812	2721

Tabela 22: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta3020_1$, considerando a política de minimização do Atraso.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
Inicial	27250	393643	7834
SNT	5957	112293	7466
100	4962	96709	7783
500	4284	78596	7714
1000	4798	88332	7809

Tabela 23: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta3020_2$, considerando a política de minimização do Atraso.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
Inicial	27990	390054	7970
SNT	4466	72520	7624
100	4227	66937	7859
500	4211	67839	7848
1000	4950	82484	7865

Tabela 24: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta5015_1$, considerando a política de minimização do Atraso.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
Inicial	33637	777600	10047
SNT	5061	129275	9442
100	7369	214453	10026
500	14734	443638	9835
1000	7736	227443	10034

Tabela 25: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta5015_2$, considerando a política de minimização do Atraso.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
Inicial	31586	734498	9948
SNT	5226	145025	9459
100	6812	196865	9904
500	9475	294995	10006
1000	6976	218807	9797

APÊNDICE C – POLÍTICA DE MINIMIZAÇÃO DO *SETUP*

São apresentadas aqui as tabelas referentes aos resultados obtidos na análise do comportamento da variável *Setup*. Os resultados são referentes aos problemas-teste $ta1515_2$, $ta3020_1$, $ta3020_2$, $ta5015_1$ e $ta5015_2$, sendo apresentados em unidades de tempo.

Tabela 26: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta1515_2$, considerando a política de minimização do Tempo de *Setup*.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
Inicial	10216	78214	2825
SNT	1563	10640	2576
100	1938	14899	2365
500	2274	18654	2309
1000	2217	19824	2305

Tabela 27: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta3020_1$, considerando a política de minimização do Tempo de *Setup*.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
Inicial	27250	393643	7834
SNT	5957	112293	7466
100	4216	75654	7249
500	5539	80048	6939
1000	7391	112440	7124

Tabela 28: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta3020_2$, considerando a política de minimização do Tempo de *Setup*.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
Inicial	27990	390054	7970
SNT	4466	72520	7624
100	4775	78017	7327
500	4904	64073	6962
1000	7281	88125	7201

Tabela 29: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta5015_1$, considerando a política de minimização do Tempo de *Setup*.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
Inicial	33637	777600	10047
SNT	5061	129275	9442
100	8226	241431	9416
500	6459	165721	8788
1000	7302	160910	8924

Tabela 30: Valores das variáveis de decisão de f_π obtidos pelo método ei-TSAB para o problema-teste $ta1515_2$, considerando a política de minimização do Tempo de *Setup*.

Valor de W_3	*Makespan*	*Atraso*	*Trocas*
Inicial	31586	734498	9948
SNT	5226	145025	9459
100	6989	203239	9506
500	6432	165261	8832
1000	7842	184940	8846