UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO

INSTITUTO POLITÉCNICO – IPRJ

Tatiana Balbi Fraga

Desenvolvimento de uma ferramenta computacional para a programação da produção de empresas do setor de confecções do município de Nova Friburgo

Nova Friburgo

2006

UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO

INSTITUTO POLITÉCNICO – IPRJ

Tatiana Balbi Fraga

Desenvolvimento de uma ferramenta computacional para a programação da produção de empresas do setor de confecções do município de Nova Friburgo

Dissertação apresentada ao Programa de Pós–Graduação em Modelagem Computacional da Universidade do Estado do Rio de Janeiro para obtenção do Título de Mestre. Área de concentração: Matemática aplicada e Computação científica

Orientador: Antônio José da Silva Neto
Co-Orientador: Francisco José da Cunha Pires Soeiro

Nova Friburgo

2006

CATALOGAÇÃO NA FONTE
UERJ/REDE SIRIUS/BIBLIOTECA CTC/E

F811 Fraga, Tatiana Balbi.
Desenvolvimento de uma ferramenta computacional para a programação da produção de empresas do setor de confecções do município de Nova Friburgo / Tatiana Balbi Fraga. - 2006.
95 f.

Orientador: Antônio José da Silva Neto
Co-Orientador: Francisco José da Cunha Pires Soeiro
Dissertação (mestrado) - Universidade do Estado do Rio de Janeiro, Instituto Politécnico.

1. Planejamento da produção – Modelos matemáticos -Teses. 2. Processos de fabricação – Modelos matemáticos - Teses . 3. Otimização matemática - Teses. 4. Vestuário – Indústria – Nova Friburgo – Teses. I. Silva Neto, Antônio José da. II. Soeiro, Francisco José da Cunha Pires. III. Universidade do Estado do Rio de Janeiro. Instituto Politécnico. IV. Título.

CDU 65.012.122:519.863

UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO

INSTITUTO POLITÉCNICO – IPRJ

Tatiana Balbi Fraga

Desenvolvimento de uma ferramenta computacional para a programação da produção de empresas do setor de confecções do município de Nova Friburgo

Dissertação apresentada ao Programa de Pós–Graduação em Modelagem Computacional da Universidade do Estado do Rio de Janeiro para obtenção do Título de Mestre. Área de concentração: Matemática aplicada e Computação científica

Aprovada em 15 de fevereiro de 2006

Banca examinadora:

_____

Prof. Carlos Alberto Fialho Thompson Leite, Ph. D.
Universidade do Estado do Rio de Janeiro. Instituto Politécnico

_____

Prof. Juarez Assumpção Muylaert Filho, Ph. D.
Universidade do Estado do Rio de Janeiro. Instituto Politécnico

_____

Prof. Stephan Stephany
Instituto Nacional de Pesquisas Espaciais

Nova Friburgo

2006

Dedico esta dissertação a minha mãe,
que sempre esteve ao meu lado,
dando o máximo de si para que eu
pudesse buscar e encontrar o meu
próprio caminho.

AGRADECIMENTOS

Agradeço a todos que contribuíram para o desenvolvimento deste trabalho:

• A minha mãe pelo apoio, carinho, e pelas broncas que muitas vezes foram necessárias.

• Ao meu orientador Prof. Antônio José da Silva Neto que acreditou em mim e esteve ao meu lado com muita paciência e atenção, sendo um exemplo de dedicação.

• Ao meu co-orientador Prof. Francisco Soeiro cuja colaboração foi imprescindível.

• Ao Prof. João Flávio Vasconcellos que esteve sempre presente nos momentos em que mais precisei de sua ajuda, um agradecimento especial.

• À CAPES pela bolsa de estudos.

• Às empresas que contribuíram para o desenvolvimento e teste da ferramenta computacional apresentada nesta dissertação.

RESUMO

O problema de seqüenciamento da produção vem sendo estudado desde o início da década de 50 do século passado e tem recebido nestes últimos cinqüenta anos uma considerável atenção de pesquisadores de todo o mundo. Como resultado atualmente encontra-se disponível uma gama de métodos de otimização e aproximação voltados para solução deste tipo de problema, sendo que a aplicação destes métodos mostra-se limitada à solução de problemas padrões de seqüenciamento, os quais consideram um conjunto de simplificações que os distanciam dos problemas ocorrentes nos ambientes reais de produção. Nesta dissertação o problema de seqüenciamento da produção sob análise trata-se especificamente do problema ocorrente nas micro e pequenas empresas do setor de confecções situadas no município de Nova Friburgo, onde foi constatado que quase não há um planejamento prévio da produção e quando o mesmo ocorre é feito com base somente em informações empíricas sem a aplicação de nenhuma metodologia e sem o auxílio de qualquer ferramenta computacional. Tal falta de planejamento resulta em um mau aproveitamento dos recursos de produção e impede que a empresa possa produzir em maior escala, o que se mostra necessário já que usualmente a demanda supera a capacidade produtiva da maioria das empresas do setor de confecções, principalmente em se tratando do sub-setor de moda íntima o qual abrange a maioria das empresas do município de Nova Friburgo. Visando melhorar o potencial competitivo destas empresas, esta dissertação se propõe a modelar matematicamente o seu processo de produção e desenvolver uma ferramenta computacional para a programação da produção baseada no método Tabu Search.

Palavras-Chave: Programação da produção, Setor de confecções, Tabu Search.

ABSTRACT

The manufacturing scheduling problem has been investigated since the 50's of the past century, and has received in the last 50 years a lot of attention from researchers around the world. As a result of such research efforts a lot of approximation and optimization methods are now available for the solution of such problems. Nonetheless, the application of these methods has been limited to standard problems of scheduling which considers a member of simplifications that do not correspond to the practical situations found in real production sets. In the present dissertation the manufacturing scheduling problem is devoted to real small and companies of productions sector of Nova Friburgo, for which has been observed that there is almost no prior production planning made, and when it is performed it is based only on empirical information without the application of a methodology or the aid of a computational tool. Such lack of planning results in a poor use of the production resources and prevents the company to produce in a larger scale, which is necessary because usually the demand is larger than the production capability of the majority of the companies of productions sector, manly in the sub-sector of underwear which corresponds to the majority of the companies of Nova Friburgo. Seeking to enhance the competitive edge of such companies the present dissertation has the purpose of modeling the production process and develop a computational tool for the production scheduling based on the Tabu Search method.

Key words: production scheduling, productions sector, Tabu Search.

SUMÁRIO

CAPÍTULO 1

1 INTRODUÇÃO

O setor de confecções de Nova Friburgo, cidade localizada na região serrana do estado do Rio de Janeiro, reúne cerca de 700 empresas (sendo aproximadamente 500 formais e 200 informais) e gera cerca de 20.000 empregos diretos, constituindo um importante papel na economia do município, com especial destaque para a produção de lingerie, que atende a 25% do mercado nacional e faz de Nova Friburgo o maior pólo de moda íntima do país [SEBRAE/RJ]. As empresas deste setor são, em sua maioria, micro e pequenas empresas e desfrutam, portanto, da principal vantagem competitiva associada às empresas deste porte que é a flexibilidade de produção, ou seja, a capacidade de se produzir simultaneamente uma grande variedade de produtos, e de deslocar-se para linhas inéditas assim que estas são exigidas pelo mercado. Contudo, nota-se que tal vantagem competitiva não é adequadamente explorada considerando-se a forma precária com a qual é realizada a programação da produção, que no caso em análise está diretamente relacionada aos processos de costura e acabamento.

Os problemas de programação da produção, também conhecidos como problemas de sequenciamento ou *scheduling problems*, vêm sendo estudados desde 1954, quando JOHNSON desenvolveu um algoritmo de otimização para um problema do tipo *flow shop* que considerava apenas duas máquinas [JAIN e MEERAN, 1999]. Posteriormente este problema ganhou novas formas, dando origem a uma série de “problemas padrões de sequenciamento” com diferentes níveis de complexidade, para os quais foram desenvolvidas e testadas diferentes metodologias. Atualmente encontra-se na literatura pelo menos três modelagens

diferentes para problemas desta natureza referentes aos casos apresentados como problemas de programação do tipo: (i) *flow shop*; (ii) *job shop*; e (iii) FMS (*Flexible Manufacturing System*).

1.1. PROBLEMAS DE PROGRAMAÇÃO DO TIPO *FLOW SHOP*

Os problemas do tipo *flow shop* aparecem na literatura como a forma mais simples na qual se apresentam os problemas de programação da produção e podem ser descritos como o sequenciamento de um conjunto de tarefas que devem ser processadas em um conjunto de máquinas distintas respeitando-se uma dada ordem de produção que deve ser a mesma para todas as tarefas. Segundo ONWOBOLU e DAVENDRA [2006], neste tipo de problema cada máquina só poderá processar uma única tarefa por vez e é considerado que inicialmente todas as tarefas estão disponíveis para processamento sendo que, no decorrer do processo, a execução de cada uma delas em cada máquina deve ser finalizada antes que a mesma seja iniciada na próxima máquina. Os autores indicam que estes casos acontecem geralmente em sistemas de manufatura, onde as tarefas são transferidas de máquina para máquina através de algum tipo de sistema automático de transporte, contudo há uma série de simplificações associadas ao *flow shop* que fazem com que tal modelo não retrate de forma adequada os ambientes reais de produção como, por exemplo, a desconsideração da relação de dependência que existe entre os tempos necessários para *setup* e transporte e a seqüência de processamento das diferentes tarefas em cada máquina. GUPTA e STAFFORD [2006] apresentam os problemas do tipo *flow shop* de uma forma mais generalizada comparando o fluxo de tarefas de um *flow shop* ao que ocorre em uma linha de produção. Segundo os autores ambos os casos apresentam um fluxo unidirecional, mas existe uma série de diferenças que devem ser consideradas, tais como:

a) um *flow shop* permite o processamento de uma variedade de tarefas, ao contrário de uma linha de produção que, geralmente, processa apenas um tipo de produto padrão;
b) no caso de um *flow shop* as tarefas não precisam ser processadas em todas as máquinas, ou seja, uma tarefa pode suprimir determinadas operações quando estas não se mostram necessárias, já no caso de uma linha de produção, todas as tarefas devem mover de uma estação para outra sem pular nenhuma estação de trabalho;
c) em um *flow shop* cada máquina é independente das outras, todavia nas operações de uma linha de produção cada estação de trabalho depende da anterior;
d) em um *flow shop* toda tarefa tem seu próprio tempo de processamento em cada máquina, enquanto que, em uma linha de produção todas as unidades de um produto têm um tempo padrão em todas as estações de trabalho.

Neste mesmo artigo os autores também apresentam 21 definições, conforme anteriormente sugeridas pelo próprio GUPTA, que descrevem de maneira bem abrangente o modelo do problema de sequenciamento original descrito por JOHNSON em 1954 (vide Tabela 1.1). Tais definições mostram as limitações de um problema do tipo *flow shop* no que diz respeito à representação de ambientes reais de produção.

Tabela 1.1– Definições apresentadas por GUPTA e STAFFORD [2006] para o problema de seqüenciamento original descrito por JOHNSON em 1954.

Definições relativas às tarefas	
J1	Cada tarefa está livre para ser executada no início do período de programação.
J2	Cada tarefa deve ter o seu próprio momento de execução que é fixo e não está sujeito a modificações
J3	Cada tarefa é independente das outras.
J4	Cada tarefa consiste em um conjunto de operações específicas, sendo que cada uma destas deve ser executada por apenas uma máquina.
J5	Cada tarefa possui uma ordem predefinida de processamento que é a mesma para todas as tarefas consideradas.

Continuação da Tabela 1.1	
J6	Cada tarefa requer um tempo de processamento entre as várias máquinas finito e conhecido. Este tempo de processamento inclui tempo de transporte e de *setup*, sendo que estes são independentes das tarefas precedentes e subseqüentes.
J7	Cada tarefa é processada apenas uma única vez em cada máquina.
J8	Cada tarefa deve esperar seu momento de processamento entre as máquinas, o que indica que existe estoque no processo.
Definições relativas às máquinas	
M1	Cada centro de máquinas é formado por apenas uma única máquina, ou seja, há apenas uma máquina de cada tipo na fábrica.
M2	Cada máquina está pronta para uso no início do período de programação.
M3	Cada máquina opera independentemente das outras, sendo assim capaz de operar pela sua própria taxa máxima de produção.
M4	Cada máquina pode processar no máximo uma tarefa de cada vez.
M5	Cada máquina está continuamente disponível para processamento das tarefas em todo o período de programação e não há interrupções como por quebra de máquinas ou manutenção.
Definições relativas às políticas de operação	
P1	Cada tarefa é processada o mais cedo possível, de forma que não há espera intencional de tarefa ou tempo ocioso de máquina.
P2	Cada tarefa é considerada uma entidade indivisível, mesmo sendo esta composta de um número de unidades individuais.
P3	Cada tarefa, uma vez aceita, é processada até que esteja completa, de forma que não é permitido cancelamento de tarefas.
P4	Cada tarefa, uma vez iniciada em uma máquina, deve ser finalizada antes que outra tarefa seja iniciada nesta mesma máquina, de forma que prioridades de antecipação não são determinadas.
P5	Cada tarefa é processada em apenas uma máquina de cada vez.
P6	Cada máquina é provida com um espaço adequado de armazenamento de forma que as tarefas possam esperar por seu momento inicial de processamento.
P7	As máquinas não são utilizadas para nenhum outro propósito durante o período de programação.
P8	Cada máquina processa as tarefas na mesma seqüência, de forma que nenhuma máquina poderá deixar de processar nenhuma tarefa.

Quando expostos em sua forma geral, conforme mencionado inicialmente por GUPTA e STAFFORD [2006], os problemas do tipo *flow shop* podem apresentar até $(n!)^m$

possibilidades de sequenciamento, sendo n o número de tarefas e m o número de máquinas, o que indica que, mesmo nos casos de problemas pequenos, com cinco tarefas, *i.e.* $n = 5$, e com cinco máquinas, *i.e.* $m = 5$, o número de possibilidades de sequenciamento ($\approx$ 25 x 10^9) pode ser tão alto que uma enumeração direta torna-se economicamente inviável. Numa versão mais simplificada, onde a ordem em que as tarefas são processadas em cada máquina deve ser sempre a mesma e nenhuma máquina pode ser desconsiderada no processamento das diferentes tarefas (assim como ocorre no modelo de JOHNSON), o número de possibilidades de sequenciamento é reduzido a $n!$ (120 para $n = 5$ e $\approx$36 x 10^5 para $n = 10$), ou seja, mesmo considerando esta versão mais simplificada, dependendo do número de tarefas, pode haver uma grande dificuldade na identificação de uma seqüência que gere um resultado ótimo.

1.2. PROBLEMAS DE PROGRAMAÇÃO DO TIPO *JOB SHOP*

Os problemas do tipo *job shop* podem ser vistos como uma generalização dos problemas apresentados na secção anterior. Aqui cada tarefa é referida como sendo um grupo de operações que, como no caso anterior, devem ser processadas em um conjunto de máquinas distintas, contudo, a ordem de processamento das operações de cada tarefa segue uma seqüência específica que varia de uma tarefa para outra conforme o problema apresentado, ou seja, o fluxo de processamento não deverá mais necessariamente ser um fluxo unidirecional e uma mesma tarefa poderá ser processada em uma mesma máquina mais de vez. As mesmas definições apresentadas na Tabela 1.1 para representação de problemas do tipo *flow shop* podem ser aplicadas na descrição dos problemas de programação da produção do tipo *job shop* exceto nos casos das definições **J5**, **J7**, **M1** e **P8**, que devem ser alteradas conforme apresentado na Tabela 1.2.

Tabela 1.2 – Generalizações apresentadas para problemas do tipo *job shop*

Definições relativas às tarefas	
J5	Cada tarefa possui uma ordem predefinida de processamento sendo que esta não deverá necessariamente ser a mesma para todas as tarefas consideradas.
J7	Toda tarefa pode ser processada mais de uma vez em cada máquina.
Definições relativas às máquinas	
M1	Pode haver várias máquinas do mesmo tipo em cada centro de máquinas.
Definições relativas às políticas de operação	
P8	A seqüência em que cada máquina processa as tarefas é variável, sendo que não será necessário que cada uma das tarefas seja processada por todas as máquinas.

Outras descrições de problemas do tipo *job shop* podem ser encontradas em GONÇALVES *et al.* [2005], JANSEN *et al.* [2005], MOSHEIOV e ORON [2005], NOWICKI e SMUTNICKI [1996, 2005], e TAVAKKOLI-MOGHADDAM e DANESHMAND-MEHR [2005], entre outros.

1.3. PROBLEMAS DE PROGRAMAÇÃO DO TIPO *FLEXIBLE MANUFACTURING SYSTEMS*

Os *flexible manufacturing systems* (FMSs), ou sistemas flexíveis de manufatura, podem ser definidos, de uma forma geral, como sendo sistemas de produção altamente automatizados, capacitados a produzir uma grande variedade de diferentes peças e produtos, usando o mesmo equipamento e o mesmo sistema de controle. Tais sistemas podem ser decompostos em pelo menos três subsistemas, que são [SOCIESC, 2005]:

- Sistema de Armazenamento e Processamento de Material - equipamentos automatizados ou robotizados que fornecem e gerenciam material.
- Sistema de Processamento - grupo de máquinas-ferramentas com comando numérico (CN) ou comando numérico computadorizado (CNC).
- Sistema de Controle Computadorizado - realiza o controle operacional do conjunto.

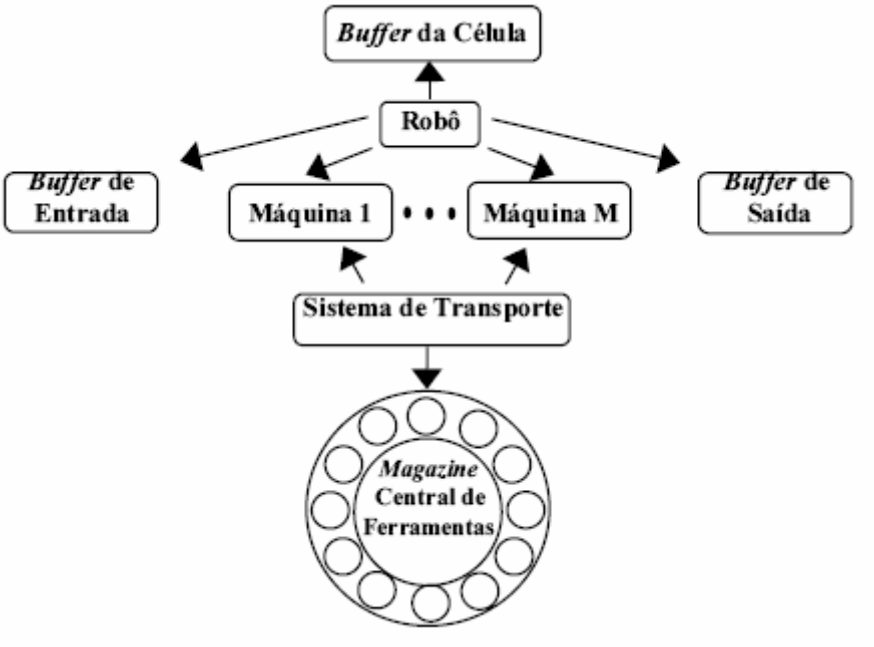


Figura 1.1 – Exemplo de um SFM onde máquinas-ferramentas integradas e controladas por computador aparecem interligadas por um veículo automatizado de transporte. [TEIXEIRA e MENDES, 1996]

Na Fig. 1.1 é apresentado um exemplo de um FMS onde aparece a interação entre estes três subsistemas. Nesta figura os sistemas de armazenamento e processamento de material são representados pelos *buffers* de entrada, de saída e da célula, e pela *magazine* central de ferramentas (locais de armazenagem), pelo robô e pelo sistema de transporte (sistemas de armazenamento e processamento de material). Já o sistema de processamento é representado pelo conjunto de máquinas (máquina 1, ..., máquina M) e o sistema de controle computadorizado, apesar de não aparecer na figura, é o sistema que controla todo o funcionamento do conjunto, cuidando para que a produção ocorra de forma harmoniosa.

Conforme apresentado por BUYURGAN *et al*. [2004], os problemas relacionados à tecnologia de manufatura flexível são relativamente complexos se comparados com os sistemas tradicionais de manufatura onde os tempos de processamentos são mais longos, os níveis de inventários são maiores e as taxas de utilização menores. Para os autores a dificuldade é originada primariamente do objetivo fundamental que existe por trás do conceito de um FMS: ser tão eficiente quanto um meio de produção em massa e ainda tão flexível quanto um modelo de fabricação do tipo *job shop*. Desta forma, uma vez que: (a) cada máquina em um FMS é completamente versátil e capaz de processar uma grande quantidade de operações diferentes; (b) o sistema pode executar diferentes operações de uma mesma tarefa simultaneamente; e (c) cada tarefa pode ter rotas alternativas através do sistema, é realmente complexo resolver problemas de programação para ambientes de produção do tipo FMS. Tal complexidade faz com que problemas deste tipo sejam abordados de forma diversificada dentro da literatura de acordo com o enfoque dado por cada autor.

Segundo SHANKER e MODI [1999], geralmente, o problema de programação em um ambiente do tipo FMS pode ser acomodado em algum dos muitos modelos de programação disponíveis, desenvolvidos para *job shops* já que os FMSs podem ser considerados como uma generalização destes, contudo, ainda segundo estes autores, algumas características dos FMSs devem ser cuidadosamente observadas e incorporadas ao problema de programação, tais como:

1. uma variedade de produtos é produzida em lotes de tamanho médio e várias partes são produzidas simultaneamente;
2. as tarefas chegam em momentos variados e suas devidas datas de processamento são geralmente apertadas;
3. há o processamento de material intensivo em capital e são empregados equipamentos de manejo do mesmo;

4. os equipamentos de processamento são funcionalmente versáteis;
5. é necessário que haja um controle, em tempo real, sobre as decisões de programação, em resposta ao comportamento dinâmico do sistema e para atingir uma utilização efetiva dos recursos;
6. são necessárias decisões a respeito dos vários recursos de fabricação no intuito de explorar a flexibilidade provida pelas alternativas de substituição de alguns dos recursos.

Aqui também as definições apresentadas na Tabela 1.1 para representação de problemas do tipo *flow shop* podem ser aplicadas na descrição dos problemas de programação da produção do tipo FMS, exceto nos casos das definições **J1**, **J2**, **J5**, **J6**, **J7**, **M1, P5** e **P8**, que devem ser alteradas tomando-se como base as informações apresentadas nesta secção, conforme indicado na Tabela 1.3.

Tabela 1.3 – Generalizações apresentadas para problemas do tipo FMS.

Definições relativas às tarefas	
J1	As tarefas chegam em momentos variados.
J2	Como o sistema é flexível e decisões com relação à programação da produção devem ser tomadas em tempo real, os momentos de execução de cada tarefa estarão sujeitos a modificações.
J5	A ordem de processamento de cada tarefa poderá ser alterada em tempo real e esta não deverá necessariamente ser a mesma para todas as tarefas consideradas.
J6	O tempo de processamento das tarefas entre as várias máquinas é um tempo finito e conhecido, contudo os tempos de transporte e de *setup* podem ser considerados separadamente, assim como pode ser considerada a relação de dependência que existe entre estes e a ordem de processamento das tarefas.
J7	Toda tarefa pode ser processada mais de uma vez em cada máquina.
Definições relativas às máquinas	
M1	Pode haver várias máquinas do mesmo tipo em cada centro de máquinas.
Definições relativas às políticas de operação	
P5	Uma mesma tarefa pode ser processada em várias máquinas ao mesmo tempo.
P8	A seqüência em que cada máquina processa as tarefas é variável, sendo que não será necessário que cada uma das tarefas seja processada por todas as máquinas.

É importante ressaltar que, apesar de não relacionado na Tabela 1.3, no caso dos FMSs os problemas de programação da produção deixam de ser problemas apenas relacionados ao seqüenciamento de tarefas e passam a considerar a alocação dos recursos de produção incluindo as ferramentas que devem ser acopladas às máquinas de acordo com a operação que será executada em cada momento.

Outras descrições de problemas do tipo FMS podem ser encontradas em SOMLÓ [2004], SWARNKAR e TIWARI [2004], YAN *et al.* [1998] e YU *et al.*[2003] entre outros.

1.4. PROBLEMA DE PROGRAMAÇÃO DA PRODUÇÃO OCORRENTE NAS EMPRESAS DO SETOR DE CONFECÇÕES DO MUNICÍPIO DE NOVA FRIBURGO

O processo de produção das empresas de confecção começa com a criação e a modelagem dos produtos. Com base nos pedidos e nas projeções de venda dos modelos gerados, são definidos os produtos que devem ser produzidos simultaneamente durante um período de tempo pré-determinado e as suas respectivas quantidades. Posteriormente, no setor de corte (Fig. 1.2 (a)), os tecidos são enfestados e é realizado o corte do enfesto procurando-se aproveitá-lo ao máximo, por intermédio do processo de encaixe. Após o corte, os componentes gerados são distribuídos em lotes, de acordo com a cor e o modelo de cada produto, e posteriormente encaminhados para a área de processamento (Fig. 1.2 (b)) onde são costurados nos processos de sub-montagens e montagens, possibilitando assim a produção da peça. Após isso, as peças são encaminhadas para o setor de limpeza e acabamento (Fig. 1.2 (c)) onde, como o próprio nome sugere, é feito o acabamento, com a colocação de peças

(a) Setor de corte: onde ocorrem os processos de formação dos lotes de produção.

(b) Setor de costura: onde ocorrem os processos de montagens e sub-montagens necessários à produção das peças.

(c) Setor de limpeza e acabamento:

Figura 1.2 – Processo de produção na forma como geralmente ocorre nas empresas de confecção do município de Nova Friburgo

acessórias, tais como rebites, botões, zipers etc, bem como retiradas pontas de linha, inspeção final e outros acabamentos pertinentes. No caso das confecções de roupas, há também a passadoria onde é feita a passagem da peça pronta com ferros de engomar. Finalmente é realizada a embalagem e os produtos se tornam disponíveis para estoque e/ou entrega ao cliente. Estas etapas podem sofrer algumas variações em função do tipo de produto que deverá ser confeccionado.

O problema de programação da produção é avaliado no momento em que são definidos quais produtos devem ser confeccionados no período mencionado sendo que, conforme apresentado na Fig. 1.3, este considera todas as operações realizadas após a geração dos lotes que, no caso em análise, constituem as tarefas que deverão ser realizadas ao longo do processo. Como, no setor de confecções, os avanços tecnológicos, como, por exemplo, os equipamentos de CAD/CAM (*Desingn*/ *Computer Aided Manufacturing*) e de comando numérico, são utilizados apenas nas fases anteriores a da costura e como o processo de confecção é intensivo em mão-de-obra, o que o torna um processo tipicamente artesanal, o modelo apresentado não pode ser tratado como um modelo de fabricação do tipo FMS, que se caracteriza justamente pelo seu alto nível de automação, contudo há também uma série de considerações que aproximam o nosso problema dos problemas desta natureza e o afastam dos problemas apresentados como *job shop*, de forma que o problema ocorrente no setor avaliado permeia entre estes dois modelos de produção. Tais considerações serão melhor definidas na próxima secção.

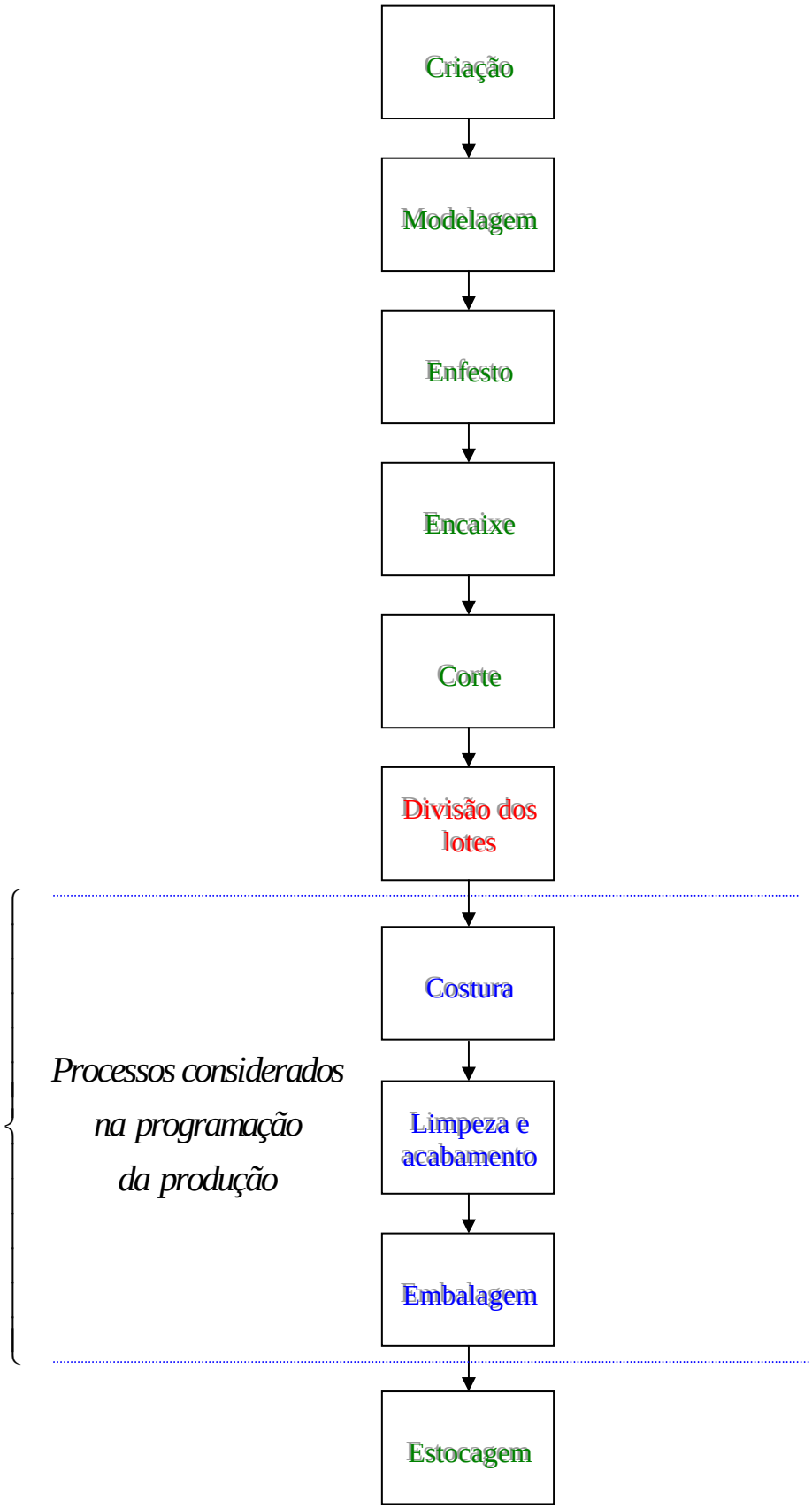


Figura 1.3 – Fluxograma do processo de produção das empresas do setor de confecções

1.4.1. Comparação entre o modelo de JOHNSON e o problema avaliado nesta dissertação

Para melhor entendimento do problema aqui tratado, o modelo de produção do setor de confecções do município de Nova Friburgo é comparado na Tabela 1.4 com o modelo de JOHNSON, conforme a descrição de GUPTA e STAFFORD [2006] apresentada na secção 1.1 desta mesma dissertação.

Tabela 1.4 – Comparação entre o modelo de JOHNSON e o problema tratado nesta dissertação

Modelo de Johnson (1954)		**Problema tratado nesta dissertação**
Definições relativas às tarefas		
J1	Cada tarefa está livre para ser executada no início do período de programação.	O mesmo serve para o caso avaliado neste trabalho.
J2	Cada tarefa deve ter o seu próprio momento de execução que é fixo e não está sujeito a modificações	O momento de execução de cada tarefa irá variar de acordo com a distribuição das operações entre as máquinas e com a seqüência de processamento destas operações em cada uma delas.
J3	Cada tarefa é independente das outras.	Aqui também não há dependência entre as tarefas.
J4	Cada tarefa consiste em um conjunto de operações específicas, sendo que cada uma destas deve ser executada por apenas uma máquina.	O mesmo serve para o caso avaliado neste trabalho.
J5	Cada tarefa possui uma ordem predefinida de processamento que é a mesma para todas as tarefas consideradas.	A ordem de processamento não deverá ser a mesma para todas as tarefas e poderá ser variável, sendo que esta deverá respeitar algumas restrições de sequenciamento inerentes a cada tarefa.
J6	Cada tarefa requer um tempo de processamento entre as várias máquinas finito e conhecido. Este tempo de processamento inclui tempo de transporte e de *setup*, sendo que estes são independentes das tarefas precedentes e subseqüentes.	O tempo de processamento das tarefas entre as várias máquinas é um tempo finito e conhecido, contudo os tempos de transporte e de *setup* devem ser considerados separadamente e são completamente dependentes das tarefas precedentes e subseqüentes.
J7	Cada tarefa é processada apenas uma única vez em cada máquina.	Todas as tarefas podem ser processadas várias vezes numa mesma máquina.

Continuação Tabela 1.4		
J8	Cada tarefa deve esperar seu momento de processamento entre as máquinas, o que indica que existe estoque no processo.	O mesmo serve para o caso avaliado neste trabalho.
Definições relativas às máquinas		
M1	Cada centro de máquinas é formado por apenas uma única máquina, ou seja, há apenas uma máquina de cada tipo na fábrica.	Pode haver várias máquinas de um mesmo tipo na fábrica. Contudo estas máquinas, apesar de servirem às mesmas finalidades, geralmente operam com diferentes tempos de processamento, de forma que as mesmas podem ser consideradas separadamente.
M2	Cada máquina está pronta para uso no início do período de programação.	O mesmo serve para o caso avaliado neste trabalho.
M3	Cada máquina opera independentemente das outras, sendo assim capaz de operar pela sua própria taxa máxima de produção.	O mesmo serve para o caso avaliado neste trabalho.
M4	Cada máquina pode processar no máximo uma tarefa de cada vez.	O mesmo serve para o caso avaliado neste trabalho.
M5	Cada máquina está continuamente disponível para processamento das tarefas em todo o período de programação e não há interrupções como por quebra de máquinas ou manutenção.	O mesmo serve para o caso avaliado neste trabalho.
P1	Cada tarefa é processada o mais cedo possível, de forma que não há espera intencional de tarefa ou tempo ocioso de máquina.	O mesmo serve para o caso avaliado neste trabalho.
P2	Cada tarefa é considerada uma entidade indivisível, mesmo sendo esta composta de um número de unidades individuais.	O mesmo serve para o caso avaliado neste trabalho.
P3	Cada tarefa, uma vez aceita, é processada até que esteja completa, de forma que não é permitido cancelamento de tarefas.	O mesmo serve para o caso avaliado neste trabalho.
P4	Cada tarefa, uma vez iniciada em uma máquina, deve ser finalizada antes que outra tarefa seja iniciada nesta mesma máquina, de forma que prioridades de antecipação não são determinadas.	O mesmo serve para o caso avaliado neste trabalho.
P5	Cada tarefa é processada em apenas uma máquina de cada vez.	Uma mesma tarefa pode ser processada em várias máquinas ao mesmo tempo.

Continuação Tabela 1.4		
P6	Cada máquina é provida com um espaço adequado de armazenamento de forma que as tarefas possam esperar por seu momento inicial de processamento.	Geralmente, quando não há espaço na máquina, este espaço é gerado adaptando-se mesinhas ou caixotes para armazenagem.
P7	As máquinas não são utilizadas para nenhum outro propósito durante o período de programação.	O mesmo serve para o caso avaliado neste trabalho.
P8	Cada máquina processa as tarefas na mesma seqüência, de forma que não são permitidos pulos entre as máquinas.	A seqüência em que cada máquina processa as tarefas é variável, e são permitidos pulos entre as máquinas.

É importante ressaltar que as considerações apresentadas na Tabela 1.4 para definição do problema de programação da produção relacionado ao setor de confecções do município de Nova Friburgo, retratam apenas parte dos aspectos do ambiente de produção avaliado. De fato outras considerações, como a limitação de insumos, *i.e.* matéria prima e aviamentos, e a possível necessidade de se incluir novas tarefas no conjunto de tarefas processadas durante o período de execução das mesmas, seriam necessárias para uma maior aproximação entre o problema descrito nesta dissertação e o problema real ocorrente nestas empresas. Contudo, como tais considerações não são tratadas neste trabalho, as mesmas ficam como sugestões para o desenvolvimento de trabalhos futuros.

1.5. MÉTODOS DE OTIMIZAÇÃO E APROXIMAÇÃO

Atualmente pode-se encontrar uma série de publicações que apresentam diferentes soluções para os problemas “padrões” de programação da produção apresentados anteriormente, beneficiando-se da existência de uma ampla variedade de métodos que, segundo JAIN e MEERAN [1999], podem ser distribuídos entre métodos de otimização e

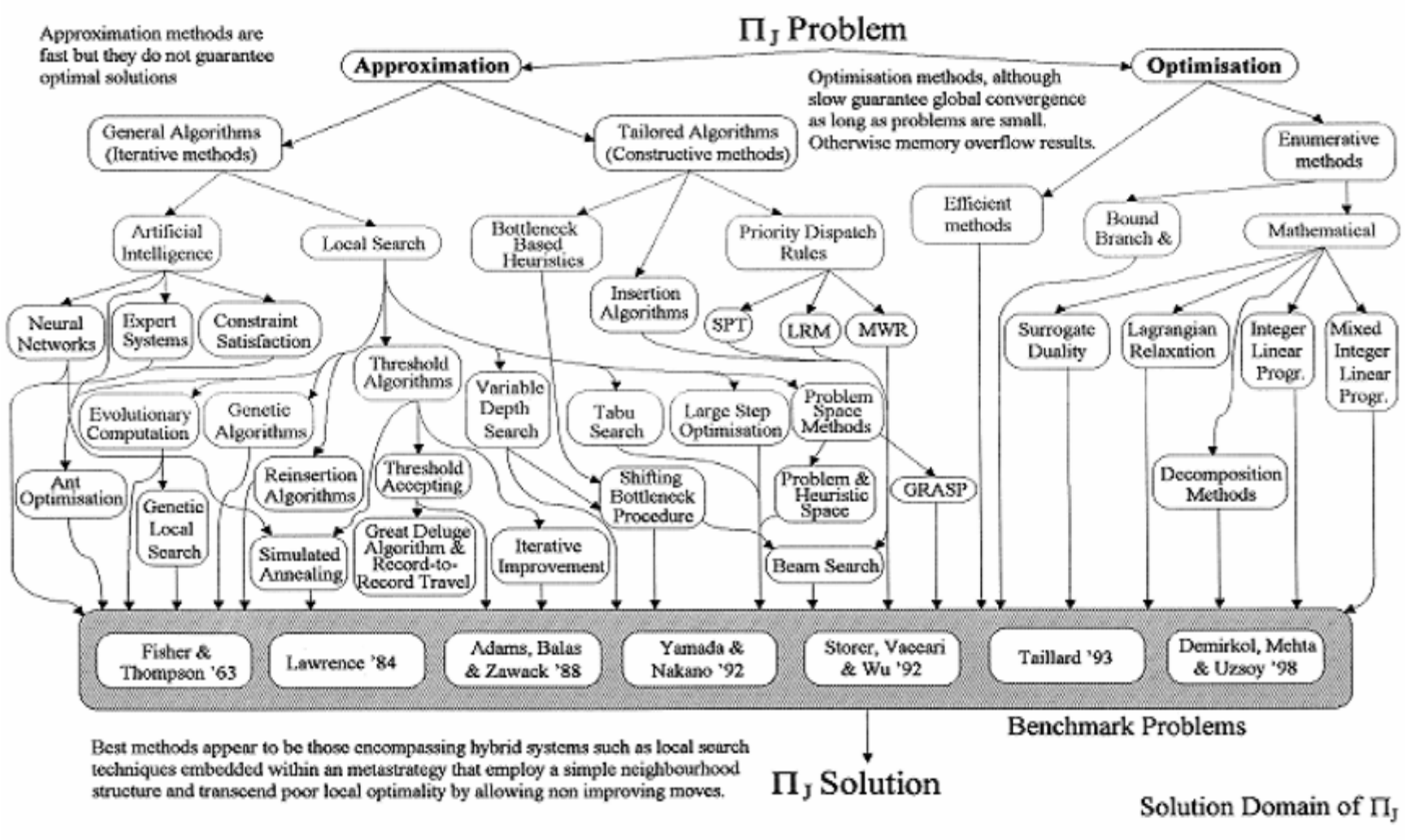


Figura 1.4 – Métodos de otimização e aproximação aplicados a um *benchmark* do tipo *job shop*, conhecido como Π_J (JAIN e MEERAN, 1999).

métodos de aproximação (Fig. 1.4). Em sua classificação os autores estavam se referindo especificamente a métodos aplicados a um *benchmark* do tipo *job shop*, conhecido como Π_J, muito utilizado em pesquisas como base para testes de desempenho dos novos algoritmos desenvolvidos, contudo os mesmos métodos são, ainda hoje, utilizados para solução de todos os tipos de problemas de programação da produção de forma que a classificação apresentada pelos autores pode ser adotada de forma generalizada.

1.5.1. MÉTODOS DE OTIMIZAÇÃO

Segundo JAIN e MEERAN [1999], os métodos de otimização são aqueles que garantem uma convergência global para problemas pequenos, mas, por outro lado, exigem um grande espaço de memória, não sendo viáveis para solução de problemas maiores. Dentre os métodos de otimização o que apresenta a principal estratégia computacional é o *Branch and Bound* (BB) onde uma árvore construída dinamicamente, representando o espaço solução de todos os sequenciamentos possíveis, é implicitamente testada. Esta técnica formula procedimentos e regras que permitem que uma grande porção da árvore seja removida, de forma que o teste se estenda a um espaço-solução resumido. Como exemplo de aplicação do BB temos o trabalho desenvolvido por CHUNG *et* al. [2005] onde o método é utilizado para a solução de um problema do tipo *flow shop* (com $n!$ possibilidades de sequenciamento), para um número indeterminado de máquinas e número de tarefas ≤ 20. Conforme apresentado pelos autores, são poucos os trabalhos publicados que utilizam métodos de otimização para problemas relacionados ao sequenciamento de tarefas em mais de uma máquina e, mesmo no caso dos problemas com apenas uma máquina, o desempenho de grande parte dos métodos desenvolvidos torna-se limitado pelo número reduzido de tarefas que podem ser consideradas, contudo existe hoje uma forte expectativa de que, com o avanço da capacidade de

processamento dos computadores, os métodos de otimização tornem-se cada vez mais efetivos na solução de problemas como os de programação da produção.

1.5.2. MÉTODOS DE APROXIMAÇÃO

Os métodos de aproximação são métodos rápidos que, no entanto, não garantem solução ótima. Estes métodos estão subdivididos entre métodos iterativos e métodos construtivos, sendo que os métodos iterativos podem ainda ser subdivididos entre métodos de busca local e métodos de inteligência artificial. Os métodos de busca local possuem estratégias capazes de direcionar algoritmos míopes a soluções ótimas pela aceitação de soluções ruins [JAIN e MEERAN, 1999]. Dentre os métodos de busca local, três merecem destaque, são eles: *Simulated Annealing* (SA), *Genetic Algorithms* (GA) e *Tabu Search* (TS). Enquanto o SA é expresso como uma técnica de busca local com orientação randômica, introduzida como uma analogia ao processo de resfriamento de um metal aquecido até o seu estado mínimo de energia, o GA se apresenta como uma técnica de busca baseada num modelo abstrato da evolução natural, onde novos indivíduos (ou seqüências) são construídos a partir de indivíduos que apresentaram melhor adaptação ao ambiente (restrições do problema) [JAIN e MEERAN, 1999]. Já o TS é uma técnica simples de aproximação iterativa com orientação randômica que, de forma inteligente, guia um processo de busca através do conceito de vizinhança.

Os métodos de inteligência artificial são essencialmente técnicas de busca heurística, baseadas em mecanismos de raciocínios lógicos e analógicos. Dentre estes, um dos mais recentes é o *ant colony optimization* (ACO), o qual é inspirado no comportamento de colônias de formigas durante a procura de alimento [SOLIMANPUR *et al.*, 2005].

1.6. OBJETIVO ESPECÍFICO DESTA DISSERTAÇÃO

Esta dissertação tem como objetivo principal o desenvolvimento de uma ferramenta computacional para a programação da produção de empresas do setor de confecções do município de Nova Friburgo.

Dentro da literatura pesquisada referente aos métodos de otimização e aproximação desenvolvidos para solução de problemas de programação da produção, o trabalho apresentado por NOWICKI e SMUTNICKI [1996], que descreve um algoritmo rápido e de fácil implementação baseado no método *tabu search*, é ainda hoje mencionado como estado da arte no que diz respeito à aplicação de tais métodos em problemas como os aqui considerados. Neste trabalho as noções de caminho crítico e blocos de operações são aplicadas na definição de vizinhança, fazendo com que o algoritmo encontre *makespans* (tempos totais de processamento) menores do que os melhores métodos de aproximação desenvolvidos (até a data de publicação referente à pesquisa) para a solução de problemas de programação do tipo *job shop*, utilizando um tempo de CPU também muito menor. Ainda hoje considera-se que tal algoritmo representou um grande passo com relação à noção de eficiência dos métodos de aproximação para problemas do tipo *job shop*, já que ofereceu uma simples implementação, com um tempo de CPU muito curto e boa precisão [NOWICKI e SMUTNICKI, 2005] Por este motivo o mesmo será utilizado como base para solução do problema tratado nesta dissertação.

Esta dissertação está estruturada em cinco capítulos.

No capítulo dois são apresentadas a modelagem e a formulação do problema de programação da produção, sendo esta última definida com base no trabalho de NOWICKI e SMUTNICKI [1996]. À formulação original proposta pelos autores são adicionadas a consideração da possibilidade de processamento de uma mesma operação em mais de uma

máquina e da ocorrência dos tempos de *setup* e de transporte, assim como a relação de dependência que existe entre estes, as máquinas que poderão processar cada operação e a seqüência de operações em cada uma destas máquinas.

No capítulo três, é apresentada uma solução para os problemas de sequenciamento da produção baseada no método *Tabu Search*. Também são apresentados uma definição geral do método TS e os detalhes de implementação considerados em cada passo do mesmo (como geração da solução inicial, geração de vizinhança, lista de restrições, etc.). Simultaneamente, serão descritos outros aspectos considerados relevantes (como, por exemplo, a forma de representação da lista tabu), para uma maior compreensão e conhecimento do algoritmo implementado.

No capítulo quatro são apresentados os resultados obtidos para os testes de validação do algoritmo proposto (considerando um caso de um problema de seqüência simples, para diferentes números de máquinas e tarefas, e um *benchmark*, conhecido na literatura como Π_j, apresentado por JAIN e MEERAN [1999]), e outros testes considerando problemas mais complexos (artificiais e reais), que representam algumas das peculiaridades do setor de confecções. Neste mesmo capítulo também são apresentados gráficos de análise de sensibilidade aos parâmetros do modelo proposto (número de vizinhos, percentual de possibilidade de troca de máquinas, etc.).

Finalmente, no capítulo cinco, é apresentada uma conclusão desta dissertação assim como algumas recomendações para trabalhos futuros.

CAPÍTULO 2

2 MODELAGEM DA PROGRAMAÇÃO DA PRODUÇÃO

No modelo proposto nesta dissertação cada tarefa J_i deve ser entendida como todo o processamento de um determinado lote de produção desde o momento em que este sai da mesa de corte até o momento em que todas as peças, geradas nos processos de montagens e sub-montagens deste mesmo lote, encontram-se prontas para distribuição e/ou armazenagem. Desta forma, J_i poderá ser definida como um conjunto de operações $J_i = \{O_{i1}, O_{i2}, O_{i3}, \cdots, O_{if_i}\}$ (com $i = 1, 2, \cdots, N$, onde N representa o número de tarefas, e f_i o número total de operações da tarefa J_i, vide Fig. 2.1), que, durante o seu processamento, devem ser executadas em um conjunto de máquinas distintas $M = \{M_1, M_2, \cdots, M_m\}$, sendo m o número total de máquinas disponíveis para processamento do conjunto de tarefas considerado, respeitando-se uma dada ordem de prioridades previamente definida de acordo com o modelo de fabricação de cada produto.

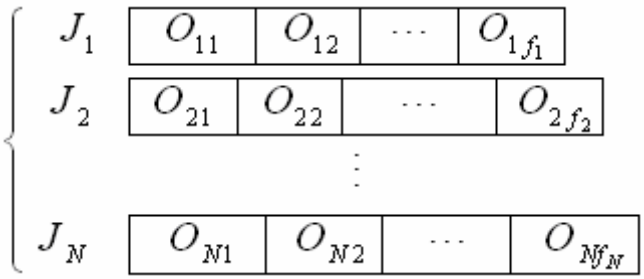


Figura 2.1 – Conjunto de operações por tarefa

Para cada operação O_{ij} existirá um conjunto finito de máquinas, dentro do conjunto de máquinas disponíveis no ambiente de produção ($M_{ij}, M_{ij} \subset M$) capaz de executá-la, sendo que o tempo de execução de cada operação, O_{ij}, em cada máquina, M_k, $k = 1, \cdots, m$, ($T_{ij,k}$) poderá variar de acordo com a máquina em que esta será processada (Tabela 2.1). O símbolo x na Tabela 2.1 significa que a operação não pode ser realizada na máquina indicada.

Tabela 2.1– Relação de tempo de processamento de operação por máquina.

	J_1			
	O_{11}	O_{12}	$\cdots$	O_{1f_1}
M_1	x	$T_{12,1}$	$\cdots$	$T_{1f_1,1}$
M_2	x	$T_{12,2}$	$\cdots$	$T_{1f_1,2}$
M_3	$T_{11,3}$	$T_{12,3}$	$\cdots$	$T_{1f_1,3}$
M_4	$T_{11,4}$	x	$\cdots$	x
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
M_m	x	x	$\cdots$	$T_{1f_1,m}$

	J_2			
	O_{21}	O_{22}	$\cdots$	O_{2f_2}
M_1	$T_{21,1}$	x	$\cdots$	x
M_2	$T_{21,2}$	x	$\cdots$	x
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
M_m	$T_{21,m}$	$T_{22,m}$	$\cdots$	$T_{2f_2,m}$

$\vdots$

	J_N			
	O_{N1}	O_{N2}	$\cdots$	O_{Nf_N}
M_1	x	$T_{N2,1}$	$\cdots$	x
M_2	$T_{N1,2}$	x	$\cdots$	$T_{Nf_N,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
M_m	$T_{N1,m}$	x	$\cdots$	x

Para melhor entendimento considere a tarefa J_1 como sendo a fabricação de um lote com 50 unidades do produto CL005 (Fig. 2.2). Esta tarefa consiste na execução de nove operações, conforme apresentado na Fig. 2.2 e na Tabela 2.2.

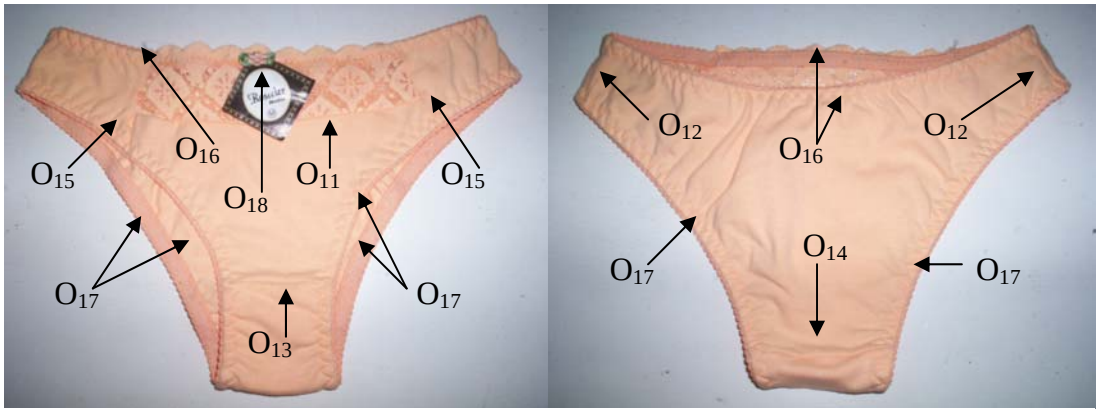


Figura 2.2 – Operações necessárias para a confecção do produto CL005.

Tabela 2.2 – Operações do produto CL005 (Ref. Fig. 2.2).

Operações	Descrição
O_{11}	União renda + frente
O_{12}	União costas + laterais (esquerda e direita)
O_{13}	União frente + forro
O_{14}	União costas + forro
O_{15}	União frente + laterais (esquerda e direita)
O_{16}	Elástico cintura
O_{17}	Elástico pernas
O_{18}	Aplicação de etiqueta + laço
O_{19}	Acabamento e embalagem

Sendo que, as restrições relacionadas à ordem de processamento deste produto se apresentam conforme a Fig. 2.3.

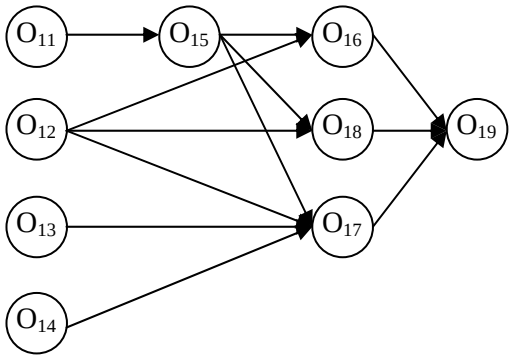


Figura 2.3 - – Relação de precedências – produto CL005.

Esta figura indica que: (a) as operações O_{11}, O_{12}, O_{13} e O_{14} poderão ser executadas em qualquer momento; (b) a operação O_{15} só poderá ser executada a partir do momento em que a operação O_{11} tiver sido finalizada; (c) as operações O_{16} e O_{18} só poderão ser executadas a partir do momento em que ambas as operações O_{12} e O_{15} tiverem sido finalizadas; (d) a operação O_{17} só poderá ser executada a partir do momento em que as operações O_{12}, O_{13}, O_{14} e O_{15} tiverem sido finalizadas; e (e) a operação O_{19} só poderá ser executada a partir do momento em que O_{16}, O_{17} e O_{18} estejam concluídas. Nesse ponto é importante ressaltar que, no decorrer do processo, dependendo da ordem em que as operações serão executadas, muitas das operações referentes à tarefa J_1 não poderão ser processadas no mesmo momento que outras operações da mesma tarefa já que existirá uma restrição relacionada ao insumo que estará sendo processado. Geralmente, no início do processamento, quando nenhuma operação foi ainda executada, as partes que deverão ser unidas (como forro, costas, etc.) encontram-se dissociadas, de forma que operações relacionadas ao processamento de partes distintas como O_{12} e O_{13}, ou O_{14} e O_{15}, poderão ser executadas ao mesmo tempo. Contudo a partir do momento que tais partes forem unidas serão geradas outras restrições de insumo de forma que será mais difícil o processamento simultâneo de diferentes operações.

O conjunto de máquinas disponíveis para processamento das tarefas no ambiente de produção considerado (M) pode ser relacionado conforme a Tabela 2.3.

Tabela 2.3 – Relação de máquinas disponíveis para processamento de tarefas

Operações	Descrição
M_1	Reta (costura básica)
M_2	Ziguezague (utilizada para rebater elásticos em lingerie, unir partes de couro, bordar, pregar zíper)
M_3	Ziguezague três pontos (utilizada para rebater elásticos em lingerie, unir partes de couro, bordar, pregar zíper)
M_4	Overloque (utilizada para fechamento ou acabamento)
M_5	Overloque (utilizada para fechamento ou acabamento)
M_6	Overloque ponto cadeia (utilizada para união de tecidos, acabamentos e fechamentos)
[1] M_7	Funcionário (limpeza e acabamento)

Já a relação entre operações e máquinas capazes de processá-las e o tempo de processamento de cada operação em cada máquina podem ser definidos da seguinte forma:

Tabela 2.4 – Relação de tempo de processamento de operação por máquina para J_1.

	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}	O_{16}	O_{17}	O_{18}	O_{19}
M_1	X	X	X	X	X	900 s	1200 s	180 s	X
M_2	X	X	X	X	X	600 s	800 s	X	X
M_3	X	X	X	X	X	450 s	600 s	X	X
M_4	600 s	400 s	200 s	200 s	400 s	X	X	X	X
M_5	600 s	400 s	200 s	200 s	400 s	X	X	X	X
M_6	900 s	600 s	300 s	300 s	600 s	X	X	X	X
M_7	X	X	X	X	X	X	X	X	1200 s

O símbolo X na Tabela 2.4 significa que a operação não pode ser realizada na máquina indicada.

[1] No modelo de produção proposto é considerado que cada funcionário é responsável pelo funcionamento de uma única máquina durante todo o período de processamento, de tal forma que funcionário e máquina formam uma única entidade a ser considerada. Quando a execução de determinada operação não exigir a utilização de equipamentos, o funcionário constituirá, sozinho, tal entidade, recebendo o mesmo tratamento. No ambiente de produção real pode acontecer de um mesmo funcionário operar várias máquinas em diferentes momentos, de fato este comportamento é típico no caso de produção em células, contudo este fato não será considerado neste trabalho.

Considerando que cada operação deverá ser processada em apenas uma máquina e que, geralmente, duas ou mais operações de uma mesma tarefa não poderão ser processadas em um mesmo momento, surgem aqui dois problemas que exigem solução, são eles: (a) qual máquina deverá processar cada operação; e (b) qual a seqüência em que as operações de uma mesma tarefa deverão ser executadas. No ambiente de produção estas questões tornam-se ainda mais complexas, já que há um conjunto de tarefas que devem ser processadas simultaneamente, disputando pelos mesmos recursos de produção (como, por exemplo, máquinas e pessoal). Assim sendo, um novo problema pode ser aqui considerado, que é: (c) dado que um grupo de operações pertencentes ou não a tarefas distintas deve ser processado por uma mesma máquina e/ou funcionário, qual a seqüência em que as operações devem ser processadas nesta máquina.

Para que tais questões sejam solucionadas é ainda necessário que seja estipulado um objetivo como, por exemplo, a redução do tempo total de processamento do conjunto de tarefas considerado. Tal objetivo é tratado na literatura como determinação da solução que gera menor *makespan* ($C_{\max}$). Outros objetivos podem ser considerados como, por exemplo:

- Atendimento aos prazos estipulados para entrega de pedidos gerando o mínimo de atraso;
- Redução do tempo ocioso de produção;
- Execução de cada operação no momento "mais tarde" possível, conforme ocorre no modelo *just in time.*

Nesta dissertação o objetivo adotado foi a redução do *makespan*, já que tal objetivo representa de forma mais realista a intenção de redução de custos, que constitui o principal interesse de grande parte dos empresários.

Para cálculo do *makespan*, deve ser considerado que:

- Quando duas operações consecutivas de uma mesma tarefa são processadas em máquinas diferentes, haverá um espaço de tempo entre o momento final de execução da primeira

operação e o momento inicial de execução da segunda operação que deverá ser reservado para transporte do material (insumo) a ser processado. Este tempo é conhecido como tempo de transporte ($TT_{k_1k_2}$) e é dado em função da distância entre as máquinas M_{k_1} e M_{k_2};

- Quando duas operações que serão executadas consecutivamente em uma mesma máquina exigem ajuste da máquina (como, por exemplo, troca de linha ou regulagem de pontos), haverá um espaço de tempo entre o momento final de execução da primeira operação e o momento inicial de execução da segunda operação que deverá ser reservado para tal ajuste. Este tempo é conhecido como tempo de *setup* ($TS_{i_1j_1i_2j_2k}$) e é dado em função das operações, $O_{i_1j_1}$ e $O_{i_2j_2}$, que deverão ser processadas consecutivamente e da máquina M_k em que estas serão executadas.
- O transporte do material é efetuado por um ou mais funcionários destinados especificamente para tal tarefa. Já o *setup* é realizado pelo funcionário responsável pelo funcionamento da máquina em que será executada a segunda operação na ordem de processamento, entre as duas operações focadas. Como o transporte e o *setup* não disputam pelos mesmos recursos (máquinas e/ou funcionários) estes podem ocorrer em um mesmo momento.

2.1 FORMULAÇÃO DO PROBLEMA DE PROGRAMAÇÃO DA PRODUÇÃO

É feita a seguir a descrição matemática do problema de programação da produção considerado nesta dissertação tendo como base as definições do modelo proposto e a formulação de um problema do tipo *flow shop* apresentada por NOWICKI e SMUTNICKI [2005].

Sendo ε_k^g o grupo de operações que deverão ser processadas na máquina M_k numa dada configuração $\varepsilon^g = \left(\varepsilon_1^g, \cdots, \varepsilon_m^g\right)$, o conjunto de possibilidades de configurações que indicam quais operações deverão ser processadas em quais máquinas é representado por $\mathrm{E} = \left\{\varepsilon^1, \varepsilon^2, \cdots, \varepsilon^{Ng}\right\}$, onde

$$Ng = \prod_{k=1}^{m} k^{x_k} \qquad \textbf{(2.1)}$$

sendo x_k o número de operações que podem ser processadas em k máquinas diferentes.

Para todo $\varepsilon^g \in \mathrm{E}$ existirá uma ordem de processamento das operações nas máquinas, $\pi^g = \left(\pi_1^g, \cdots, \pi_m^g\right)$, onde $\pi_k^g = \left(\pi_k^g(1), \cdots, \pi_k^g\left(Nm_k^g\right)\right)$ representa a permutação em M_k, sendo Nm_k^g o número de operações que deverão ser processadas nesta mesma máquina na configuração ε^g. Dado que Π_k^g simboliza o grupo de todas as permutações em M_k na configuração ε^g, então $\pi^g \in \Pi^g = \Pi_1^g \times \Pi_2^g \times \cdots \times \Pi_m^g$. A ordem de processamento π^g pode ser modelada graficamente por $G\left(\pi^g\right) = \left(O, R \cup E\left(\pi^g\right)\right)$ onde O representa um grupo de nós e $R \cup E\left(\pi^g\right)$ um grupo de arcos. Os arcos do grupo R representam a ordem de processamento nas tarefas enquanto que os arcos do grupo $E\left(\pi^g\right)$ representam a ordem de processamento nas máquinas. Cada arco R pertencente ao grafo tem peso $TT_{k_1k_2}$, cada arco $E\left(\pi^g\right)$ pertencente ao grafo tem peso $TS_{i_1j_1i_2j_2k}$, e cada nó $O_{ij} \in O$ tem peso $T_{ij,k}$, sendo $T_{ij,k}$ o tempo de execução da operação O_{ij} na máquina M_k.

A ordem de processamento π^g cujo grafo não contém um ciclo, será chamada de ordem de processamento viável. Para a ordem de processamento viável π^g, o comprimento do maior caminho em $G\left(\pi^g\right)$ entre os caminhos que seguem para o nó O_{ij} (incluindo o peso

do nó) será denotado por $r_\pi(ij)$. Já o comprimento do maior caminho em $G(\pi^g)$ entre os caminhos que partem do nó O_{ij} (incluindo o peso do nó) será denotado por $q_\pi(ij)$. O *makespan* $C_{\max}(\pi^g)$ para esta ordem de processamento π^g equivale ao comprimento do maior caminho (caminho crítico) em $G(\pi^g)$.

Com base nesta descrição o objetivo proposto pode ser reescrito como a busca de uma possível ordem de processamento $\pi^g \in \Pi$ que minimize $C_{\max}(\pi^g)$, assim sendo a função objetivo pode ser formulada como

$$C^*_{\max} = \min\left(C_{\max}(\pi^g) : \forall \pi^g \in \Pi^g\right) \quad \textbf{(2.2)}$$

Os momentos iniciais de processamento de cada operação (S_{ijk}) podem ser determinados conforme

$$S_{ijk} = r_\pi(ij) - T_{ij,k} \quad \textbf{(2.3)}$$

Para melhor entendimento da formulação apresentada, vamos considerar uma segunda tarefa J_2 como sendo a fabricação de 30 unidades do produto ST003. Esta tarefa consiste na execução de cinco operações cuja relação de dependência representa uma relação de dependência simples, onde O_{2j} será sempre dependente de $O_{2(j-1)}$, exceto no caso em que $j = 1$ (Fig. 2.4).

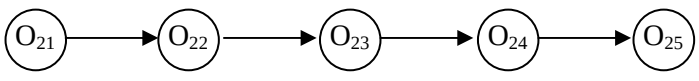


Figura 2.4 – Relação de precedências – produto ST003.

Já a relação entre estas operações e as máquinas capazes de processá-las e o tempo de processamento de cada operação em cada máquina aparecem conforme apresentado na

Tabela 2.5. O símbolo X apresentado nesta tabela significa que a operação não pode ser realizada na máquina indicada.

Tabela 2.5 – Relação de tempo de processamento de operação por máquina para J_2

	O_{21}	O_{22}	O_{23}	O_{24}	O_{25}
M_1	600 s	X	300 s	X	X
M_2	500 s	X	350 s	X	X
M_3	450 s	X	400 s	X	X
M_4	X	400 s	X	200 s	X
M_5	X	500 s	X	250 s	X
M_6	X	600 s	X	300 s	X
M_7	X	X	X	X	720 s

Unindo as informações apresentadas nas Tabelas 2.4 e 2.5, teremos que três operações podem ser executadas em apenas uma máquina ($x_1 = 3$), quais seja O_{18}, O_{19} e O_{25}, e onze operações podem ser executadas em três máquinas diferentes ($x_3 = 11$), três operações podem ser executadas em apenas uma máquina ($x_1 = 3$), quais seja O_{1j} com $j = 1,2,\cdots,7$ e O_{2j} com $j = 1,2,\cdots,4$, de tal forma que, partindo-se da Eq. (2.1), teremos:

$$Ng = 1^3 \times 2^0 \times 3^{11} \times 4^0 \times 5^0 \times 6^0 \times 7^0 = 177.147 .$$

Este valor indica que há 177.147 possibilidades de configurações de distribuição de operações por máquinas, sendo que na Tabela 2.6 são apresentas três destas possibilidades.

A primeira possibilidade ε^1 indica que as operações $O_{16}, O_{17}, O_{18}, O_{21}$ e O_{23} deverão ser executadas na máquina M_1, as operações $O_{11}, O_{12}, O_{13}, O_{14}$ e O_{15} deverão ser executadas na máquina M_4, as operações O_{22}, O_{24} deverão ser executadas na máquina M_5 e, finalmente, as operações O_{19}, O_{25} deverão ser executadas na máquina M_7.

Tabela 2.6 – Exemplos de possibilidades de configurações de distribuição de operações por máquina

1ª possibilidade (ε^1)	2ª possibilidade (ε^2)	3ª possibilidade (ε^3)
$\varepsilon_1^1 = \{O_{16}, O_{17}, O_{18}, O_{21}, O_{23}\}$	$\varepsilon_1^1 = \{O_{16}, O_{18}, O_{21}, O_{23}\}$	$\varepsilon_1^1 = \{O_{18}\}$
$\varepsilon_2^1 = \{\ \}$	$\varepsilon_2^1 = \{O_{17}\}$	$\varepsilon_2^1 = \{\ \}$
$\varepsilon_3^1 = \{\ \}$	$\varepsilon_3^1 = \{\ \}$	$\varepsilon_3^1 = \{O_{16}, O_{17}, O_{21}, O_{23}\}$
$\varepsilon_4^1 = \{O_{11}, O_{12}, O_{13}, O_{14}, O_{15}\}$	$\varepsilon_4^1 = \{\ O_{12}, O_{13}, O_{15}\}$	$\varepsilon_4^1 = \{\ O_{12}, O_{13}, O_{15}\}$
$\varepsilon_5^1 = \{O_{22}, O_{24}\}$	$\varepsilon_5^1 = \{O_{22}, O_{24}\}$	$\varepsilon_5^1 = \{O_{22}, O_{24}\}$
$\varepsilon_6^1 = \{\ \}$	$\varepsilon_6^1 = \{O_{11}, O_{14}\}$	$\varepsilon_6^1 = \{O_{11}, O_{14}\}$
$\varepsilon_7^1 = \{O_{19}, O_{25}\}$	$\varepsilon_7^1 = \{O_{19}, O_{25}\}$	$\varepsilon_7^1 = \{O_{19}, O_{25}\}$

Sendo Nm_k^g o número de operações que deverão ser processadas na máquina M_k na configuração ε^g, o número de possibilidades de ordens de processamento das operações nas máquinas ($N\pi^g$) será dado por:

$$N\pi^g = \prod_{k=1}^{m} \left(Nm_k^g ! \right) \qquad \textbf{(2.4)}$$

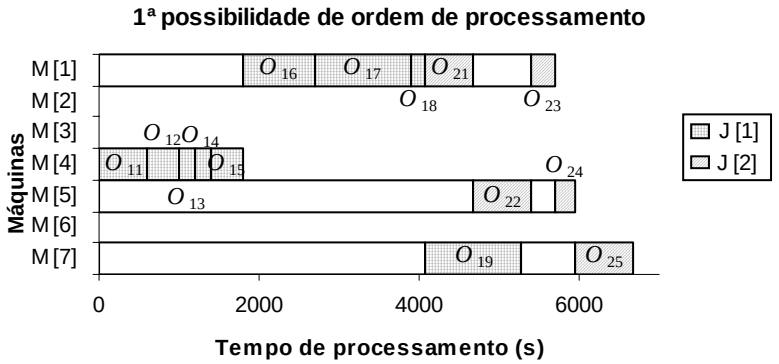
de forma que, para ε^1 (Tabela 2.6), teremos

$$N\pi^g = 5! \times 5! \times 2! \times 2! = 57.600 ,$$

ou seja, para tal possibilidade de configuração de distribuição de operações por máquinas haverão 57.600 possibilidades de ordens de seqüenciamento sendo que nem todas serão consideradas viáveis. Neste ponto é possível verificar o tamanho da complexidade do problema aqui avaliado, mesmo para um exemplo “simples” que considera o processamento de apenas duas tarefas. No ambiente real de produção das empresas de confecção geralmente são processadas em torno de dez tarefas simultaneamente. Na Fig. 2.5 são apresentadas duas possibilidades de ordem de sequenciamento para ε^1, assim como um gráfico de Gantt destas opções relacionando o tempo de processamento.

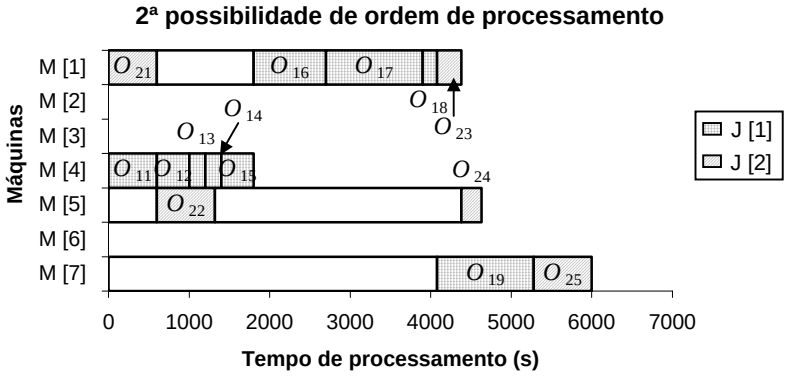
Conforme figurado nestes gráficos, a primeira possibilidade de ordem de processamento (Fig. 2.5(a)) apresenta $C_{\max}\left(\pi^1\right) = 6.670\,s$, já no caso da segunda possibilidade (Fig. 2.5(b)), $C_{\max}\left(\pi^1\right) = 6.000\,s$, o que indica que, entre as duas possibilidade demonstradas, a que corresponde à melhor solução de acordo com a função objetivo adotada (Eq. (2.2)) é a segunda.

$\pi_1^1 = (O_{16}, O_{17}, O_{18}, O_{21}, O_{23})$
$\pi_2^1 = (\ )$
$\pi_3^1 = (\ )$
$\pi_4^1 = (O_{11}, O_{12}, O_{13}, O_{14}, O_{15})$
$\pi_5^1 = \{O_{22}, O_{24}\}$
$\pi_6^1 = (\ )$
$\pi_7^1 = (O_{19}, O_{25})$



(a)

$\pi_1^1 = (O_{21}, O_{16}, O_{17}, O_{18}, O_{23})$
$\pi_2^1 = (\ )$
$\pi_3^1 = (\ )$
$\pi_4^1 = (O_{11}, O_{12}, O_{13}, O_{14}, O_{15})$
$\pi_5^1 = \{O_{22}, O_{24}\}$
$\pi_6^1 = (\ )$
$\pi_7^1 = (O_{19}, O_{25})$

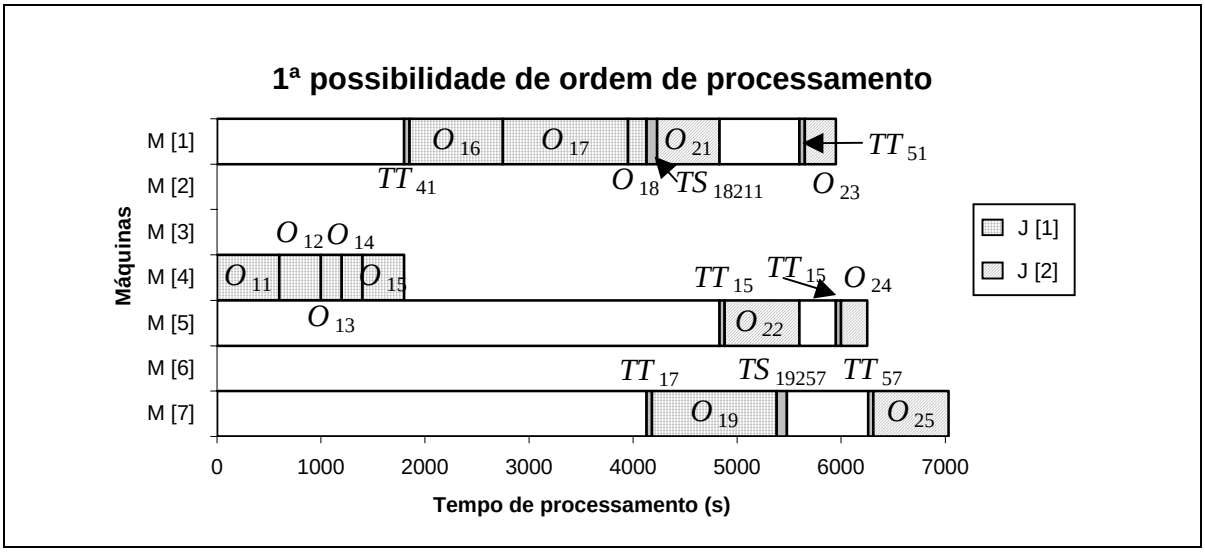


(b)

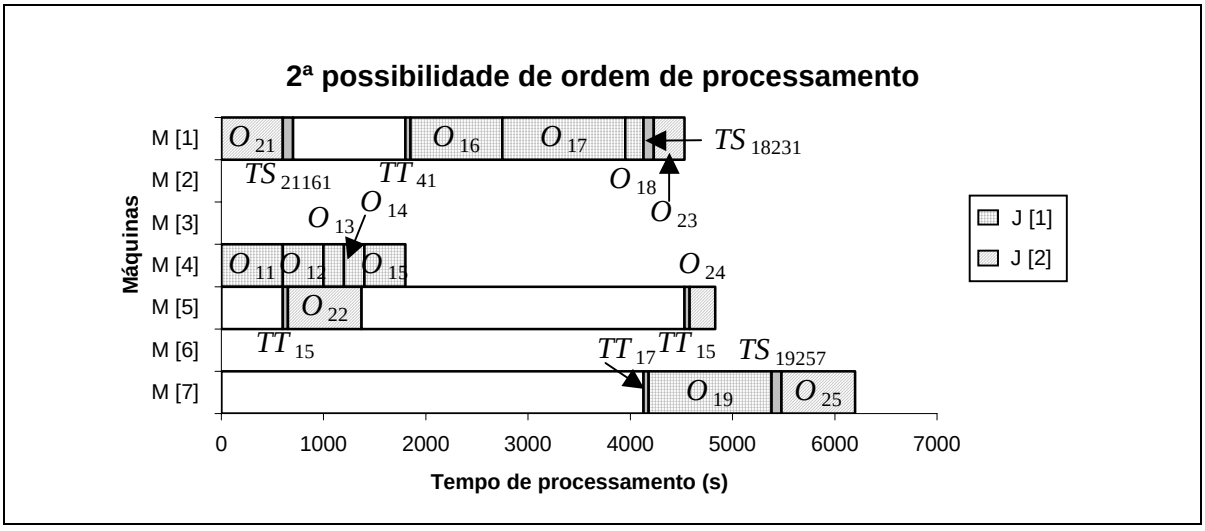
Figura 2.5 – Exemplo de duas possibilidades de ordem de processamento (a) e (b) para ε^1

No caso apresentado a função objetivo foi calculada supondo-se que os tempos de *setup* e de transporte são nulos, *i.e.* $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall i_1, j_1, i_2, j_2, k, k_1, k_2$. No entanto, tomando-se como base as seguintes considerações: (a) o tempo de *setup* ocorrerá e terá valor $TS_{i_1 j_1 i_2 j_2 k} = 100s, \forall i_1, j_1, i_2, j_2, k$ quando duas operações de tarefas distintas forem executadas

seqüencialmente em uma mesma máquina; e (b) o tempo de transporte existirá e terá valor $TT_{k_1k_2} = 50s, \forall\, k_1, k_2$ quando duas operações de uma mesma tarefa executadas seqüencialmente e diretamente dependentes forem alocadas em diferentes máquinas, os gráficos referentes às duas possibilidades de ordem de processamento são representados conforme Fig. 2.6.



(a)



(b)

Figura 2.6 – Primeira (a) e segunda (b) possibilidades de ordem de processamento para ε^1 considerando os tempos de transporte e de *setup*.

Neste caso, para a primeira possibilidade de ordem de processamento $C_{\max}\left(\pi^1\right) = 7.080\,s$, enquanto que para a segunda $C_{\max}\left(\pi^1\right) = 6.200\,s$. Apesar da segunda possibilidade ainda representar a melhor solução, nota-se uma diferença significativa no *makespan*. Pode ser que em diferentes casos esta diferença possa afetar o resultado da melhor solução encontrada.

2.2 – INOVAÇÕES ASSOCIADAS À FORMULAÇÃO APRESENTADA

Dois pontos fundamentais distinguem a formulação aqui exibida da formulação originalmente apresentada por NOWICKI e SMUTNICKI [2005]. Estes pontos e suas respectivas finalidades podem ser relacionados conforme abaixo:

- Foi adicionado o conceito de configuração de distribuição de operações por máquinas à formulação original permitindo que o modelo relacione a possibilidade de execução de uma mesma operação em diferentes máquinas.
- Foram aferidos pesos ao grupo de arcos $R \cup E\left(\pi^g\right)$ permitindo que os tempos de transporte e de *setup* sejam considerados, assim como as relações de dependência que existem entre estes e a ordem de processamento do grupo de tarefas avaliado.

CAPÍTULO 3

3 SOLUÇÃO PARA O PROBLEMA DE PROGRAMAÇÃO DA PRODUÇÃO

Conforme descrito no primeiro capítulo desta dissertação, existe uma gama de métodos que podem ser empregados de forma conjunta ou separadamente na busca de soluções ótimas ou sub-ótimas para os diversos problemas relacionados à programação da produção. O método *Tabu Search* tem se destacado dos demais métodos por ser de fácil aplicação e por apresentar um bom desempenho na maioria dos casos em que tem sido aplicado. De fato, dentro da literatura pesquisada foi o método cuja aplicação apresentou os melhores resultados para problemas do tipo *flow shop* [NOWICKI e SMUTNICKI, 2005]. Assim sendo o método *Tabu Search* foi tomado como base para solução do problema aqui proposto.

3.1 *TABU SEARCH*

O método *Tabu Search* (TS) consiste em um processo iterativo onde, a cada passo, são gerados *N* vizinhos a partir de uma solução atual. A melhor solução vizinha, cujo valor da função objetivo é o melhor dentre as soluções vizinhas, é aceita como a nova solução atual mesmo que seu valor seja pior que a solução anterior, o que evita que o algoritmo se prenda em mínimos locais. Para evitar ciclos na trajetória de busca, uma lista *tabu* deve conter as últimas *t* soluções visitadas as quais não devem ser mais aceitas como soluções durante um período predeterminado (número de iterações) chamado *tenure*, sendo que o valor de *tenure*

corresponde ao valor da variável *t*. Além disso, a melhor solução de toda a execução é armazenada. Na Fig. 3.1 é apresentado um algoritmo indicando os passos básicos do método *Tabu Search*.

Passo 1 *G*erar uma solução inicial $\pi^g(0)$

Calcular $C_{\max}\left(\pi^g(0)\right)$

$C^*_{\max} = C_{\max}\left(\pi^g(0)\right)$

$S^* = \pi^g(0)$, *onde* S^* *representa a permutação* π^g *que gera* $C^*_{\max}$

$\pi^g = \pi^g(0)$

Passo 2 Enquanto *o* critério de parada não for atingido faça:

Gerar vizinhança $N\left(\pi^g\right)$

Calcular $C_{\max}\left(\pi^g\right)$ para cada elemento de $N\left(\pi^g\right)$

Selecionar melhor vizinho S'

Atualizar lista tabu

Se $C\max\left(S'\right)$ é menor que $C\max\left(S^*\right)$ então:

$S^* = S'$

$C^*_{\max} = C'_{\max}$

Rertornar S^*

Figura 3.1– Algoritmo base do método Tabu Search

Uma das principais vantagens do TS é a sua versatilidade. Este método foi projetado de forma a permitir que a cada passo (desde a escolha da solução inicial até a geração de diversos tipos de listas de restrições) diferentes técnicas sejam empregadas no intuito de otimizar a sua aplicação em diferentes tipos de problemas, o que promove uma busca contínua de combinações de métodos que permitam a geração de melhores resultados em tempos menores de processamento computacional. Nas próximas seções tais técnicas serão apresentadas na forma como estas aparecem na ferramenta computacional gerada para solução do problema tratado nesta dissertação.

3.1.1. Solução inicial

A solução inicial, $\pi^g(0)$, é o ponto de partida do método *Tabu Search*. Segundo LAGUNA [1994], não é necessário que $\pi^g(0)$ seja inicialmente viável, contudo, a forma como é traçada gera uma influência direta na velocidade de convergência do programa assim como na sua própria convergência. Conforme descrito pelo autor, tal solução pode ser construída de forma randômica ou através de métodos construtivos, sendo que a segunda forma é mais utilizada por geralmente fornecer um "melhor" ponto de partida (medido pela função objetivo). A vantagem dos métodos randômicos está no fato de que, nestes casos, o programa tende a fugir de mínimos locais com maior facilidade, principalmente quando o programa é reiniciado mais de uma vez com base em diferentes pontos de partida, em contrapartida soluções formadas através de algum modo inteligente tendem a convergir com maior velocidade.

No caso da ferramenta computacional desenvolvida, a solução inicial $\pi^g(0)$ poderá ser fornecida pelo usuário ou formada com base nas seguintes considerações:

- A configuração inicial $\varepsilon^g(0)$ será formada de modo de que $\varepsilon_1^g(0)$ agrupe todas as operações que podem ser processadas em M_1, $\varepsilon_2^g(0)$ agrupe todas as operações que podem ser processadas em M_2, excluindo-se as que podem ser processadas em M_1, $\varepsilon_3^g(0)$ agrupe todas as operações que podem ser processadas em M_3 excluindo-se as que podem ser processadas em M_1 e M_2, e assim sucessivamente.
- A ordem de processamento das operações O_{ij} em cada máquina $\pi_k^g(0)$ respeitará a seguinte ordem de prioridade: a) menor índice *i* e b) menor índice *j*.

Com base no exemplo apresentado no Capítulo 2 com duas tarefas, sete máquinas e quatorze operações, a solução inicial gerada pelo programa aparecerá da seguinte forma:

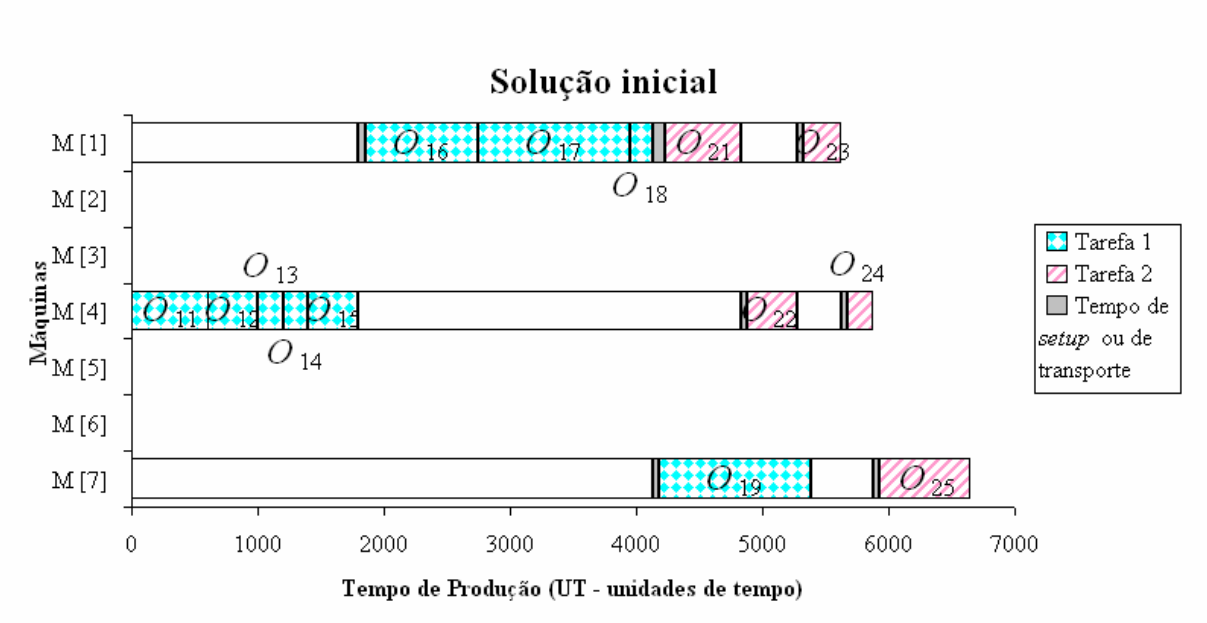


Figura 3.2 – Solução inicial para o exemplo com 2 tarefas, 7 máquinas e 14 operações.

No exemplo aqui considerado e descrito no Capítulo 2 foi usado o segundo (s) como unidade de tempo (UT). A partir deste ponto da dissertação será considerada a unidade de tempo genérica UT.

3.1.2. Vizinhança

O processo de geração de vizinhança constitui peça fundamental do método *Tabu Search* já que a velocidade de processamento de qualquer programa que envolva a aplicação deste método está diretamente relacionada à forma com a qual a vizinhança será formada. NOWICKI e SMUTNICKI [2005] corroboram com tal afirmação explicando que, os processos repetitivos usados na escolha da solução através da exploração da vizinhança são usualmente a parte dos algoritmos baseados em métodos de busca local (como, por exemplo, o *Tabu Search*) que mais consome tempo sendo que, nestes casos, o custo total de cálculo dependerá sempre do número de vizinhos e da quantidade de cálculos por vizinho.

Na geração da vizinhança cada vizinho é formado através de uma função, na literatura conhecida como movimento, que transforma uma solução em outra. O movimento pode ocorrer de diferentes formas, sendo que a mais comum é a troca entre a posição de duas operações consecutivas ou não consecutivas. Dentre todas as formas de movimento encontradas na literatura revisada, aplicadas em problemas de programação da produção do tipo j*ob shop*, a que se mostrou mais eficiente foi a que gerava uma vizinhança substancialmente pequena extraída pela aplicação de movimentos de troca em blocos de operações no caminho crítico conforme estabelecido na referência conforme estabelecido na referência [NOWICKI e SMUTNICKI, 1996].

No caso do problema aqui avaliado, como existe uma relação de precedência entre operações que poderão ser executadas em uma mesma máquina, o movimento de troca acaba por limitar a geração de vizinhança, já que é relativamente difícil encontrar uma nova solução viável aplicando-se esta técnica. Tal fato fez com que a função movimento adotada fosse baseada no critério de reposicionamento, conforme a explicação apresentada a seguir.

Tomando-se como base o critério de reposicionamento, a vizinhança $N\left(\pi^g\right)$ gerada a partir da ordem de processamento π^g é constituída pelo conjunto de todas as ordens de processamento geradas pelo reposicionamento na mesma máquina de uma operação qualquer $\mu\left(\pi^g\right)$, somado ao conjunto de todas as novas permutações geradas em função da troca da máquina responsável pelo processamento de cada operação $\nu\left(\pi^g\right)$.

Para melhor entendimento tomemos como base a solução inicial apresentada na Fig. 3.2. Duas formas de movimento podem ser representadas pelo reposicionamento de O_{21} na primeira posição de M_2 (única posição aceitável já que na configuração atual não foi alocada nenhuma operação em M_2) (Fig. 3.3), e pelo reposicionamento de O_{21} na primeira posição da máquina M_1 (Fig. 3.4).

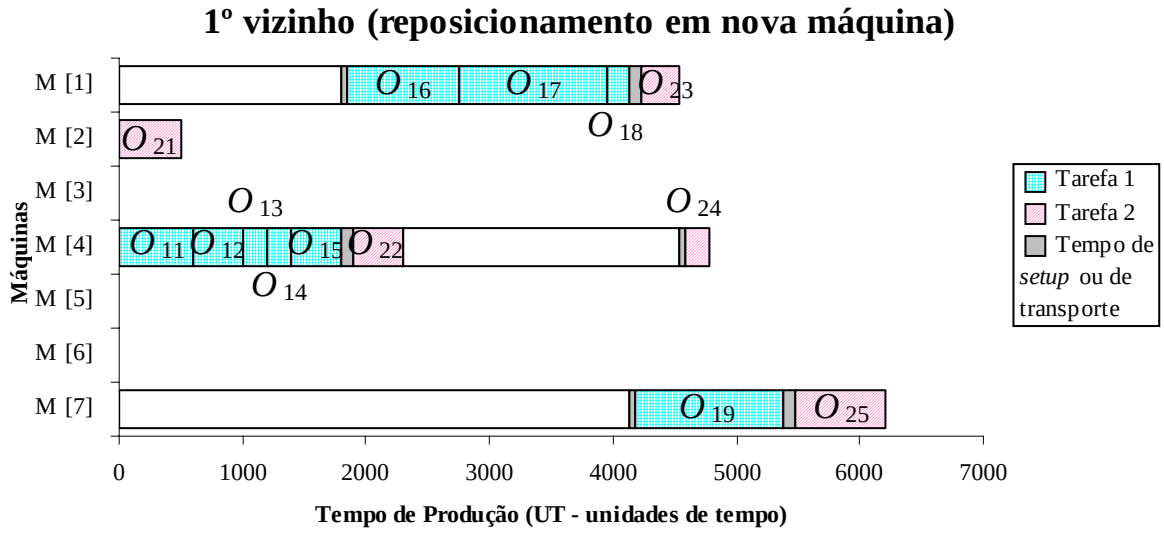


Figura 3.3 – Vizinho gerado pelo reposicionamento de O_{21} em M_2.

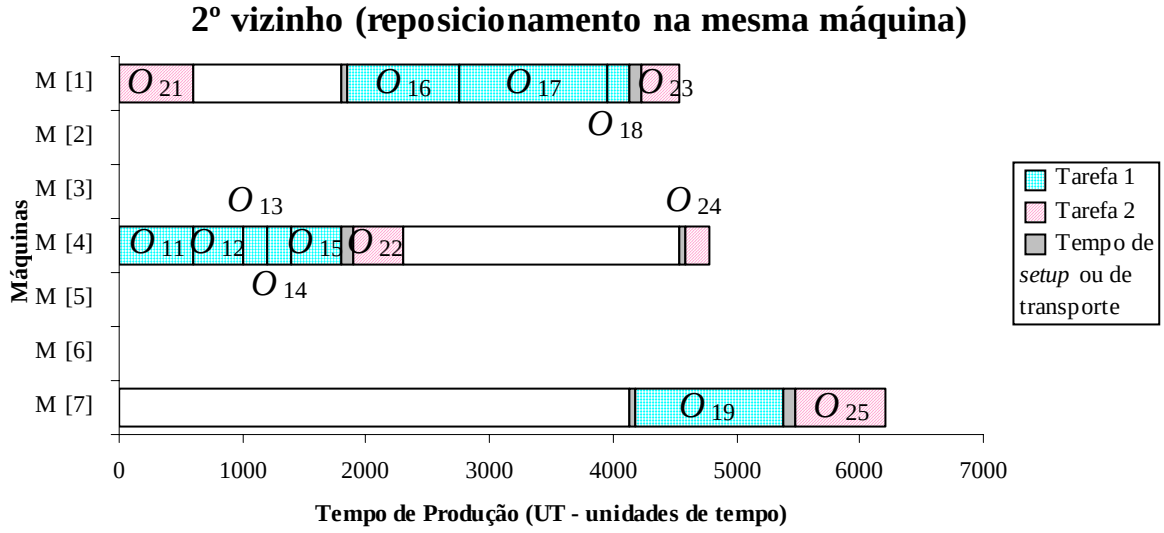


Figura 3.4 – Vizinho gerado pelo reposicionamento de O_{21} em M_1.

Na ferramenta computacional proposta, tal vizinhança, gerada a cada iteração, será uma vizinhança reduzida ao tamanho de *N* soluções vizinhas, sendo que o número de elementos

extraídos do conjunto $\nu(\pi^g)$, $Nel(\nu(\pi^g))$, será dado em função de um valor especificado a priori que define a possibilidade de troca entre máquinas $K(0 \le K \le 1)$, conforme

$$Nel(\nu(\pi^g)) = \text{inteiro mais próximo}(N \times K) \quad \textbf{(3.1)}$$

Já o número de elementos extraídos do conjunto $\mu(\pi^g)$, $Nel(\mu(\pi^g))$, poderá ser obtido pela seguinte equação

$$Nel(\mu(\pi^g)) = N - Nel(\nu(\pi^g)) \quad \textbf{(3.2)}$$

Outro fator que pode ser considerado ou não na ferramenta computacional desenvolvida é a probabilidade de todos os vizinhos serem formados reposicionando-se apenas as operações alocadas na máquina com maior *makespan*. Para tanto foi considerada uma nova variável (ibestmcn) a qual deve ter valor igual à unidade quando se desejar usar esta alternativa e valor nulo em caso contrário.

3.1.3. Melhor vizinho

Pelo método *Tabu Search* o melhor vizinho será a solução encontrada em cada iteração que melhor atender à função objetivo, ou seja, neste caso a solução que gerar menor *makespan*, sendo que, caso esta solução esteja relacionada na lista tabu como movimento proibido ou caso não seja viável, ela não poderá ser aceita, dando lugar a uma segunda melhor solução. Conforme descrito por LAGUNA [1994], critérios de aspiração poderão ser definidos para que, em determinas circunstâncias, as restrições definidas como tabu possam ser violadas, como por exemplo, nos casos em que a solução seja a melhor solução encontrada até o momento. Tal condição é conhecida na literatura como critério de aspiração da melhor solução.

Na ferramenta computacional proposta a melhor solução é encontrada conforme descrito acima, sendo que o critério de aspiração da melhor solução é também considerado.

Nos casos dos dois vizinhos gerados para a solução inicial (Figs. 3.3 e 3.4), $C_{\max}\left(1^{\text{o}}\text{ vizinho}\right) = C_{\max}\left(2^{\text{o}}\text{ vizinho}\right) = 6.200\ s$, de tal forma que a melhor solução entre estas duas seria a que fosse gerada primeiro entre os demais vizinhos.

3.1.4. Lista tabu

Um elemento base fundamental do *Tabu Search* é o uso de memória flexível que, no ponto de vista deste método, engloba os processos de criação e exploração de estruturas para obter vantagens históricas, combinando as atividades de adquirir e beneficiar-se das informações (LAGUNA, 1994). A lista tabu permite que o uso da memória flexível seja explorado de diferentes formas, como, por exemplo, relatando o número de ocorrências de cada tipo de movimento (*frequency memory*), ou conferindo ao último movimento o status de movimento proibido e mantendo este status durante um período predeterminado (*recency memory*).

Na ferramenta computacional desenvolvida a lista tabu consiste em uma matriz de tamanho $N_{opt} \times N_{opt}$, onde N_{opt} é o número total de operações do conjunto de tarefas considerado. Inicialmente todos os elementos desta matriz têm valor nulo. No momento em que o melhor vizinho é escolhido, caso o melhor vizinho tenha sido gerado através do reposicionamento de uma operação (O_{i_1}) na mesma máquina em que estava anteriormente alocada, aos elementos $\left[O_{i_1}\right]\left[O_{i_2}\right]$ e $\left[O_{i_2}\right]\left[O_{i_1}\right]$ da matriz serão conferidos o status de movimento proibido durante um número de iterações pré-definido (tenure), geralmente igual a três, sendo O_{i_2} a operação que, no processo de geração do melhor vizinho, tomou a posição

anterior de O_{i_1} . Neste momento os elementos $\left[O_{i_1}\right]\left[O_{i_2}\right]$ e $\left[O_{i_2}\right]\left[O_{i_1}\right]$ recebem o valor da variável *tenure* sendo que, posteriormente, a cada iteração este mesmo valor será reduzido em uma unidade até o momento em que tais elementos retornam a ter valor nulo deixando de apresentar o status de movimento proibido.

3.1.5. Critérios de parada

Conforme apresentado por NOWICKI e SMUTNICKI [1996], diferentes critérios de parada podem ser aplicados ao método Tabu Search, como por exemplo: (a) quando encontrada uma solução suficientemente próxima da solução esperada ou da solução ótima do problema, caso esta seja conhecida; (b) quando é atingido um número máximo de iterações sem que a solução tenha sido melhorada; e (c) quando o tempo de processamento limite é extrapolado. Na ferramenta computacional desenvolvida nesta dissertação o critério de parada é estabelecido da seguinte forma: são inicialmente determinados um número máximo de iterações (N_{maxite}) e a resposta esperada, caso a resposta esperada seja atingida o programa pára e retorna a solução referente à resposta encontrada, caso a mesma não seja atingida o programa continua até que o número máximo de iterações seja alcançado e retorna a solução referente à melhor resposta obtida.

CAPÍTULO 4

4 RESULTADOS E DISCUSSÕES

4.1 TESTES DE VALIDAÇÃO PARA O PROBLEMA DE PROGRAMAÇÃO DA PRODUÇÃO

Para validação da ferramenta computacional desenvolvida foram utilizados dois problemas testes, sendo o primeiro a avaliação de um caso geral cujas respostas são conhecidas, e o segundo a aplicação de um problema padrão (*benchmark*), conhecido na literatura como problema Π_j. Estes dois problemas serão descritos nas próximas seções.

4.1.1 Problema N x N com seqüência simples

Neste problema é considerado que N tarefas devem ser processadas em N máquinas distintas, sendo que cada tarefa é formada por N operações cuja relação de dependência consiste em uma relação de dependência simples, conforme anteriormente apresentado na Fig. 2.4 (secção 2.1). O valor aferido ao tempo de processamento das operações em cada máquina é de 20 UT (unidades de tempo), independente da operação considerada e da máquina em que esta será executada, sendo que todas as máquinas poderão executar qualquer uma das $N \times N$ operações pertencentes ao conjunto de tarefas que deverão ser processadas no período avaliado. Este problema é aplicado a três diferentes casos conforme relacionado abaixo:

- **1º caso:** os tempos de *setup* e de transporte são ignorados, *i.e.* $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.
- **2º caso:** o tempo de *setup* aparece quando duas operações de tarefas distintas são executadas seqüencialmente em uma mesma máquina, sendo o valor do mesmo igual a 10 UT, independente da máquina e das operações precedentes e subseqüentes;
- **3º caso:** o tempo de transporte é também considerado quando duas operações de uma mesma tarefa forem executadas em diferentes máquinas, caso exista uma restrição de precedência direta entre estas duas operações, ou seja, caso a execução de uma destas operações só possa ser iniciada a partir do momento em que a execução da outra operação tiver sido finalizada. Neste problema o valor do tempo de *setup* permanece igual ao apresentado no segundo caso e o valor do tempo de transporte será igual a 15 UT, independente das máquinas de origem e de destino.

No primeiro caso, como não há penalização (acréscimo de tempo) pela troca de máquina na seqüência de execução das operações de uma mesma tarefa, assim como não há penalização pela troca de tarefas na seqüência de execução das operações de uma determinada máquina, haverá um conjunto de soluções ótimas onde as $n$ésimas operações da ordem de relação de precedência de cada uma das tarefas (O_{in}, $n = 1, \cdots, N$) deverão ser distribuídas entre as $n$ésimas posições na ordem de execução de cada máquina, sendo que as operações de uma mesma tarefa não deverão ser necessariamente processadas na mesma máquina (vide Fig. 4.1). Já no segundo caso deverá ser reservado um tempo para *setup* (durante o qual nenhuma operação poderá ser executada na máquina considerada) sempre que duas operações consecutivas na seqüência de execução das operações de uma determinada máquina forem de tarefas diferentes e no terceiro caso deverá ser reservado também um tempo para transporte (durante o qual nenhuma das duas operações consideradas poderá ser executada) caso exista uma restrição de precedência direta entre duas operações que serão executadas em máquinas

diferentes, sendo que o mesmo tempo reservado para *setup* poderá ser usado para transporte de forma que será considerado o maior tempo exigido. Assim sendo, para que o resultado ótimo seja atingido no segundo e no terceiro caso será necessário que as operações de uma mesma tarefa sejam processadas em uma mesma máquina (vide Fig. 4.2).

Conforme apresentado nestas figuras, independente do caso considerado, para qualquer solução pertencente ao conjunto de soluções ótimas $C_{\max}\left(\pi^{g}\right) = 60\ \text{UT}$. Para este problema teste o valor da solução ótima irá variar diretamente em função de N, conforme

$$C_{\max}\left(\pi^{g}\right) = N \times UT \qquad \textbf{(4.1)}$$

Onde UT é o tempo de processamento de cada operação.

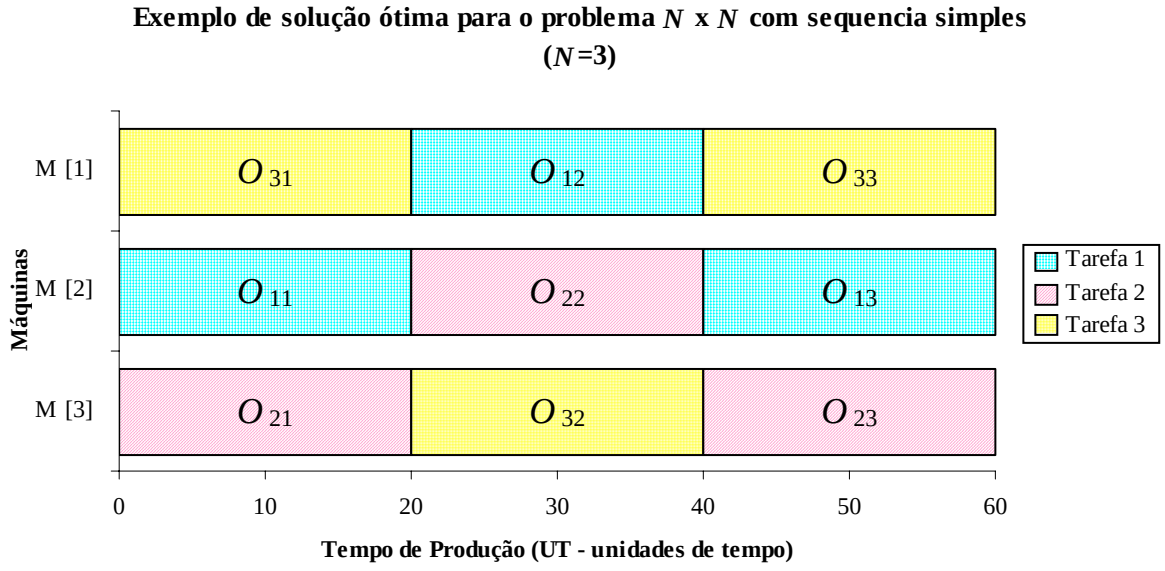


Figura 4.1 – Exemplo de solução ótima para o problema *N* x *N* com seqüência simples (*N* = 3) onde $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

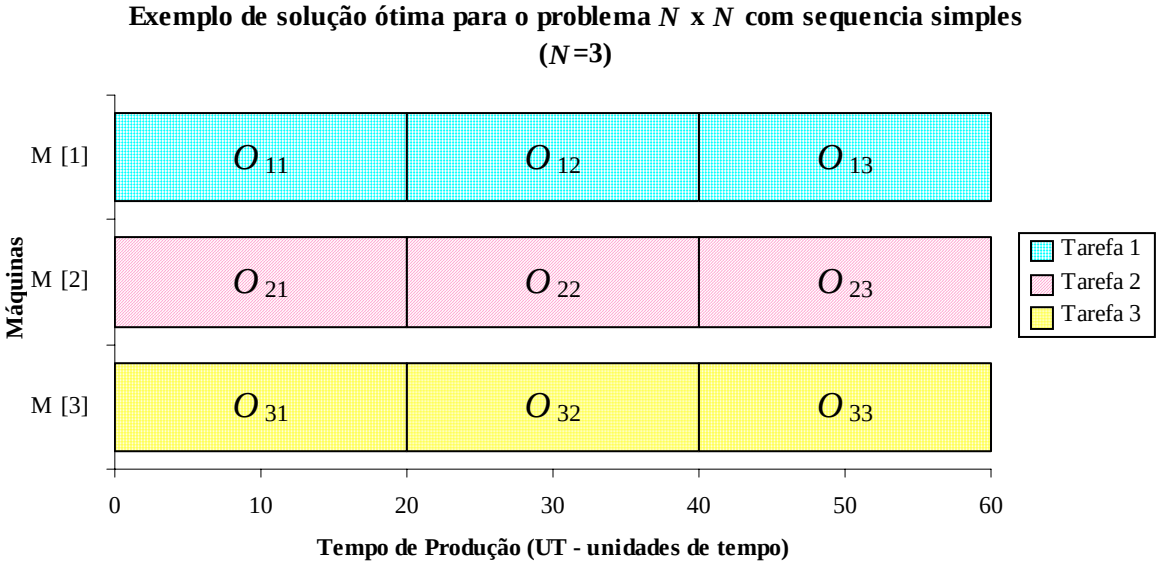


Figura 4.2 – Exemplo de solução ótima para o problema *N* x *N* com seqüência simples (*N* = 3) onde $TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 0$ ou $TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 15$, $\forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

4.1.2 Resultados para o problema N x N com seqüência simples

Este problema tem como finalidade provar que o programa é capaz de chegar aos resultados esperados conforme apresentado nas Figs. 4.1 e 4.2. Assim sendo para geração dos resultados presentes nesta secção foi considerado uma baixo valor para N ($N = 3$) e para cada um dos casos (1º - não considerando os tempos de *setup* e de transporte, 2º - considerando apenas o tempo de *setup*, e 3º - considerando os tempos de *setup* e de transporte) a ferramenta computacional apresentada nesta dissertação foi rodada três vezes obtendo os resultados apresentados nas Figs. 4.3 a 4.11.

Na legenda destas figuras é indicado o tempo de CPU necessário para a obtenção da solução. Estes tempos são relacionados a um computador com processador Pentium 4 com 2.80 GHz.

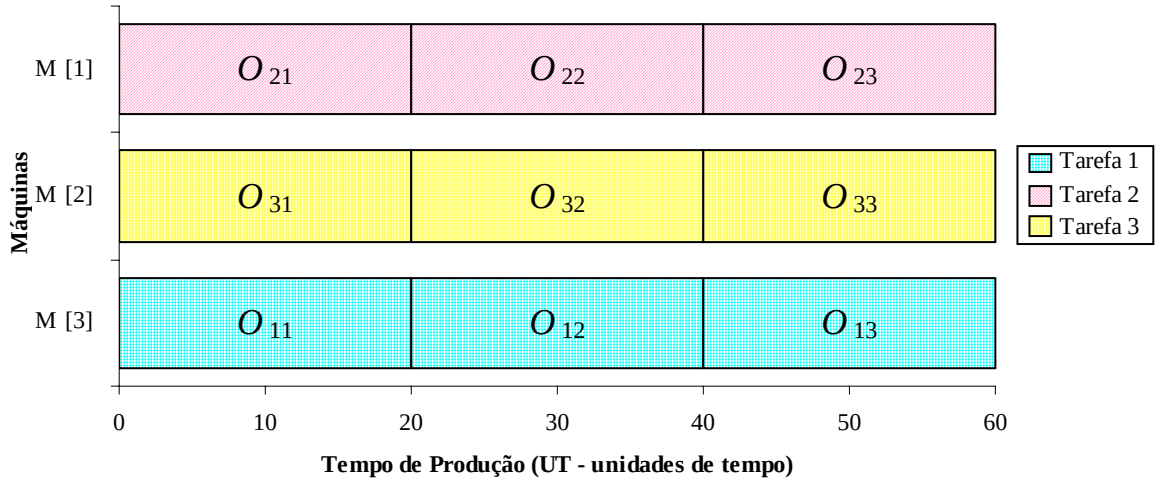


Figura 4.3 – Solução encontrada para o problema $N \times N \left(N = 3 \right)$ com seqüência simples (1ª rodada). Esta solução foi encontrada após 32 iterações, em um tempo de CPU menor que 1 segundo. Caso 1 $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

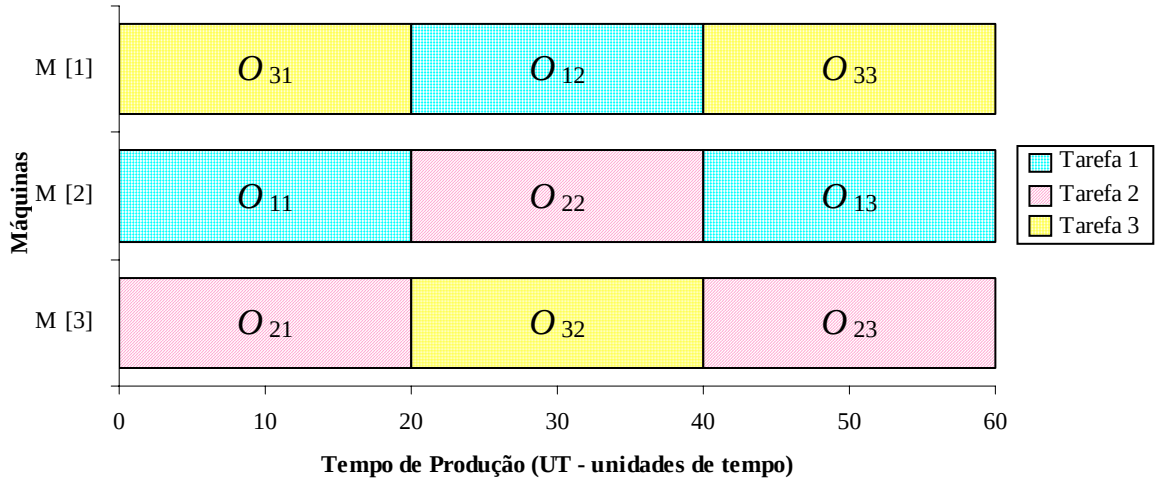


Figura 4.4 – Solução encontrada para o problema $N \times N \left(N = 3 \right)$ com seqüência simples (2ª rodada). Esta solução foi encontrada após 13 iterações, em um tempo de CPU menor que 1 segundo. Caso 1 $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

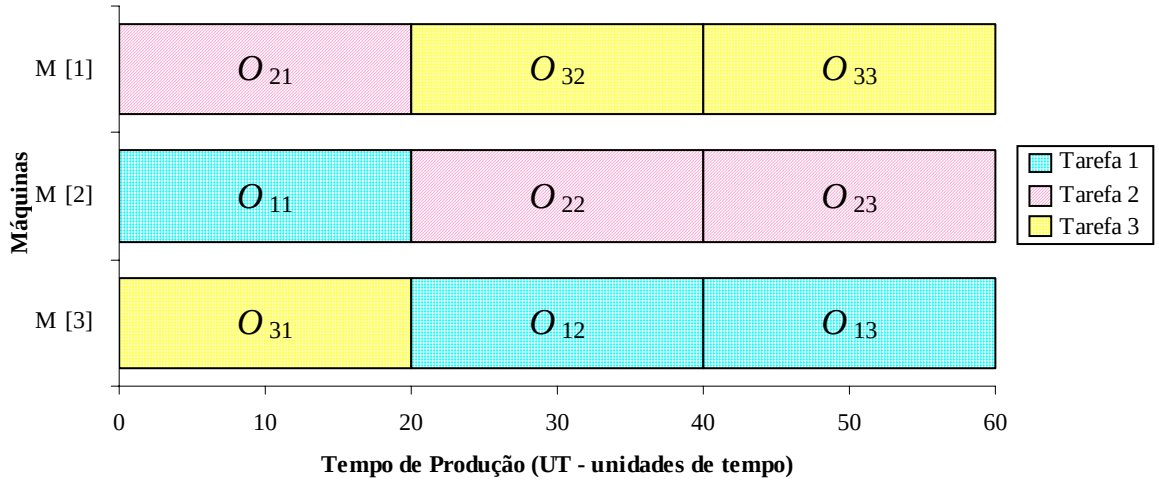


Figura 4.5 – Solução encontrada para o problema $N \times N\left(N=3\right)$ com seqüência simples (3ª rodada). Esta solução foi encontrada após 36 iterações, em um tempo de CPU menor que 1 segundo. Caso 1 $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

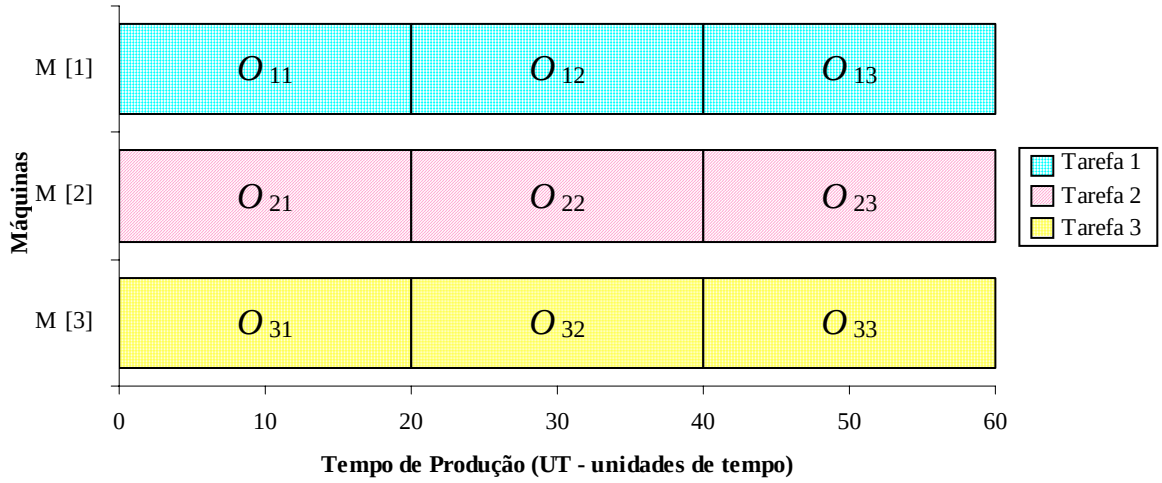


Figura 4.6 – Solução encontrada para o problema $N \times N\left(N=3\right)$ com seqüência simples considerando tempo de *setup* = 10 UT (1ª rodada). Esta solução foi encontrada após 396 iterações, em um tempo de CPU menor que 1 segundo. Caso 2 $TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

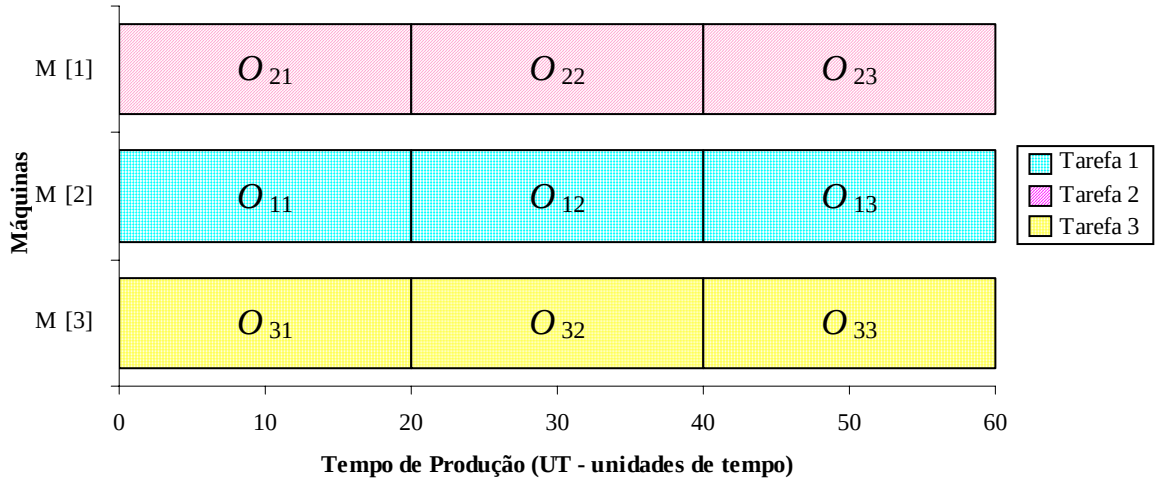


Figura 4.7 – Solução encontrada para o problema $N \times N \left(N = 3\right)$ com seqüência simples considerando tempo de *setup* = 10 UT (2ª rodada). Esta solução foi encontrada após 23 iterações, em um tempo de CPU menor que 1 segundo. Caso 2 $TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 0, \forall i_1, i_2, j_1, j_2, k, k_1, k_2$.

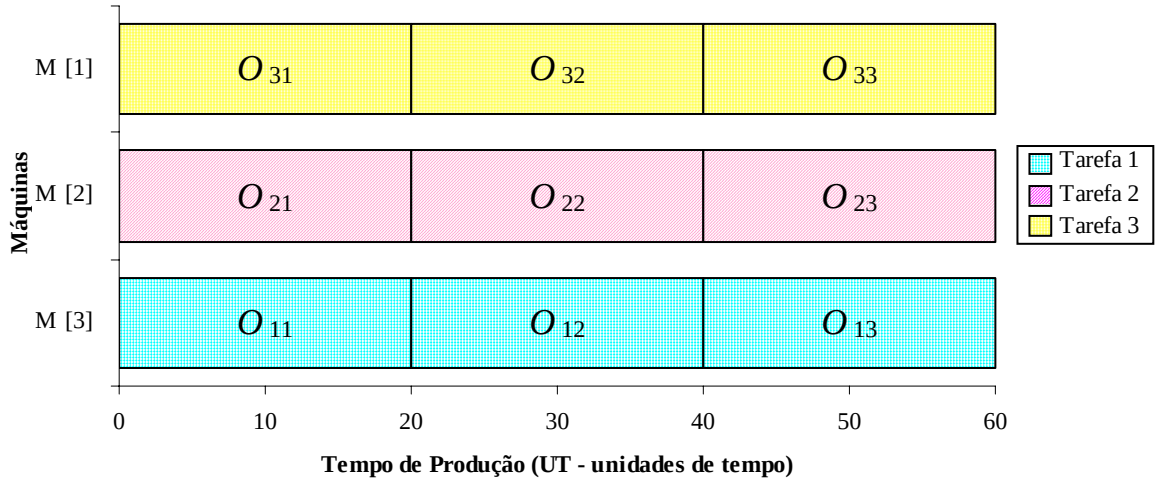


Figura 4.8 – Solução encontrada para o problema $N \times N \left(N = 3\right)$ com seqüência simples considerando tempo de *setup* = 10 UT (3ª rodada). Esta solução foi encontrada após 96 iterações, em um tempo de CPU menor que 1 segundo. Caso 2 $TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 0, \forall i_1, i_2, j_1, j_2, k, k_1, k_2$.

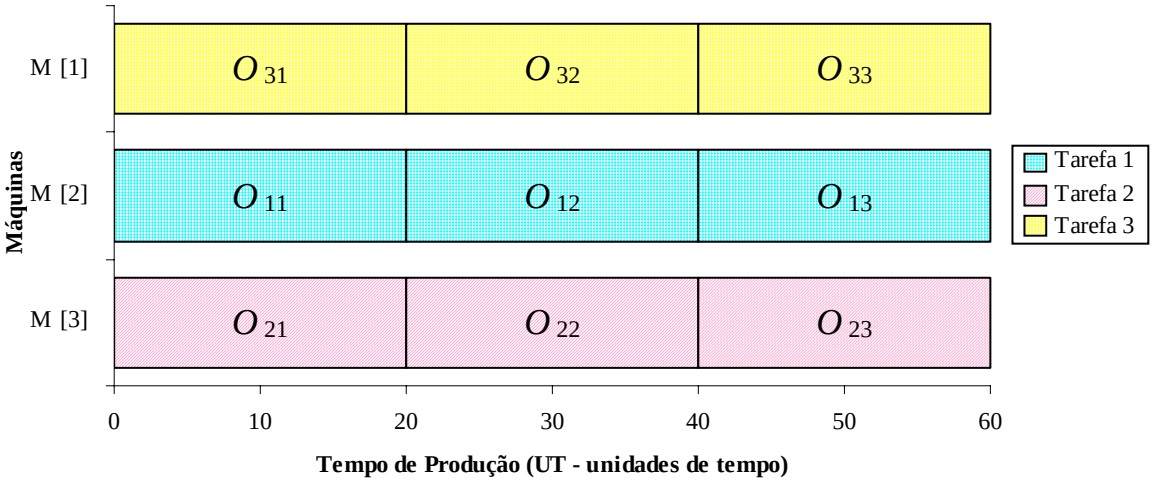


Figura 4.9 – Solução encontrada para o problema $N \times N\left(N=3\right)$ com seqüência simples considerando tempo de *setup* = 10 UT e tempo de transporte = 15 UT (1ª rodada). Esta solução foi encontrada após 56 iterações, em um tempo de CPU menor que 1 segundo. Caso 3
$TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 15, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

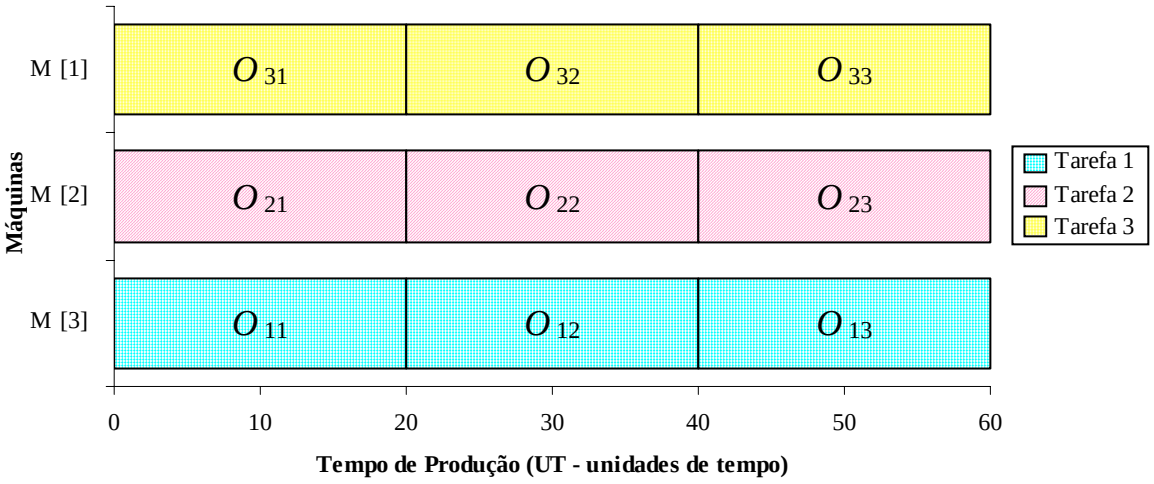


Figura 4.10 – Solução encontrada para o problema $N \times N\left(N=3\right)$ com seqüência simples considerando tempo de *setup* = 10 UT e tempo de transporte = 15 UT (2ª rodada). Esta solução foi encontrada após 168 iterações, em um tempo de CPU menor que 1 segundo. Caso 3
$TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 15, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

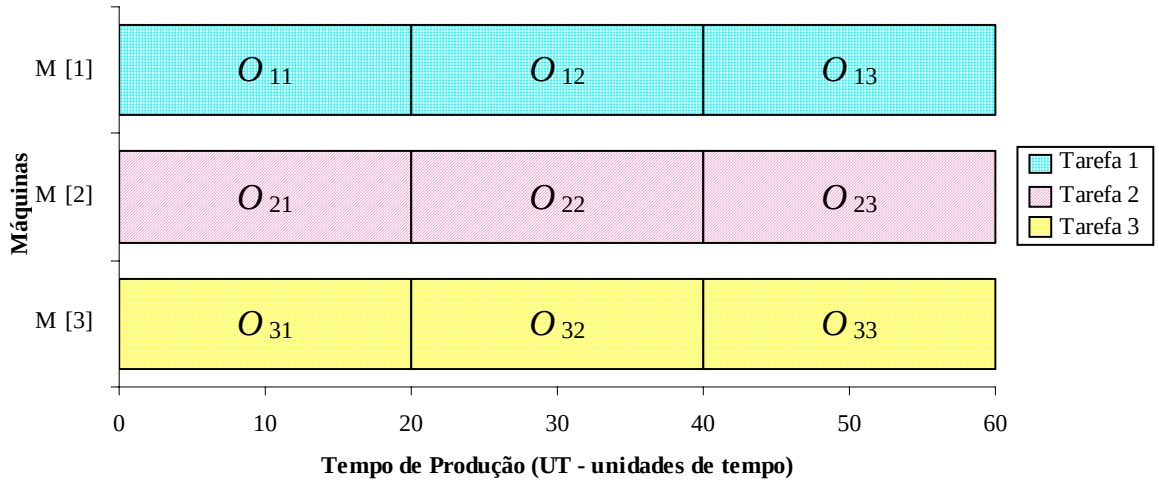


Figura 4.11 – Solução encontrada para o problema $N \times N\,(N = 3)$ com seqüência simples considerando tempo de *setup* = 10 UT e tempo de transporte = 15 UT (3ª rodada). Esta solução foi encontrada após 157 iterações, em um tempo de CPU menor que 1 segundo. Caso 3 $TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 15, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

Os gráficos de Gantt apresentdos nas Figs. 4.3 a 4.11 foram gerados tomando-se como base os seguintes valores para as variáveis de projeto:

- Número de vizinhos (Nngh): 5 (vide definição na seção 3.1.2)
- Possibilidade de troca entre máquinas (MSP): 30 % (vide definição na seção 3.1.2)
- Tenure: 3 (vide definição na seção 3.1)
- Ibestmcn: 0 (vide definição na seção 3.1.2)

Conforme pode ser verificado nas Figs. 4.3 a 4.11 quando $N = 3$ a ferramenta computacional apresentada nesta dissertação é capaz de atingir o resultado esperado (vide Figs. 4.1 e 4.2) para qualquer um dos três casos considerados. O mesmo comportamento foi verificado para $N = 5$ (vide anexo 1). Estes resultados indicam que a ferramenta computacional desenvolvida é capaz de convergir para os diferentes resultados pertencentes ao conjunto de solução ótimas mesmo quando os tempos de *setup* e de transporte são considerados.

4.1.3 Análise de sensibilidade para o problema N x N com seqüência simples

O gráfico apresentado na Fig. 4.12, representa a variação média (para 10 soluções encontradas) do tempo de CPU necessário para que a ferramenta computacional apresentada nesta dissertação encontre a resposta ótima do problema N x N com seqüência simples, para $N = 5$, em função da variação do número de vizinhos ($Nngh$). Neste caso é possível notar que, num intervalo entre cinco e vinte vizinhos, o tempo necessário de CPU diminui (seguindo uma curva de potência) com o aumento de $Nngh$. Isso ocorre porque quando se aumenta o número de vizinhos, apesar de aumentar linearmente o tempo de CPU relativo a cada iteração, o número total de iterações necessárias para que o programa encontre uma resposta ótima é reduzido (seguindo uma curva de potência) compensando então o aumento do tempo de CPU relativo a cada iteração (vide Figs. 4.13 e 4.14).

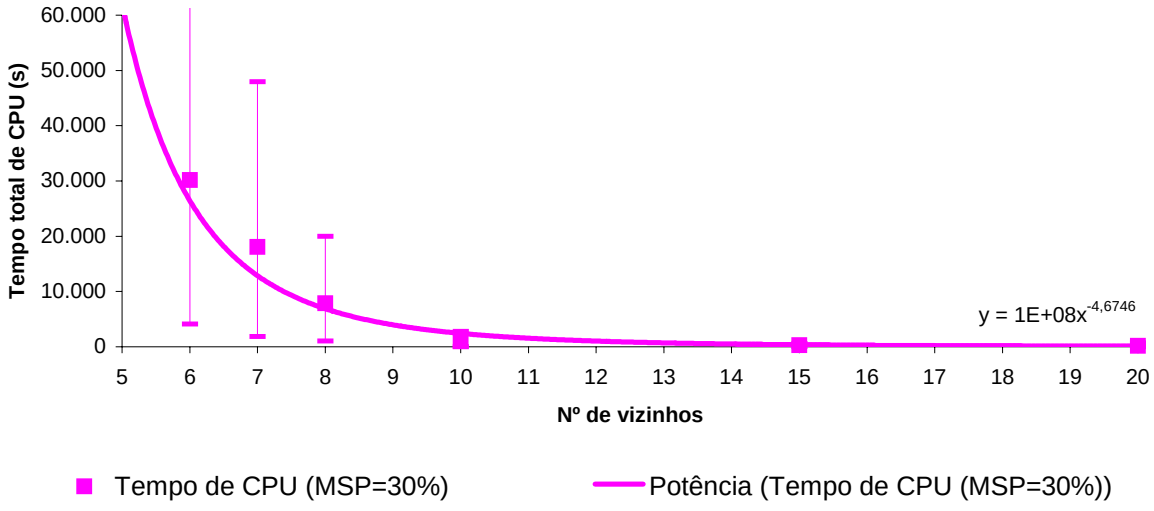


Figura 4.12 – Análise de sensibilidade da variação do tempo de CPU em função do número de vizinhos para o problema *N* x *N* (onde *N*=5) com seqüência simples, considerando o valor de *MSP* (percentual de troca entre máquinas) igual a 30%. $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

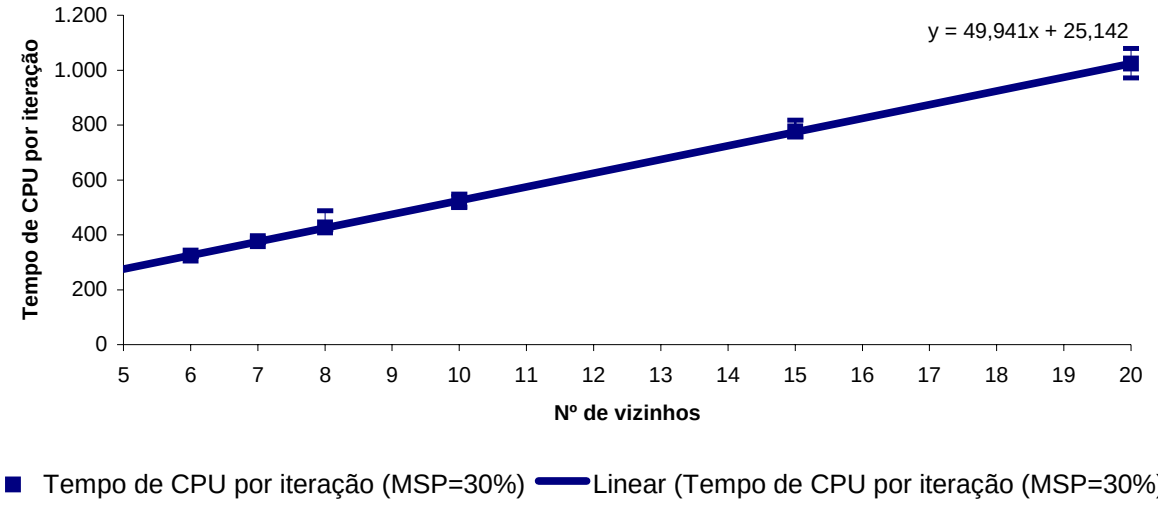


Figura 4.13 – Análise de sensibilidade da variação do tempo de CPU por iteração em função do número de vizinhos para o problema *N* x *N* (onde *N*=5) com seqüência simples, considerando o valor de *MSP* igual a 30%. $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$ **.**

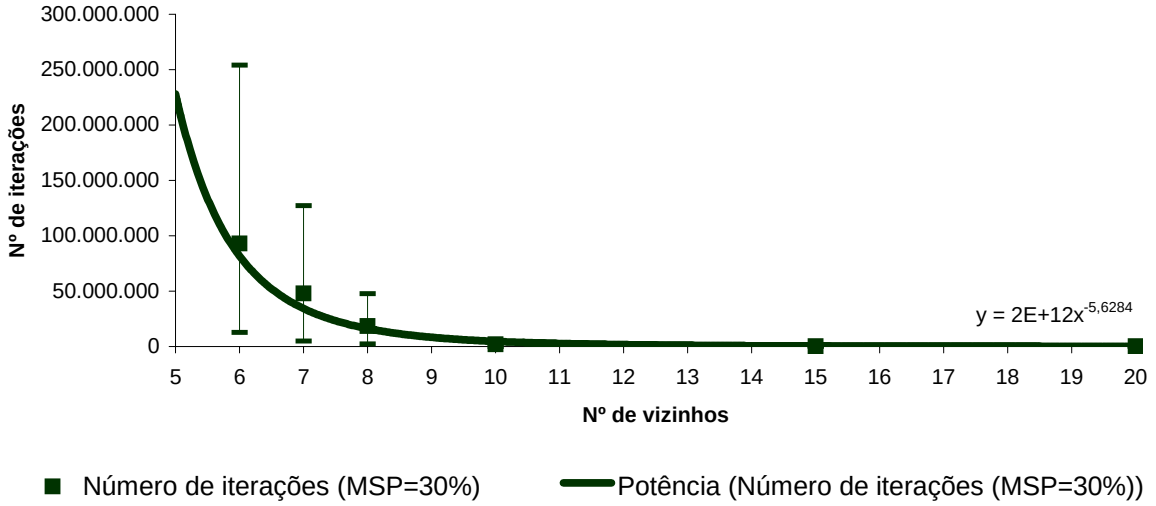


Figura 4.14 – Análise de sensibilidade do número de total de iterações em função do número de vizinhos para o problema *N* x *N* (onde *N*=5) com seqüência simples, considerando *MSP* igual a 30%. $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$ **.**

O mesmo pode ser verificado aumentado-se o valor do percentual de troca entre máquinas (*MSP*) de 30% para 90%, sendo que, conforme apresentado na Fig. 4.15, no caso de *MSP* = 90%, para um mesmo número de vizinhos o tempo de CPU necessário para que o programa encontre uma resposta ótima é consideravelmente menor.

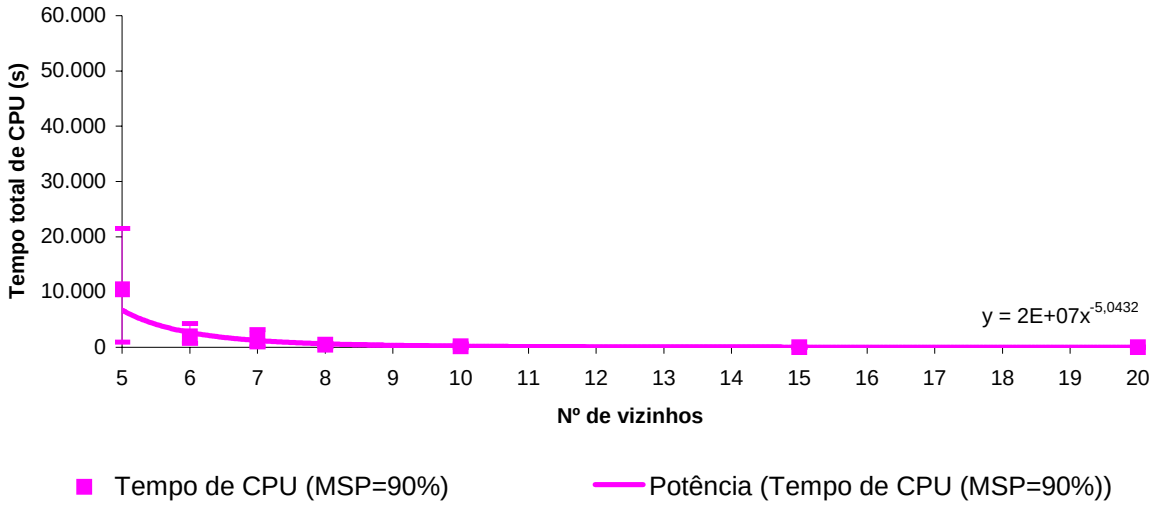


Figura 4.15 – Análise de sensibilidade da variação do tempo de CPU em função do número de vizinhos para o problema *N* x *N* (onde *N*=5) com seqüência simples, considerando o valor de MSP (percentual de troca entre máquinas) igual a 90%. $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

Na Fig. 4.16 o gráfico apresentado na Fig. 4.15 é reproduzido alterando-se a escala do eixo relacionado ao tempo de CPU de tal forma que seja possível analisar um período de uma hora (3600 segundos). Nesta figura é possível verificar que a partir de um determinado número de vizinhos a queda do tempo de CPU torna-se desprezível (< 20 segundos). Já na Fig. 4.17 verifica-se que durante um longo intervalo para o número de vizinhos o tempo de CPU necessário para que o programa encontre uma resposta ótima é um tempo reduzido (< 10 segundos), sendo que após um determinado número de vizinhos este valor retorna a subir. Este comportamento implica em uma forte dificuldade para se obter um valor para o número de vizinhos que possa ser considerado como ótimo, ou seja, que seja capaz de reduzir ao máximo o tempo de CPU necessário para que a ferramenta computacional desenvolvida encontre a resposta esperada. Contudo deve-se levar em conta que dentro de um intervalo relativamente grande de número de vizinhos qualquer valor pode ser escolhido sem que isso se reflita em alguma alteração considerável no tempo de CPU.

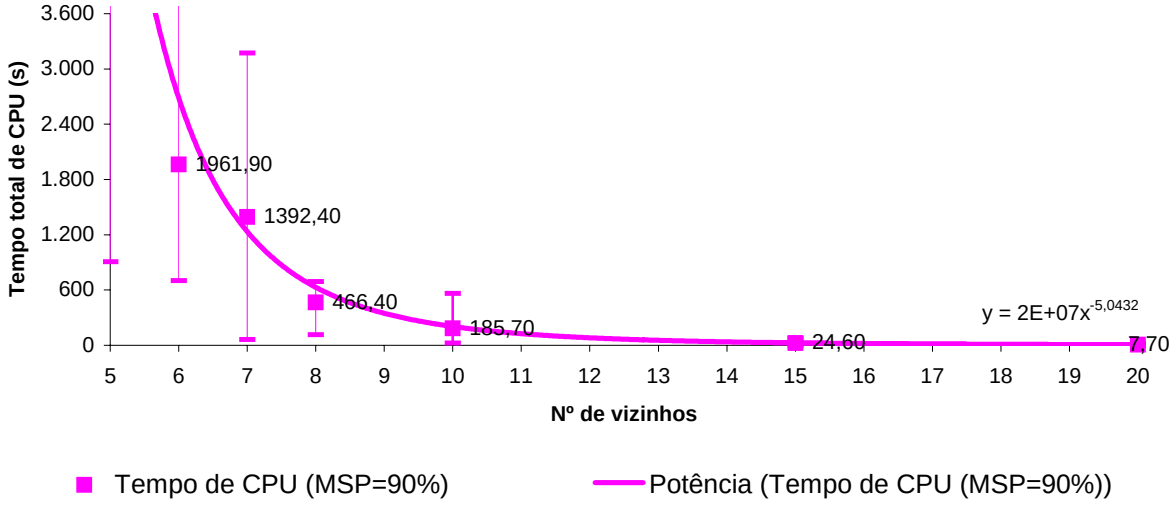


Figura 4.16 – Análise de sensibilidade da variação do tempo de CPU em função do número de vizinhos para o problema *N* x *N* (onde *N*=5) com seqüência simples, considerando o valor de MSP (percentual de troca entre máquinas) igual a 90% (Escala do eixo relativo ao tempo de CPU aumentada).
$TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

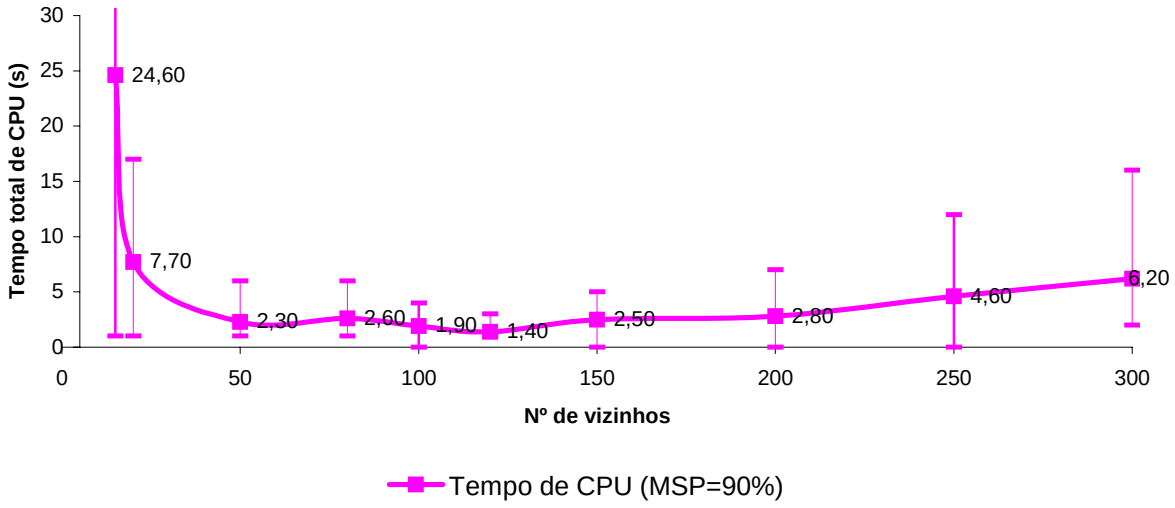


Figura 4.17 – Análise de sensibilidade da variação do tempo de CPU em função do número de vizinhos para o problema *N* x *N* (onde *N*=5) com seqüência simples, considerando o valor de MSP igual a 90% (Escala do eixo relativo ao tempo de CPU aumentada e intervalo de número de vizinhos ampliado).
$TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

Cada ponto nas figuras 4.12 a 4.17 representa uma média das soluções obtidas em 10 corridas (rodadas) do programa, e as barras representam os intervalos ao longo dos quais as soluções se distribuem, sendo indicados os menores e os maiores valores obtidos.

4.1.4 Problema Π_j

Com base nas definições apresentadas no primeiro capítulo, o problema Π_j pode ser descrito como um problema do tipo *job shop* onde cada operação poderá ser processada em apenas uma máquina. Assim sendo, este problema se resume à definição da ordem de sequenciamento do conjunto de operações que deverá ser processado em cada uma das máquinas consideradas. A relação de máquinas que devem executar cada operação e o tempo de execução destas operações nas respectivas máquinas são relacionados conforme a Tabela 4.1. Mais detalhes sobre o problema tratado nesta secção podem ser encontrados no trabalho publicado por JAIN e MEERAN [1999]. É importante ressaltar que este problema é um caso particular do problema mais geral tratado nesta dissertação. Ele é, porém, um problema padrão (*benchmark*) aqui utilizado para validação do algoritmo implementado.

4.1.5 Análise de sensibilidade para o problema Π_j

Os gráficos apresentados nas Figs. 4.18 e 4.19 foram gerados mantendo-se o número total de iterações constante (com valores de 100, 1.000 e 10.000), e variando-se o número de vizinhos, sendo que foram assumidos os seguintes valores para as variáveis de projeto:

- Número de vizinhos (Nngh): variando entre 5 e 500 (vide definição na seção 3.1.2)
- Possibilidade de troca entre máquinas (MSP): 90 % (vide definição na seção 3.1.2)
- Tenure: 3 (vide definição na seção 3.1)
- Ibestmcn: 0 (vide definição na seção 3.1.2)

Tabela 4.1– Problema Π_j para dez máquinas e dez tarefas com dez operações cada [JAIN e MEERAN, 1999]. Os valores de $T_{ij,k}$ são expressos em UT.

J_i	J_1		J_2		J_3		J_4		J_5		J_6		J_7		J_8		J_9		J_{10}	
O_{ij}	M_k	$T_{ij,k}$	M_k	$T_{ij,k}$	M_k	$T_{ij,k}$	M_k	$T_{ij,k}$	M_k	$T_{ij,k}$	M_k	$T_{ij,k}$	M_k	$T_{ij,k}$	M_k	$T_{ij,k}$	M_k	$T_{ij,k}$	M_k	$T_{ij,k}$
O_{i1}	1	29	1	43	2	91	2	81	3	14	3	84	2	46	3	31	1	76	2	85
O_{i2}	2	78	3	90	1	85	3	95	1	6	2	2	1	37	1	86	2	69	1	13
O_{i3}	3	9	5	75	4	39	1	71	2	22	6	52	4	61	2	46	4	76	3	61
O_{i4}	4	36	10	11	3	74	5	99	6	61	4	95	3	13	6	74	6	51	7	7
O_{i5}	5	49	4	69	9	90	7	9	4	26	9	48	7	32	5	32	3	85	9	64
O_{i6}	6	11	2	28	6	10	9	52	5	69	10	72	6	21	7	88	10	11	10	76
O_{i7}	7	62	7	46	8	12	8	85	9	21	1	47	10	32	9	19	7	40	6	47
O_{i8}	8	56	6	46	7	89	4	98	8	49	7	65	9	89	10	48	8	89	4	52
O_{i9}	9	44	8	72	10	45	10	22	10	72	5	6	8	30	8	36	5	26	5	90
O_{i10}	10	21	9	30	5	33	6	43	7	53	8	25	5	55	4	79	9	74	8	45

Cada ponto presente nestes gráficos consiste na média dos resultados obtidos em dez corridas da ferramenta computacional. Vale ressaltar que neste problema os tempo de *setup* e de transporte não são considerados, *i.e.* $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall i_1, j_1, i_2, j_2, k, k_1, k_2$.

No primeiro gráfico, onde é representada a variação do *makespan* encontrado, é possível notar que a partir de um determinado valor para o número de vizinhos, que aqui chamaremos de valor limite de vizinhos, a redução do *makespan* deixa de ser significativa, sendo que à medida em que se aumenta o número de iterações este valor é reduzido, contudo esta redução também varia deixando de ser significativa a partir de um determinado número de iterações. Outra observação que pode ser extraída deste mesmo gráfico é que há um intervalo de valores onde o aumento do número de iterações reduz de forma significativa o *makespan*, sendo que, à medida em que o número de iterações é aumentado, o ganho na redução do *makespan* também vai decrescendo.

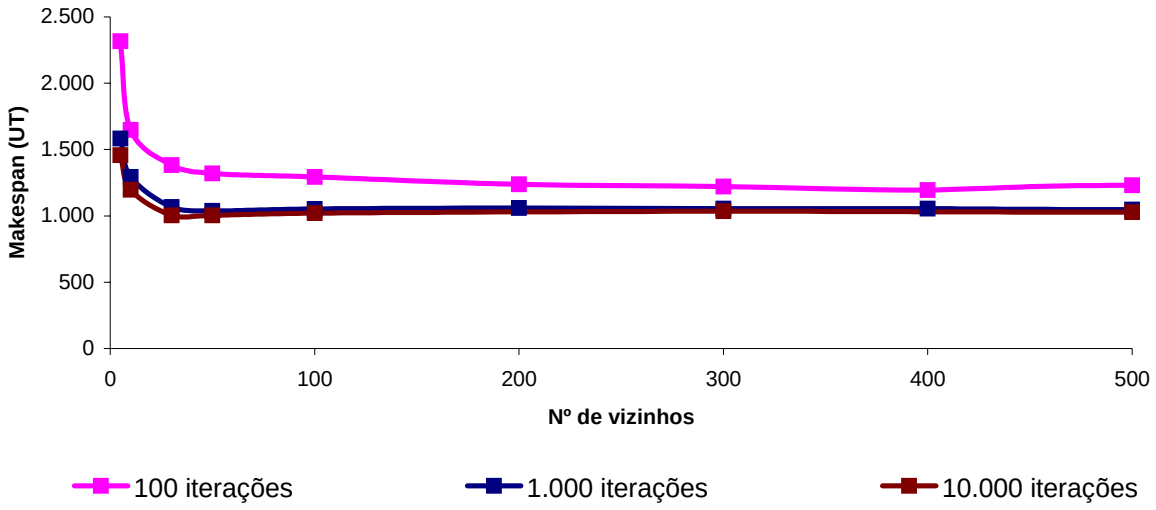


Figura 4.18 – Análise de sensibilidade da variação do makespan em função do número de vizinhos para o problema Π_j, considerando o valor de MSP igual a 90%.

Já o tempo de CPU cresce linearmente com o aumento do número de vizinhos, sendo que para um maior número de iterações a variação do tempo de CPU torna-se mais sensível ao aumento deste número de vizinhos (Fig. 4.19).

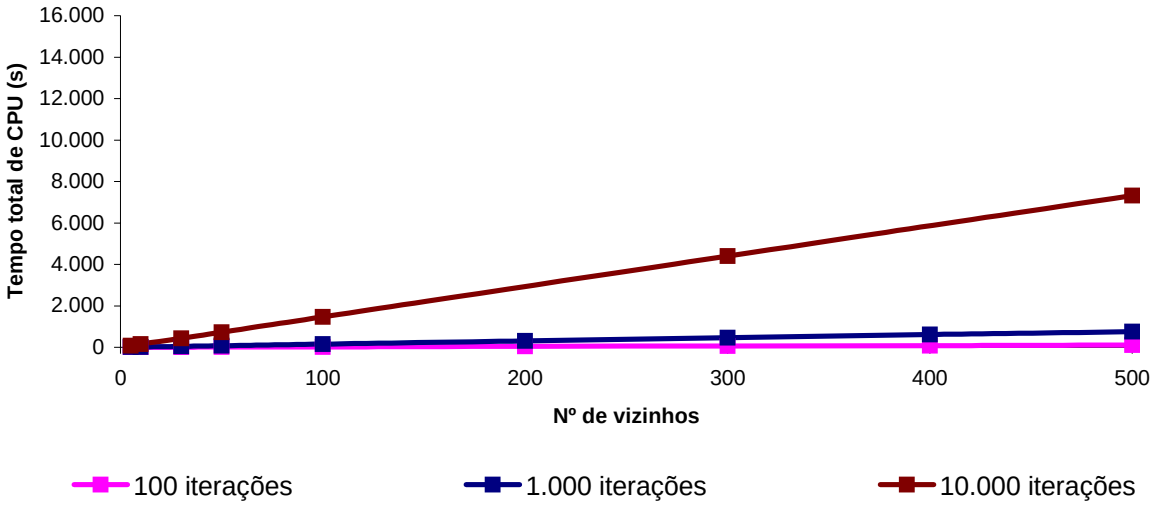


Figura 4.19 – Análise de sensibilidade da variação do tempo de CPU em função do número de vizinhos para o problema Π_j, considerando o valor de MSP igual a 90%.

Conforme pode ser verificado no gráfico apresentado na Fig. 4.20, os desvios padrões dos resultados encontrados (*makespans*) são valores baixos (< 10%). Tal resultado demonstra que ocorre uma boa convergência para os resultados apresentados.

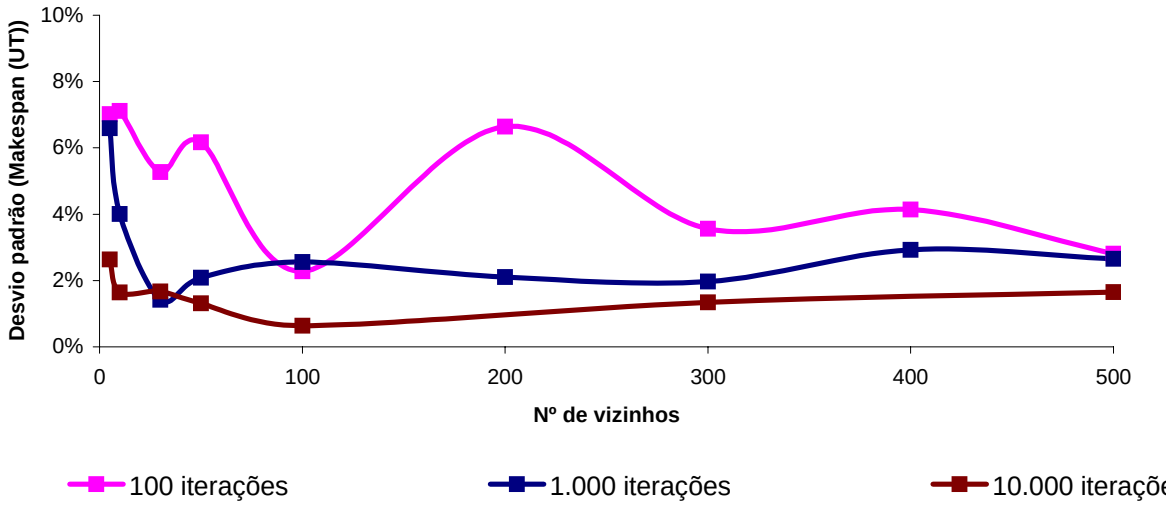


Figura 4.20 – Análise de sensibilidade da variação do desvio padrão em função do número de vizinhos para o problema Π_j, considerando o valor de MSP igual a 90%.

4.1.6 Resultados para o problema Π_j

Com base nos gráficos apresentados na secção 4.1.6 as seguintes variáveis de projeto foram consideradas:

- Número de vizinhos (Nngh): 30 (vide definição na seção 3.1.2)
- Possibilidade de troca entre máquinas (MSP): 0 % (vide definição na seção 3.1.2)
- Tenure: 3 (vide definição na seção 3.1)
- Ibestmcn: 0 (vide definição na seção 3.1.2)

A ferramenta computacional foi testada duas vezes fixando-se o número de iterações respectivamente em 100.000 e 300.000. Os resultados obtidos são apresentados na Tabela 4.2.

Tabela 4.2 – Resultado encontrado para o problema Π_j

	Nº de Iterações	***Makespan* (UT)**	**Tempo de CPU**
1ª rodada	100.000	981	4.140 s
2ª rodada	300.000	967	12.544 s

O problema Π_j apresentado nesta secção possui $C_{\max}\left(\textit{solução ótima}\right) = 930$ UT. Um gráfico de Gantt apresentando uma solução ótima para este problema pode ser encontrado no trabalho publicado por JAIN e MEERAN [1999].

Pela Tabela 4.2 é possível notar que a ferramenta computacional apresentada nesta dissertação é capaz de chegar a resultados próximos do resultado ótimo, sendo que seria provavelmente necessário um tempo de CPU muito alto para que esta ferramenta pudesse chegar ao resultado apresentado pelos autores. Este fato se explica porque a ferramenta computacional aqui apresentada foi desenvolvida para casos mais gerais, de forma que as restrições definidas por este problema exigem um grande esforço computacional desta ferramenta tornando-a um pouco menos eficiente para solução do mesmo. Outra questão que deve ser considerada é que, no caso do problema ocorrente no setor de confecções de Nova Friburgo, não existe a necessidade de se encontrar respostas ótimas uma vez que respostas

consideradas sub-ótimas já representariam um grande ganho na capacidade produtiva das empresas do setor estudado.

4.2 EXEMPLOS PRÁTICOS PARA O PROBLEMA DE PROGRAMAÇÃO DA PRODUÇÃO

Nesta seção são apresentados dois exemplos práticos de problemas similares ao que ocorre no ambiente de produção das empresas de confecção, sendo um deles o caso com 2 tarefas, 7 máquinas e 14 operações apresentado no capítulo dois desta dissertação, e o outro a retratação de um caso real observado em um dia de produção de uma determinada empresa do setor de confecções de Nova friburgo.

4.2.1 Exemplo apresentado no Capítulo 2 com 2 tarefas, 7 máquinas e 14 operações.

O problema apresentado no Capítulo 2 desta dissertação contém um conjunto de soluções ótimas, onde $C_{\max}\left(\textit{solução ótima}\right) = 3.720$ UT , sendo representada na Fig. 4.21 uma destas soluções. Tal solução pode ser considerada ótima pela constatação de que, partindo-se de uma configuração que resulte nesta solução, as operações pertencentes ao caminho crítico não podem ser reposicionadas de forma a melhorar a solução obtida.

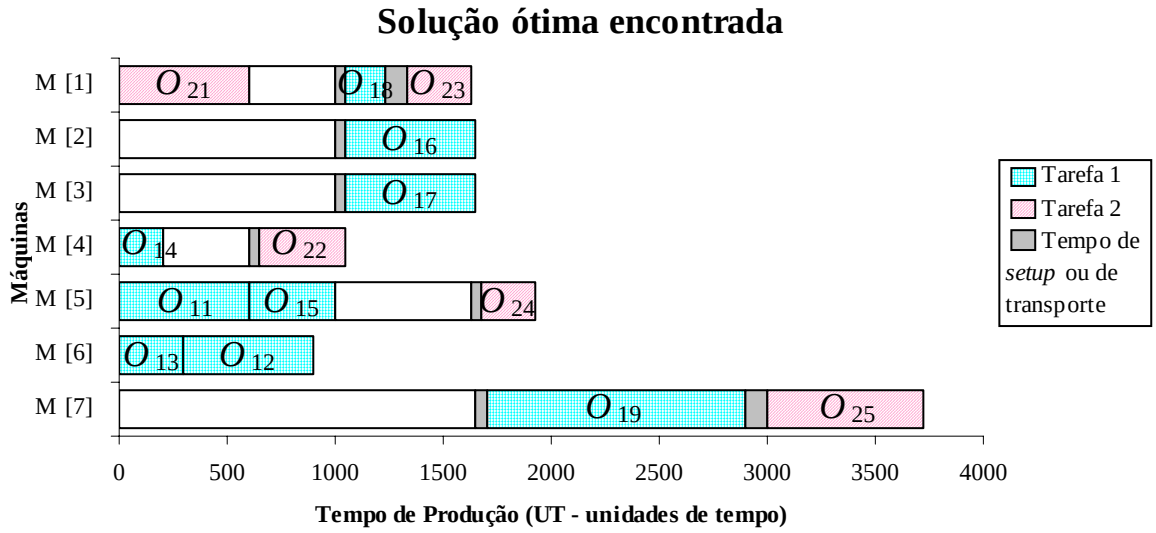


Figura 4.21 – Solução ótima encontrada pela ferramenta computacional apresentada nesta dissertação para o exemplo com 2 tarefas, 7 máquinas e 14 operações.

A Tabela 4.3 apresenta o número de iterações e o tempo de CPU necessários para que, em cada uma das dez vezes que o programa foi rodado consecutivamente, o programa atingisse a solução esperada ($C_{\max}\left(\textit{solução ótima}\right) = 3.720$ UT), sendo que esta tabela foi gerada tomando-se como base os seguintes valores para as variáveis de projeto:

- Número de vizinhos (Nngh): 5 (vide definição na seção 3.1.2)
- Possibilidade de troca entre máquinas (MSP): 50 % (vide definição na seção 3.1.2)
- Tenure: 3 (vide definição na seção 3.1)
- Ibestmcn: 0 (vide definição na seção 3.1.2)
- Resultado esperado: 3.720 UT

Tabela 4.3 – Número total de iterações e tempo de CPU exigidos em cada rodada para obtenção do resultado esperado para o exemplo com 2 tarefas, 7 máquinas e 14 operações.

Rodada	**1ª**	**2ª**	**3ª**	**4ª**	**5ª**	**6ª**	**7ª**	**8ª**	**9ª**	**10ª**
Nº de iterações	64	25	185	575	355	399	157	278	250	170
Tempo de CPU	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s

Conforme apresentado nesta tabela, no caso de problemas pequenos, como o apresentado nesta secção, o programa computacional desenvolvido é capaz de chegar a resultados ótimos exigindo um baixo tempo de CPU (< 1 s), mesmo quando há restrições

relacionadas à possibilidade de processamentos das operações em cada máquina e as restrições de sequenciamento de execução das operações de cada tarefa possibilitam que diferentes operações de uma mesma tarefa sejam executadas ao mesmo tempo.

Os gráficos de Gantt referentes às duas primeiras repostas encontradas em cada uma das duas primeiras rodadas encontram-se no Anexo 2 desta dissertação. Nestes gráficos nota-se a variedade de soluções ótimas encontradas pela ferramenta computacional aqui apresentada o que comprova que esta ferramenta é capaz de chegar a diferentes resultados percorrendo diferentes áreas do conjunto solução.

4.2.2 Exemplo com 7 tarefas, 20 máquinas e 72 operações elaborado com base em uma situação real observada em um dia de trabalho de uma determinada empresa do setor de confecções de Nova Friburgo

Neste problema sete tarefas (vide Tabela 4.4) devem ser processadas em 18 máquinas distintas mais dois funcionários que operam sem máquinas (vide Tabela 4.5), sendo as 18 máquinas dispostas conforme apresentado na Fig. 4.22. O tempo de transporte entre estas 18 máquinas varia em função da distância entre as mesmas sendo que este tempo deverá ser considerado quando duas operações diretamente dependentes forem executadas em máquinas diferentes (Fig. 4.22 e Tabela 4.6). É importante ressaltar que os funcionário referenciados como M_{19} e M_{20} não aparecem na Fig. 4.22 e nem na Tabela 4.6. Isso ocorre porque o tempo de transporte entre estes e entre estes e as máquinas tem valor nulo. Já o tempo de *setup* segue o mesmo padrão apresentado no caso anterior conforme sugerido para o exemplo com 2 tarefas, 7 máquinas 14 operações descrito no Capítulo 2 desta dissertação.

Tabela 4.4 – Relação de tarefas para o problema com 7 tarefas, 20 máquinas e 72 operações

Tarefa	Modelo	Quantidade	Cor
J_1	Duda	40	Preta
J_2	Sabrina	36	Lucivita
J_3	Pink	280	Verde
J_4	Calça Cinto	64	Amarela
J_5	Camisete Antonela	300	Branca
J_6	Sandy	300	Beje
J_7	Camisete Renda	300	Azul

Tabela 4.5 – Relação de máquinas para problema com 7 tarefas, 20 máquinas e 72 operações

Máquinas	Descrição
M_1 a M_7	Overloque
M_8 a M_{11}	Colarete
M_{12} a M_{14}	Ziguezague três pontos
M_{15}	Interloque
M_{16} e M_{17}	Automática
M_{18}	Automática Lisa
M_{19} e M_{20}	Funcionário (limpeza e acabamento)

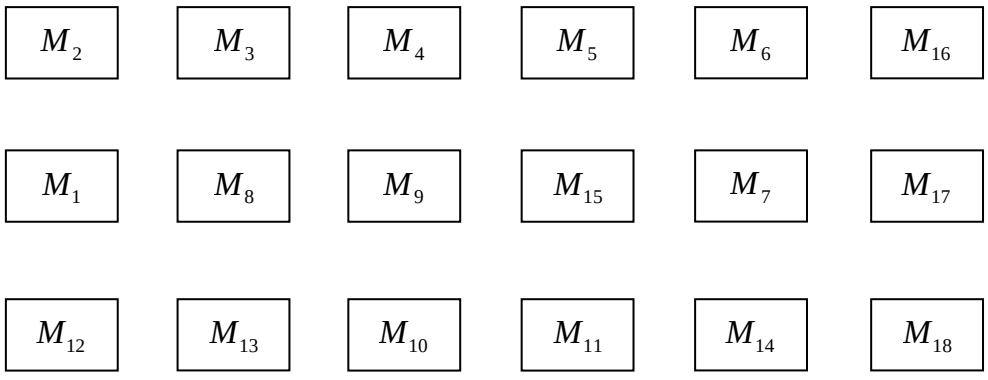


Figura 4.22 – Disposição real das máquinas em uma empresa de Nova Friburgo para o problema com 7 tarefas, 20 máquinas e 72 operações

Tabela 4.6 – Tempo de transporte entre máquinas para problema com 7 tarefas, 20 máquinas e 72 operações (expresso em unidades de tempo UT)

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}	M_{17}	M_{18}
M_1	---	25	25	30	35	40	40	25	30	30	35	25	25	40	35	45	45	45
M_2	25	---	25	30	35	40	40	25	30	35	40	30	30	45	35	45	45	50
M_3	25	25	---	25	30	35	35	25	25	30	35	30	30	40	30	40	40	45
M_4	30	30	25	---	25	30	30	25	25	30	30	35	30	35	25	35	35	40
M_5	35	35	30	25	---	25	25	30	25	30	30	40	35	30	25	30	30	35
M_6	40	40	35	30	25	---	25	35	30	35	30	45	40	30	25	25	25	30
M_7	40	40	35	30	25	25	---	35	30	30	25	40	35	25	25	25	25	25
M_8	25	25	25	25	30	35	35	---	25	25	30	25	25	35	30	40	40	40
M_9	30	30	25	25	25	30	30	25	---	25	25	30	25	30	25	35	35	35
M_{10}	30	35	30	30	30	35	30	25	25	---	25	30	25	30	25	40	35	35
M_{11}	35	40	35	30	30	30	25	30	25	25	---	35	30	25	25	35	30	30
M_{12}	25	30	30	35	40	45	40	25	30	30	35	---	25	40	35	50	45	45
M_{13}	25	30	30	30	35	40	35	25	25	25	30	25	---	35	30	45	40	40
M_{14}	40	45	40	35	30	30	25	35	30	30	25	40	35	---	25	30	25	25
M_{15}	35	35	30	25	25	25	25	30	25	25	25	35	30	25	---	30	30	30
M_{16}	45	45	40	35	30	25	25	40	35	40	35	50	45	30	30	---	25	30
M_{17}	45	45	40	35	30	25	25	40	35	35	30	45	40	25	30	25	---	25
M_{18}	45	50	45	40	35	30	25	40	35	35	30	45	40	25	30	30	25	---

As tabelas com os conjuntos de operações pertencentes a cada tarefa encontram-se no Anexo 3 desta dissertação, sendo que para todas estas tarefas a relação de dependência entre suas respectivas operações representa uma relação de dependência simples, onde O_{2j} será sempre dependente de $O_{2(j-1)}$, exceto no caso em que $j=1$, conforme apresentado anteriormente na seção 2.1. Já a relação de máquinas que devem executar cada operação e o tempo de execução destas operações nas respectivas máquinas são relacionados conforme apresentado no Anexo 4.

4.2.3 Análise de sensibilidade para o problema com 7 tarefas, 20 máquinas e 72 operações

Os gráficos apresentados nas Figs. 4.23 e 4.24 foram gerados seguindo os mesmos padrões dos gráficos referentes às Figs. 4.18 e 4.19, sendo que foram considerados os seguintes valores para as variáveis de projeto:

- Número de vizinhos (Nngh): variando entre 5 e 500 (vide definição na seção 3.1.2)
- Possibilidade de troca entre máquinas (MSP): 90 % (vide definição na seção 3.1.2)
- Tenure: 3 (vide definição na seção 3.1)
- Ibestmcn: 0 (vide definição na seção 3.1.2)

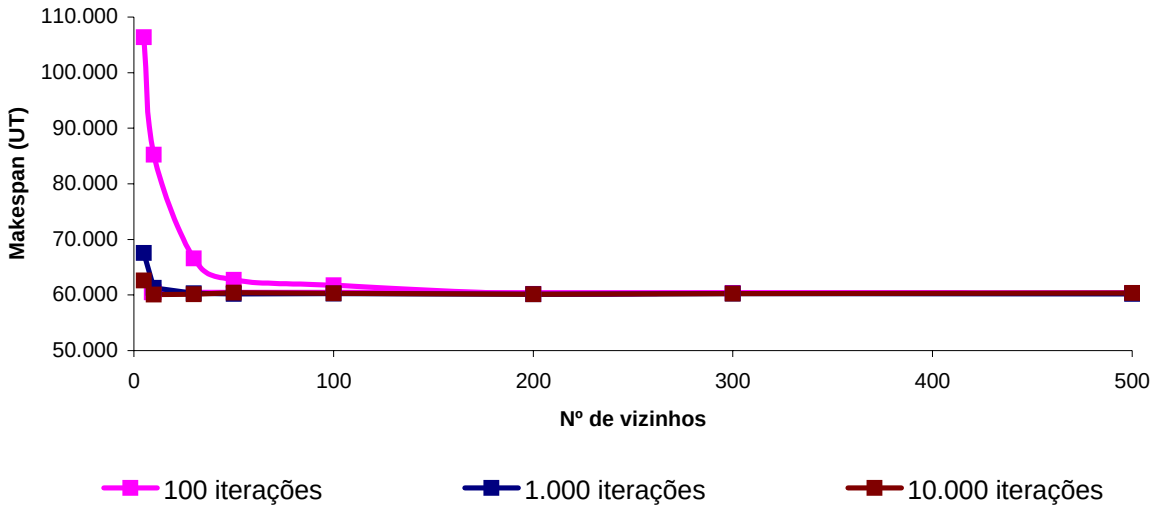


Figura 4.23 – Análise de sensibilidade da variação do *makespan* em função do número de vizinhos para o problema com 7 tarefas, 20 máquinas e 72 operações, considerando o valor de MSP igual a 90%.

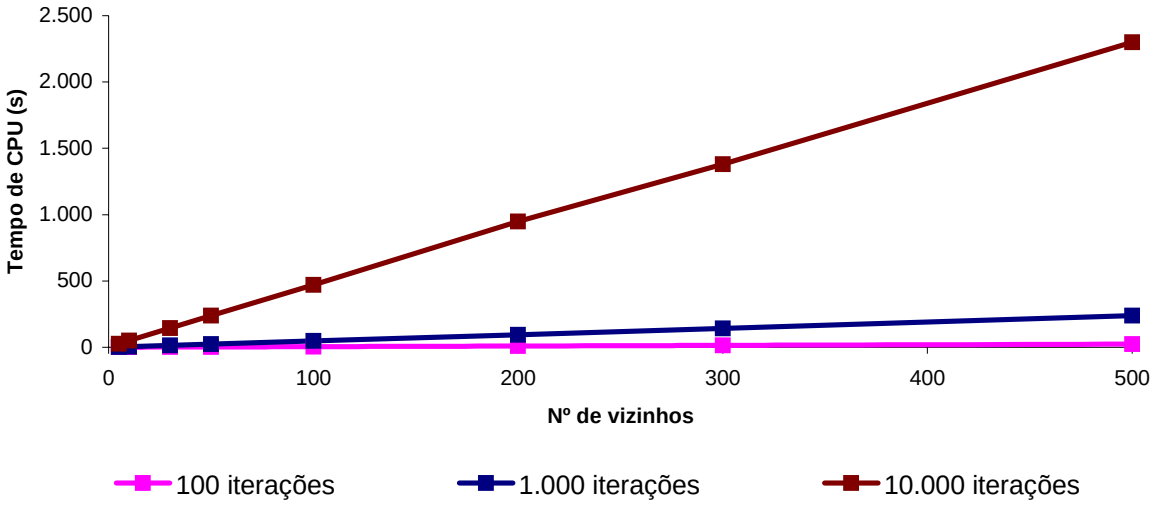


Figura 4.24 – Análise de sensibilidade da variação do tempo de CPU em função do número de vizinhos para o problema com 7 tarefas, 20 máquinas e 72 operações, considerando o valor de MSP igual a 90%.

No caso dos gráficos relativos ao problema aqui apresentado, é possível notar que a variação do *makespan* em função do aumento do número de vizinhos assume um comportamento similar àquele do problema Π_j, sendo que no caso do problema apresentado nesta seção para um número baixo de vizinhos (entre 5 e 30) a redução do *makespan* é mais sensível ao aumento do número de iterações, o que implica em uma redução no valor limite de vizinhos (para aproximadamente 10 vizinhos). Assim sendo observa-se que o valor ótimo para o número de vizinhos será diferente de acordo com cada problema avaliado.

4.2.4 Resultados para o problema com 7 tarefas, 20 máquinas e 72 operações

Tomando-se como base as mesmas variáveis de projeto apresentadas na seção anterior e fixando o valor do número de vizinhos em 10 o programa foi rodado três vezes para um número máximo de iterações igual a 1.000.000. Desta forma foram encontrados os resultados apresentados na Tabela 4.7.

Tabela 4.7 – Resultados encontrado para o problema com 7 tarefas, 20 máquinas e 72 operações

	Nº de Iterações	***Makespan* (UT)**	**Tempo de CPU**
1ª rodada	1.000.000	60.070	4.868 s
2ª rodada	1.000.000	60.070	4.874 s
3ª rodada	1.000.000	60.070	4.851 s

Conforme apresentado na Tabela 4.7, a ferramenta computacional apresentada nesta dissertação tende a convergir para soluções onde $C_{\max}\left(\textit{solução encontrada}\right) = 60.070$ UT . Na Fig. 4.25 é apresentado um gráfico de Gantt referente à solução encontrada na primeira rodada. Neste gráfico é possível verificar que a solução encontrada é uma solução ótima, já que o valor do *makespan* relacionado ao caminho crítico (representado pelas operações da tarefa J_7) não pode ser reduzido.

Partindo-se destes resultados é observa-se que a ferramenta computacional apresentada nesta dissertação é capaz de convergir para resultados ótimos (ou pelo menos sub-ótimos, mas com pequena dispersão) mesmo quando um número maior de operações é considerado.

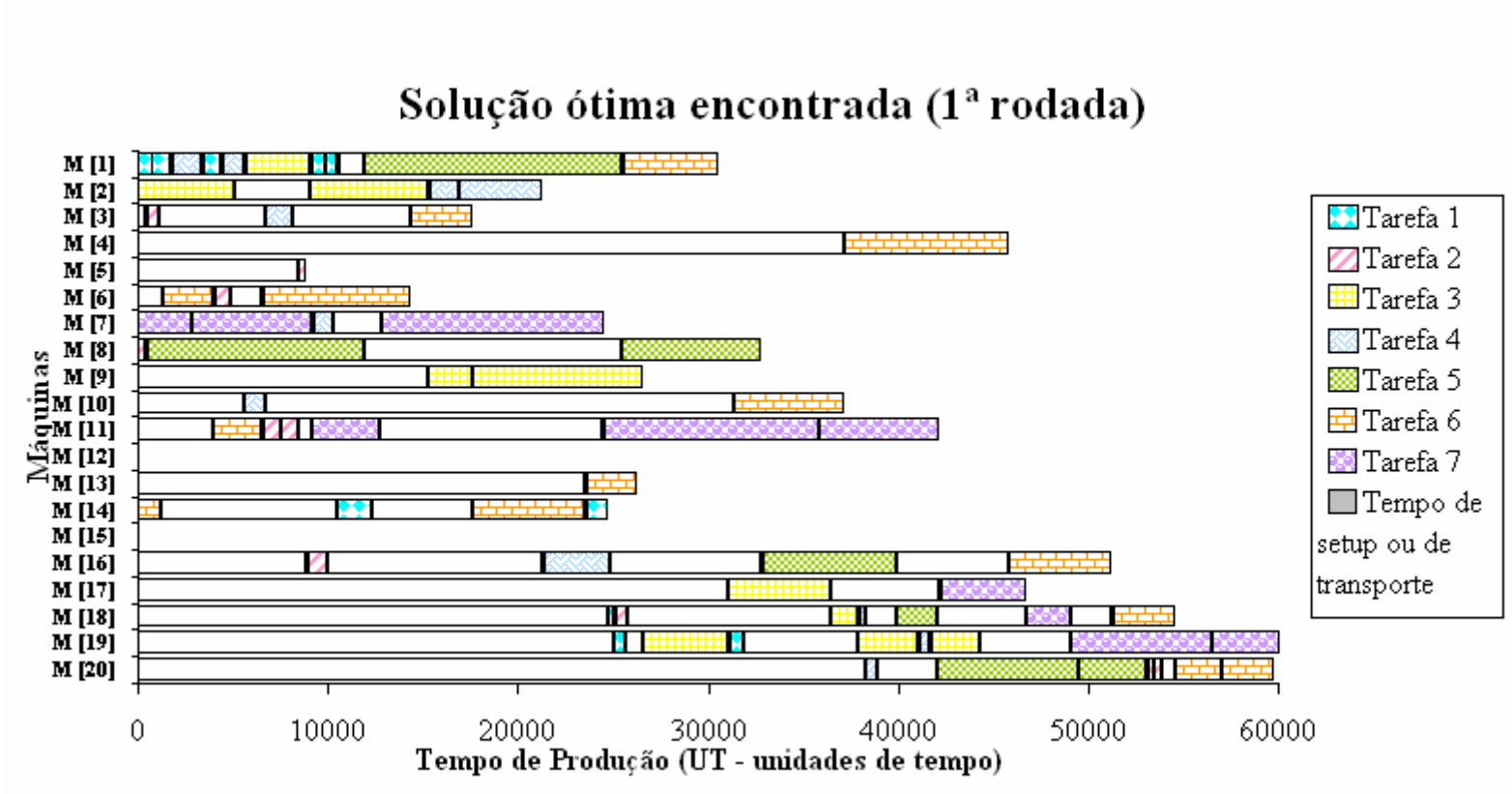


Figura 4.25 – Solução encontrada para problema com 7 tarefas, 20 máquinas e 72 operações

CAPÍTULO 5

5 CONCLUSÕES E TRABALHOS FUTUROS

5.1. CONCLUSÕES

Nesta dissertação foi desenvolvida uma ferramenta computacional para a programação da produção de empresas do setor de confecções situadas no município de Nova Friburgo baseada no método *Tabu Search*. Com base nos testes realizados foi possível chegar a algumas conclusões conforme descrito a seguir.

(1) Nas seções 4.1 e 4.2 desta dissertação são apresentados quatro problemas sendo dois utilizados para validação da ferramenta computacional desenvolvida e dois exemplos práticos, representando o que ocorre em ambientes reais de produção das empresas do setor de confecções avaliado. Nos quatro casos esta mesma ferramenta mostrou-se capaz de chegar aos resultados esperados, ótimos ou sub-ótimos, tanto para os problemas testes apresentados quanto para os problemas de distribuição e sequenciamento de tarefas baseados em casos reais ocorrentes nas empresas do setor de confecções. Nos casos testes apresentados na seção 4.1 foram consideradas duas situações limites, ambas com ordem de sequenciamento de operações por tarefa simples. Na primeira situação foi analisado um caso onde todas as operações podiam ser processadas em qualquer máquina, ou seja, nenhuma restrição de execução foi considerada, já a segunda situação apresentou um caso oposto onde cada operação só poderia ser processada em apenas uma máquina. Tais situações demonstraram a capacidade do programa para considerar tanto as restrições de ordem de processamento

quanto as restrições de execução das operações nas máquinas. Para os casos elaborados com base no problema real ocorrente nas empresas do setor analisado, apresentados na seção 4.2, foram considerados um problema com poucas tarefas, máquinas e operações, onde a ordem de sequenciamento das operações de uma das tarefas permitia que duas ou mais operações de uma mesma tarefa fossem processadas em um mesmo momento, e um outro problema com um número maior de tarefas, máquinas e operações considerando apenas ordens de sequenciamento de operações por tarefa simples. Em ambos os casos o programa conseguiu obter resultados ótimos em baixos tempos de CPU, sendo que o tamanho do problema influiu fortemente no tempo necessário para que tais respostas fossem encontradas.

(2) Com base nos gráficos de análise de sensibilidade apresentados nas secções 4.1.5 e 4.2.3 foi possível verificar que para os dois problemas apresentados, *i.e* problema Π_j e problema com 7 tarefas, 20 máquinas e 72 operações, o tempo de CPU cresce linearmente com o aumento do número de vizinhos, sendo que para um maior número de iterações a variação do tempo de CPU torna-se mais sensível ao aumento deste mesmo número. Já o *makespan* decresce de acordo com o aumento do número de vizinhos até um determinado valor limite, sendo que depois que este valor é ultrapassado o *makespan* apresenta um pequeno aumento em função do aumento do número de vizinhos, sendo que este primeiro aumento apresenta-se mascarado pelo comportamento randômico dos resultados obtidos. Outro fator importante a ser considerado é que o aumento do número de iterações gera uma redução no *makespan* encontrado sendo que esta redução também é limitada de acordo com o valor do número de vizinhos. Tais constatações levam a concluir que para diferentes problemas haverá diferentes valores para o número ótimo de vizinhos, através do qual a ferramenta

computacional desenvolvida será capaz de encontrar resultados ótimos ou sub-ótimos sendo necessário um menor tempo de CPU.

(3) Com base nos resultados apresentados nas seções 4.1 e 4.2 nota-se que a ferramenta computacional desenvolvida é funcional e apresenta um bom desempenho para a solução dos problemas de programação da produção ocorrentes nas empresas do setor de confecções sendo que a mesma poderá ser utilizada como ferramenta auxiliar no processo de planejamento.

(4) Apesar da complexidade do problema tratado e da ferramenta computacional desenvolvida esta é amigável para o usuário, sendo necessário apenas preencher as tabelas com as relações de dependência entre as operações de cada tarefa, e com os tempos de transporte e de *setup*, que são atividades de rotina para o proficional que realiza a programação da produção nas empresas, bem como alguns parâmetros aqui denominados variáveis de projeto. Estas ultimas devem ser escolhidas com base na experiência e no uso do software desenvolvido.

5.2. RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Como forma de complementação e prosseguimento do estudo proposto nesta dissertação algumas sugestões para trabalhos futuros são apresentadas:

(1) Utilização de outros métodos de otimização e aproximação, como algoritmos genéticos e recozimento simulado, para solução do problema apresentado visando reduzir o tempo de CPU.

(2) Utilização de métodos híbridos, buscando obter vantagens das características específicas de cada método também no intuito de reduzir o tempo de CPU.

(3) Consideração da possibilidade de um mesmo funcionário ser capaz de trabalhar em diferentes máquinas, sendo que o número de funcionários poderá ser menor do que o de máquinas, conforme ocorre em algumas situações reais.

(4) Consideração de quebras de máquinas, de antecipação de tarefas, e de outras variações das considerações apresentadas na Tabela 1.4, o que tornariam o problema tratado nesta dissertação mais próximo do problema real ocorrente nas empresas analisadas.

(5) Consideração da alocação e limitação de insumos, como matéria prima e aviamentos.

(6) Aplicação da ferramenta desenvolvida em outros tipos de problemas de programação.

REFERÊNCIAS

BUYURGAN, N.; SAYGIN, C.; KILIC, S. E. Tool allocation in flexible manufacturing systems with tool alternatives. Robotics and Computer–Integrated Manufacturing, v. 20, n. 4, p. 341-349, 2004.

CHUNG, C-S.; FLYNN, J.; KIRCA, O. A branch and bound algorithm to minimize total tardiness for m-machine permutation flowshop problems. European Journal of Operation Research, v. 174, n. 1, p. 1-10, oct. 2006.

GONÇALVES, J. F.; MENDES, J. J.; M., RESENDE, M. G. C. A hybrid genetic algorithm for the job-shop scheduling problem. European Journal of Operation Research, v. 167, n. 1, p. 77-95, 2005.

GUPTA, J. N. D.; STAFFORD JR., E. F. Flowshop scheduling research after five decades. European Journal of Operation Research, v. 169, n. 3, p. 699-711, 2006.

JAIN, A. S.; MEERAN, S. Deterministic job-shop scheduling: past, present and future. European Journal of Operation Research, v. 113, n. 2, p. 390-434, 1999.

JANSEN, K.; MASTROLILLI, M.; SOLIS-OBA, R. Approximation schemes for job shop scheduling with controllable processing times. European Journal of Operation Research, v. 167, n. 2, p. 297-319, 2005.

KOST, G. G.; ZDANOWICZ, R. Modeling of manufacturing systems and robot motions. Journal of Materials Processing Technology, v. 164-165, p. 1369-1378, may 2005.

LAGUNA, M. A guide to implement Tabu Search. Investigación Operativa, v.4, n.1, 1994.

MOSHEIOV, G.; ORON, D. A note on flow-shop and job-shop batch scheduling with identical processing-time jobs. European Journal of Operation Research, v. 161, n. 1, p. 285-291, 2005.

NOWICKI, E.; SMUTNICKI, C. A fast tabu search algorithm for the job shop problem. Management Science, v. 42, n. 6, p. 797-813, 1996.

NOWICKI, E.; SMUTNICKI, C. An advanced tabu search algorithm for the job shop problem. Journal of Scheduling, v. 8, n. 2, p. 145-159, 2005.

ONWUBOLU, G.; DAVENDRA, D., Scheduling flow shops using differential evolution algorithm. European Journal of Operational Research,, v. 171, n. 2, p. 674-692, 2006.

SHANKER, K.; MODI, B. K. A branch and bound based heuristic for multi-product resource constrained scheduling problem in FMS environment. European Journal of Operation Research, v. 113, n. 1, p. 80-90, 1999.

SOLIMANPUR, M..; VRAT, P.; SHANKAR, R. An ant algorithm for the single row layout problem in flexible manufacturing systems. Computers & Operations Research, v. 32, n. 3, p. 583-598, 2005.

SOMLÓ, J. Suitable switching policies for FMS scheduling. Mechatronics, v. 14, n. 2, p. 199-225, 2004.

SWARNKAR, R.; TIWARI, M. K. Modeling machine loading problem of FMSs and its solution methodology using a hybrid tabu search and simulated annealing-based heuristic approach. Robotics and Computer–Integrated Manufacturing, v. 20, n. 3, p. 199-209, 2004.

TAVAKKOLI-MOGHADDAM, R.; DANESHMAND-MEHR, M. A computer simulation model for job shop scheduling problems minimizing makespan. Computers & Industrial Engineering, v. 48, n. 4, p. 811-823, 2005.

TEIXEIRA, E. M. A.; E MENDES, A. S. Modelo de programação horária com uma célula flexível de manufatura com uso de múltiplas ferramentas. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, 19., 1996, Goiânia. Anais... Goiânia: [SBMAC], 1996. p. 230-231.

YAN, H-S.; WANG, N-S.; ZHANG, J-G.; CUI, X-Y. Modelling, scheduling and simulation of flexible manufacturing systems using extended stochastic high-level evaluation Petri nets. Robotics and Computer–Integrated Manufacturing, v. 14, n. 2, p. 121-140, 1998.

YU, H.; REYES, A.; CANG, S.; LLOYD, S. Combined Petri net and AI-based heuristic hybrid search for flexible manufacturing systems – part I. Petri net modeling and heuristic search. Computers & Industrial Engineering, v. 44, n. 4, p. 527-543, 2003.

YU, H.; REYES, A.; CANG, S.; LLOYD, S. Combined Petri net and AI-based heuristic hybrid search for flexible manufacturing systems – part II. Heuristic hybrid search. Computers & Industrial Engineering, v. 44, n. 4, p. 545-566, 2003.

REFERÊNCIAS NA INTERNET

SEBRAE/RJ. Arranjos produtivos locais: perfil das concentrações de atividades econômicas no estado do Rio de Janeiro. Disponível em: <http://www.sebraerj.com.br>. Acesso em: janeiro de 2006.

SOCIESC. FMS Sistema flexível de manufatura: o que é um FMS?. Disponível em: <http://www.grima.ufsc.br/sociesc/paginas/oque.html>. Acesso em: janeiro de 2006.

ANEXO 1

SOLUÇÕES ÓTIMAS PARA O PROBLEMA *N* X *N* COM SEQÜÊNCIA SIMPLES

PARA $N = 5$

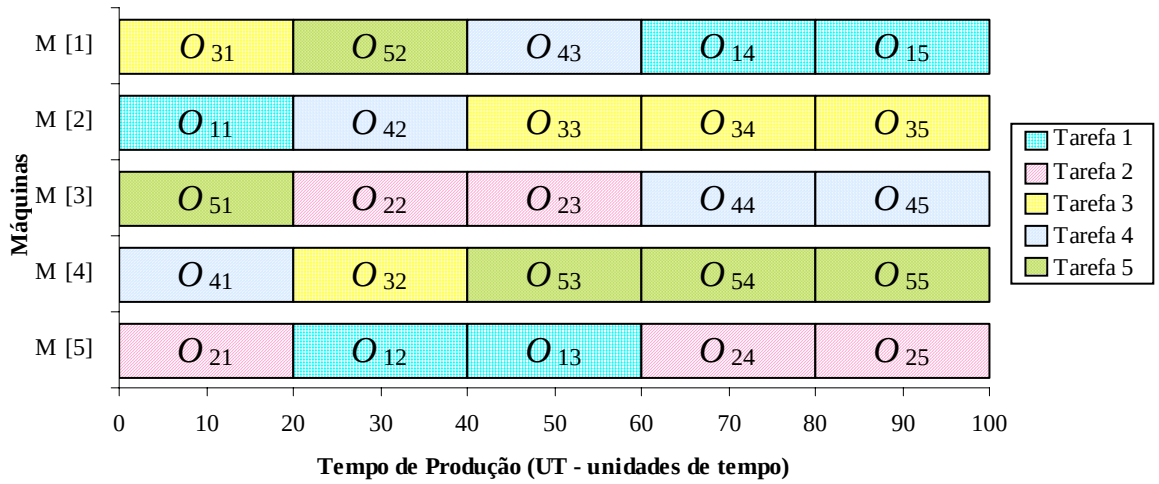


Figura A.1 – Solução encontrada para o problema $N \times N\ (N = 5)$ com seqüência simples (1ª rodada). Esta solução foi encontrada após 94.116 iterações, em um tempo de CPU menor que 40 segundos.
$TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$

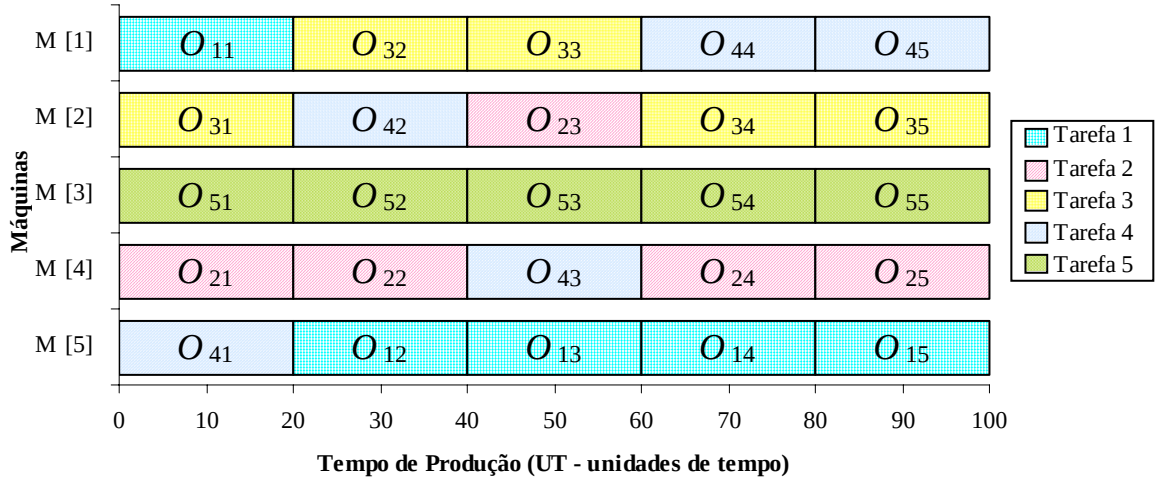


Figura A.2 – Solução encontrada para o problema $N \times N\ (N = 5)$ com seqüência simples (2ª rodada). Esta solução foi encontrada após 147.880 iterações, em um tempo de CPU menor que 63 segundos.
$TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

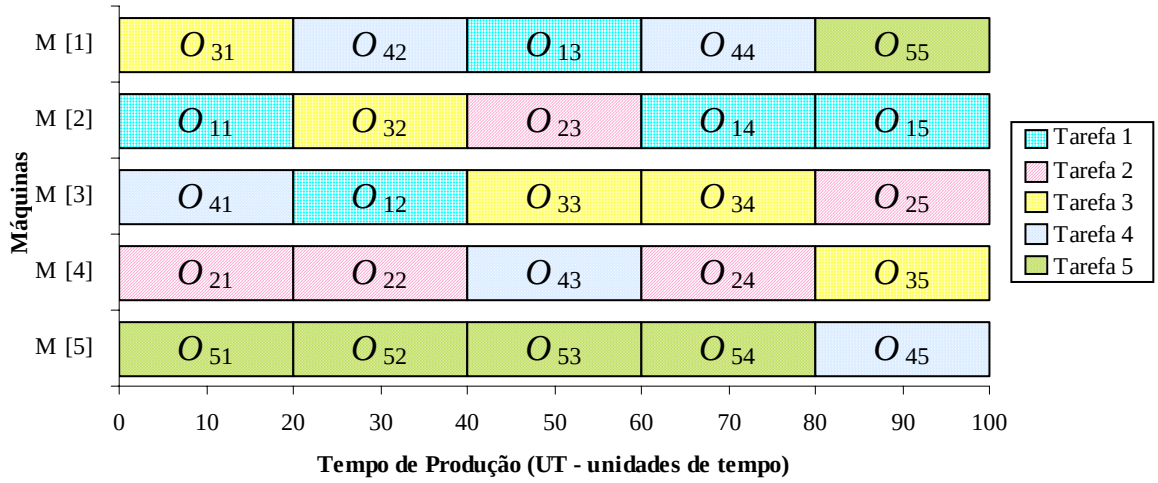


Figura A.3 – Solução encontrada para o problema $N \times N\,(N = 5)$ com seqüência simples (3ª rodada). Esta solução foi encontrada após 546.540 iterações, em um tempo de CPU menor que 229 segundos. $TS_{i_1 j_1 i_2 j_2 k} = TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$

Os gráficos de Gantt acima (Figs. A.1 a A.3) foram gerados tomando-se como base os seguintes valores para as variáveis de projeto:

- Número de vizinhos (Nngh): 10 (vide definição na seção 3.1.2)
- Possibilidade de troca entre máquinas (MSP): 90 % (vide definição na seção 3.1.2)
- Tenure: 3 (vide definição na seção 3.1)
- Ibestmcn: 0 (vide definição na seção 3.1.2)

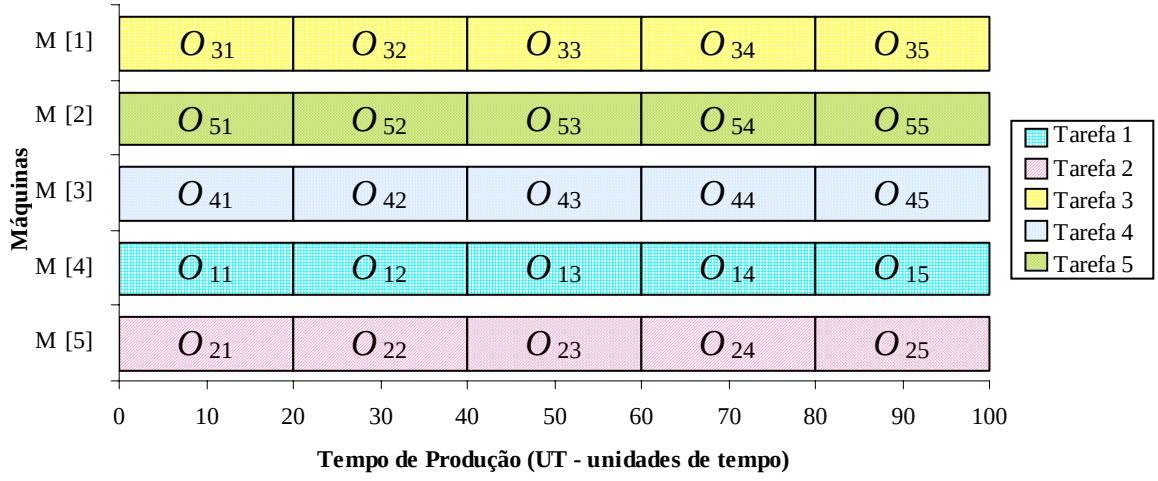


Figura A.4 – Solução encontrada para o problema $N \times N\,(N = 5)$ com seqüência simples considerando tempo de *setup* = 10 UT (1ª rodada). Esta solução foi encontrada após 5.775 iterações, em um tempo de CPU menor que 5 segundos. $TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$

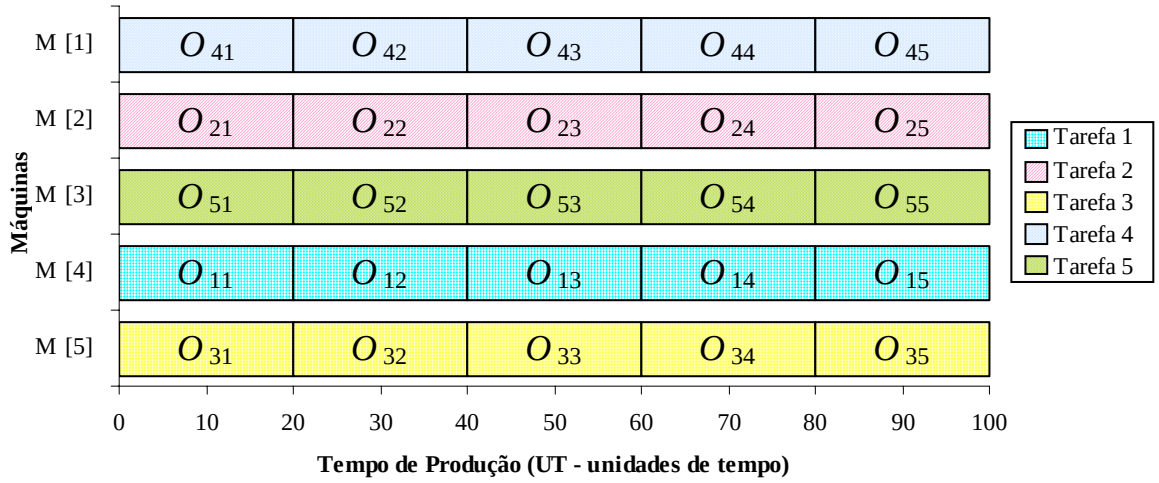


Figura A.5 – Solução encontrada para o problema $N \times N\,(N = 5)$ com seqüência simples considerando tempo de *setup* = 10 UT (2ª rodada). Esta solução foi encontrada após 1.838 iterações, em um tempo de CPU menor que 3 segundos. $TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

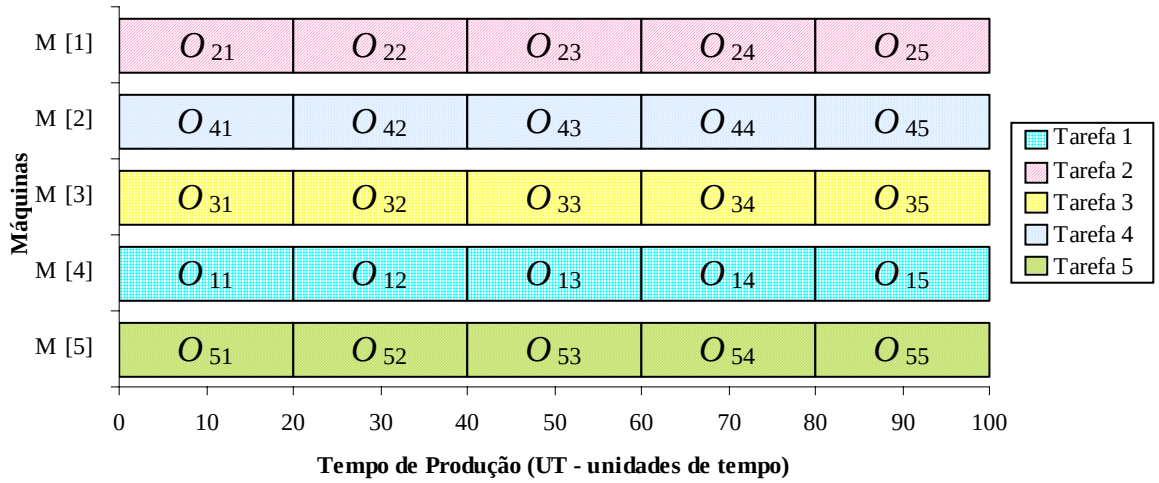


Figura A.6 – Solução encontrada para o problema $N \times N\,(N = 5)$ com seqüência simples considerando tempo de *setup* = 10 UT (3ª rodada). Esta solução foi encontrada após 455 iterações, em um tempo de CPU menor que 2 segundos. $TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 0, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

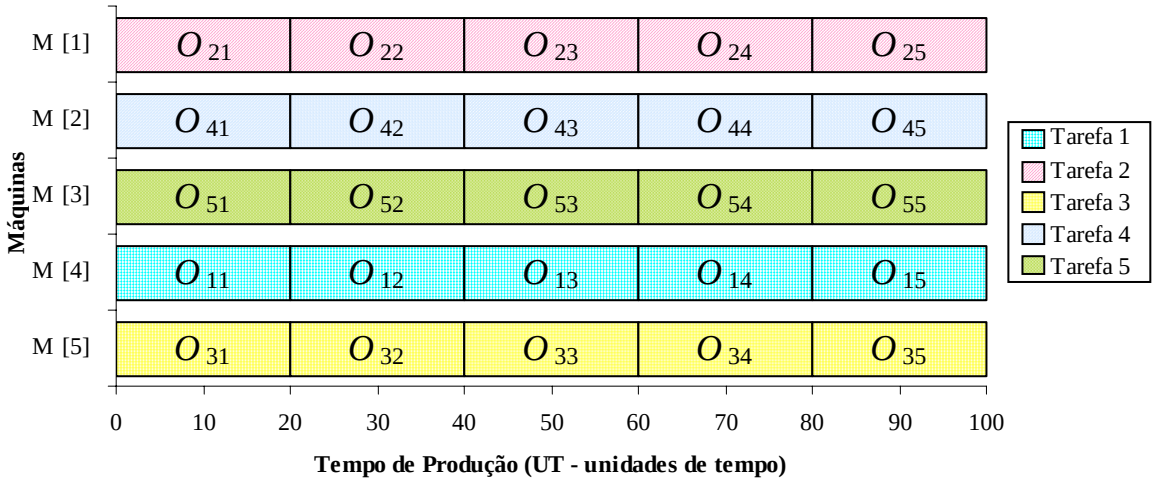


Figura A.7 – Solução encontrada para o problema $N \times N\left(N = 5\right)$ com seqüência simples considerando tempo de *setup* = 10 UT e tempo de transporte = 15 UT (1ª rodada). Esta solução foi encontrada após 25.723 iterações, em um tempo de CPU menor que 21 segundos.
$TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 15, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

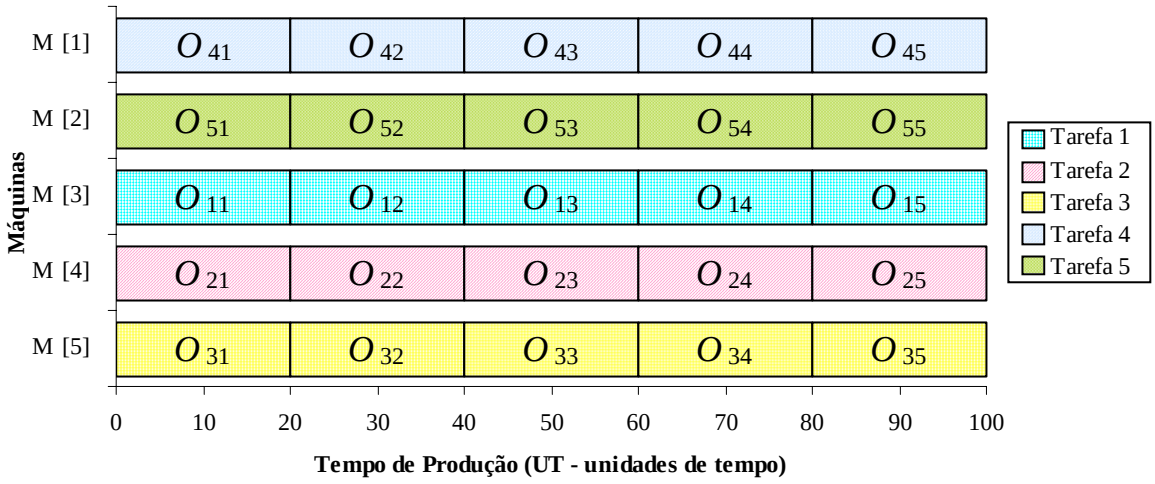


Figura A.8 – Solução encontrada para o problema $N \times N\left(N = 5\right)$ com seqüência simples considerando tempo de *setup* = 10 UT e tempo de transporte = 15 UT (2ª rodada). Esta solução foi encontrada após 15.009 iterações, em um tempo de CPU menor que 12 segundos.
$TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 15, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

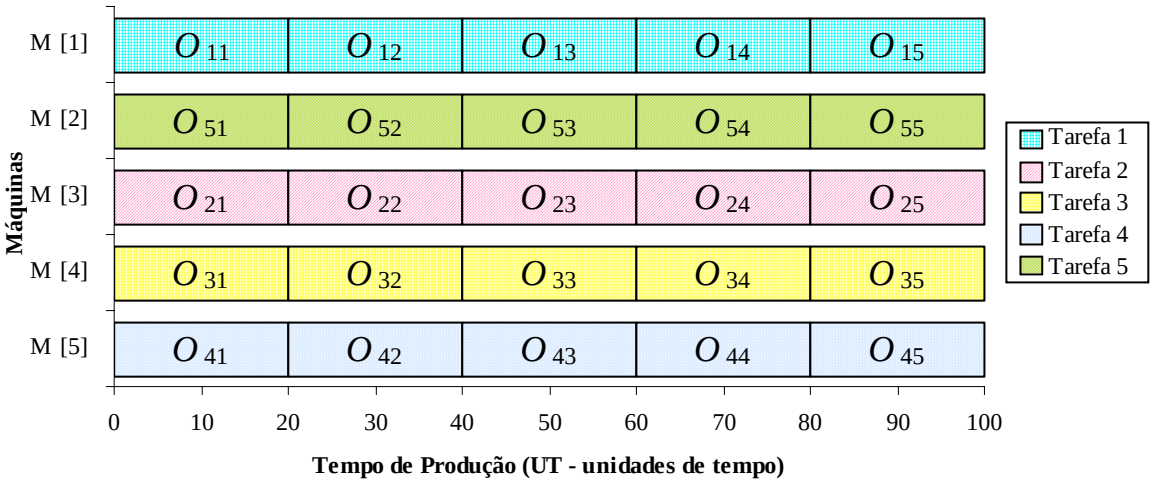


Figura A.9 – Solução encontrada para o problema $N \times N\,(N = 5)$ com seqüência simples considerando tempo de *setup* = 10 UT e tempo de transporte = 15 UT (1ª rodada). Esta solução foi encontrada após 32.950 iterações, em um tempo de CPU menor que 26 segundos.
$TS_{i_1 j_1 i_2 j_2 k} = 10$ e $TT_{k_1 k_2} = 15, \forall\, i_1, i_2, j_1, j_2, k, k_1, k_2$.

Os gráficos de Gantt acima (Figs. A.4 a A.9) foram gerados tomando-se como base os seguintes valores para as variáveis de projeto:

- Número de vizinhos (Nngh): 20 (vide definição na seção 3.1.2)
- Possibilidade de troca entre máquinas (MSP): 90 % (vide definição na seção 3.1.2)
- Tenure: 3 (vide definição na seção 3.1)
- Ibestmcn: 1 (vide definição na seção 3.1.2)

ANEXO 2

SOLUÇÕES ÓTIMAS PARA O PROBLEMA COM 2 TAREFAS, 7 MÁQUINAS E 14 OPERAÇÕES – 1ª E 2ª RODADA (VIDE TABELA 4.3)

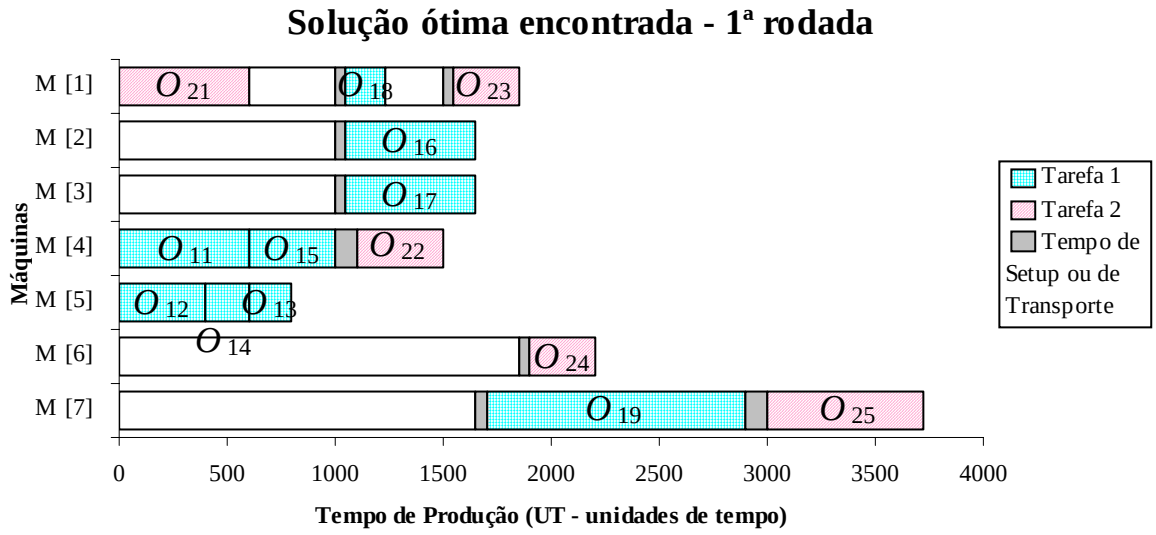


Figura A.1 – Solução encontrada para o problema com 2 tarefas, 7 máquinas e 14 operações (1ª rodada).

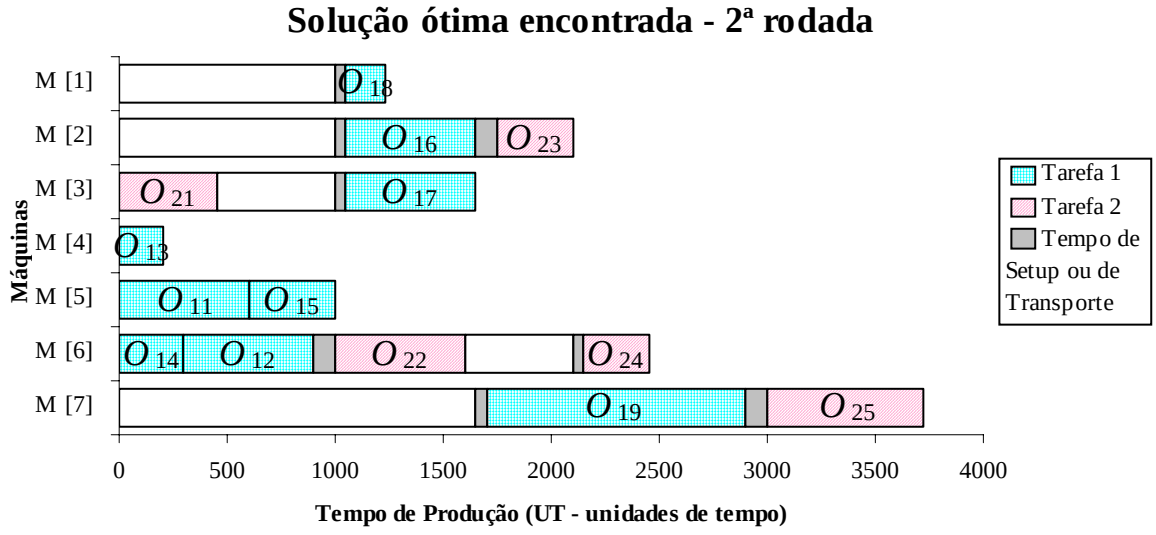


Figura A.2 – Solução encontrada para o problema com 2 tarefas, 7 máquinas e 14 operações (2ª rodada).

ANEXO 3

CONJUNTO DE OPERAÇÕES POR TAREFA PARA O PROBLEMA COM 7 TAREFAS, 20 MÁQUINAS E 72 OPERAÇÕES

Tabela A.1 – Operações da tarefa J_1 para o problema com 7 tarefas, 20 máquinas e 72 operações

Tarefa J_1	
Operações	**Descrição**
O_{11}	Fechar frente
O_{12}	Fechar frente com fundo e costas
O_{13}	Pregar palas nas costas
O_{14}	Fechar lados
O_{15}	Chulear fundo
O_{16}	Pregar elástico nas pernas
O_{17}	Pregar elástico na cintura com etiquetas
O_{18}	Pregar laço e tag
O_{19}	Limpar e revisar
O_{110}	Embalar

Tabela A.2 – Operações da tarefa J_2 para o problema com 7 tarefas, 20 máquinas e 72 operações

Tarefa J_2	
Operações	**Descrição**
O_{21}	Pregar barrinha na frente
O_{22}	Fechar frente
O_{23}	Fechar fundo com costas e chulear
O_{24}	Pregar sanduíche nas pernas com sobras
O_{25}	Pregar sanduíche nas costas c/ etiquetas e sobras
O_{26}	Fechar frente com costas
O_{27}	Arrematar cintura e fundo
O_{28}	Pregar laço e tag
O_{29}	Limpar e revisar
O_{210}	Embalar

Tabela A.3 – Operações da tarefa J_3 para o problema com 7 tarefas, 20 máquinas e 72 operações

Tarefa J_3	
Operações	**Descrição**
O_{31}	Fechar fundo com frente e costas
O_{32}	Chulear fundo
O_{33}	Pregar palas com etiquetas
O_{24}	Pregar sanduíche na frente e cintura
O_{35}	Pregar sanduíche nas pernas, costas e cintura
O_{36}	Enfiar fivela
O_{37}	Arrematar laterais
O_{38}	Pregar tag
O_{39}	Limpar e revisar
O_{310}	Embalar

Tabela A.4 – Operações da tarefa J_4 para o problema com 7 tarefas, 20 máquinas e 72 operações

Tarefa J_4	
Operações	**Descrição**
O_{41}	Fechar fundo com frente e costas
O_{42}	Chulear fundo
O_{43}	Pregar sanduíche nas pernas
O_{44}	Fechar lados com etiquetas
O_{45}	Chulear passador
O_{46}	Fechar cós com cinto
O_{47}	Fechar cós na cintura
O_{48}	Arrematar lados
O_{49}	Pregar tag
O_{410}	Limpar e revisar
O_{411}	Embalar

Tabela A.5 – Operações da tarefa J_5 para o problema com 7 tarefas, 20 máquinas e 72 operações

Tarefa J_5	
Operações	**Descrição**
O_{51}	Transpassar renda na frente e costas
O_{52}	Fechar lados com etiquetas
O_{53}	Pregar sanduíche no decote e cavas
O_{54}	Arrematar alças e lados
O_{55}	Pregar tag
O_{56}	Limpar e revisar
O_{57}	Embalar

Tabela A.6 – Operações da tarefa J_6 para o problema com 7 tarefas, 20 máquinas e 72 operações

Tarefa J_6	
Operações	**Descrição**
O_{61}	Pregar elástico na renda
O_{62}	Fechar renda na frente
O_{63}	Pespontar renda
O_{64}	Fechar fundos com costas e frente
O_{65}	Chulear fundo
O_{66}	Pregar elástico nas pernas
O_{67}	Pregar elástico na cintura
O_{68}	Encapar laterais
O_{69}	Pespontar laterais
O_{610}	Fechar palas laterais
O_{611}	Arrematar lados
O_{612}	Pregar tag
O_{613}	Limpar e revisar
O_{614}	Embalar

Tabela A.7 – Operações da tarefa J_7 para o problema com 7 tarefas, 20 máquinas e 72 operações

Tarefa J_7	
Operações	**Descrição**
O_{71}	Fechar renda
O_{72}	Pregar renda na frente
O_{73}	Pespontar renda
O_{74}	Fechar lados com etiqueta
O_{75}	Fazer bainha
O_{76}	Pregar sanduíche nas costas e cavas
O_{77}	Arrematar alças
O_{78}	Pregar tag
O_{79}	Limpar e revisar
O_{710}	Embalar

ANEXO 4

RELAÇÃO DE TEMPO DE PROCESSAMENTO DE OPERAÇÃO POR MÁQUINA PARA PROBLEMA COM 7 TAREFAS, 20 MÁQUINAS E 72 OPERAÇÕES

Nas tabelas A.8 a A.14 apresentadas a seguir o símbolo x representa que a operação não pode ser realizada na máquina indicada.

Tabela A.8 – Relação de tempo de operação por máquina para o problema com 7 tarefas, 20 máquinas e 72 operações para a tarefa J_1

	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}	O_{16}	O_{17}	O_{18}	O_{19}	O_{110}
M_1	672 UT	1032 UT	984 UT	768 UT	576 UT	X	X	X	X	X
M_2	672 UT	1032 UT	984 UT	768 UT	576 UT	X	X	X	X	X
M_3	672 UT	1032 UT	984 UT	768 UT	576 UT	X	X	X	X	X
M_4	672 UT	1032 UT	984 UT	768 UT	576 UT	X	X	X	X	X
M_5	672 UT	1032 UT	984 UT	768 UT	576 UT	X	X	X	X	X
M_6	672 UT	1032 UT	984 UT	768 UT	576 UT	X	X	X	X	X
M_7	672 UT	1032 UT	984 UT	768 UT	576 UT	X	X	X	X	X
M_8	X	X	X	X	X	X	X	X	X	X
M_9	X	X	X	X	X	X	X	X	X	X
M_{10}	X	X	X	X	X	X	X	X	X	X
M_{11}	X	X	X	X	X	X	X	X	X	X
M_{12}	X	X	X	X	X	1848 UT	1056 UT	X	X	X
M_{13}	X	X	X	X	X	1848 UT	1056 UT	X	X	X
M_{14}	X	X	X	X	X	1848 UT	1056 UT	X	X	X
M_{15}	X	X	X	X	X	X	X	X	X	X
M_{16}	X	X	X	X	X	X	X	X	X	X
M_{17}	X	X	X	X	X	X	X	X	X	X
M_{18}	X	X	X	X	X	X	X	288 UT	X	X
M_{19}	X	X	X	X	X	X	X	X	600 UT	648 UT
M_{20}	X	X	X	X	X	X	X	X	600 UT	648 UT

Tabela A.9 – Relação de tempo de operação por máquina para o problema com 7 tarefas, 20 máquinas e 72 operações para a tarefa J_2

	O_{21}	O_{22}	O_{23}	O_{24}	O_{25}	O_{26}	O_{27}	O_{28}	O_{29}	O_{210}
M_1	X	605 UT	778 UT	X	X	432 UT	X	X	X	X
M_2	X	605 UT	778 UT	X	X	432 UT	X	X	X	X
M_3	X	605 UT	778 UT	X	X	432 UT	X	X	X	X
M_4	X	605 UT	778 UT	X	X	432 UT	X	X	X	X
M_5	X	605 UT	778 UT	X	X	432 UT	X	X	X	X
M_6	X	605 UT	778 UT	X	X	432 UT	X	X	X	X
M_7	X	605 UT	778 UT	X	X	432 UT	X	X	X	X
M_8	454 UT	X	X	929 UT	864 UT	X	X	X	X	X
M_9	454 UT	X	X	929 UT	864 UT	X	X	X	X	X
M_{10}	454 UT	X	X	929 UT	864 UT	X	X	X	X	X
M_{11}	454 UT	X	X	929 UT	864 UT	X	X	X	X	X
M_{12}	X	X	X	X	X	X	X	X	X	X
M_{13}	X	X	X	X	X	X	X	X	X	X
M_{14}	X	X	X	X	X	X	X	X	X	X
M_{15}	X	X	X	X	X	X	X	X	X	X
M_{16}	X	X	X	X	X	X	1080 UT	X	X	X
M_{17}	X	X	X	X	X	X	1080 UT	X	X	X
M_{18}	X	X	X	X	X	X	X	648 UT	X	X
M_{19}	X	X	X	X	X	X	X	X	346 UT	324 UT
M_{20}	X	X	X	X	X	X	X	X	346 UT	324 UT

Tabela A.10 – Relação de tempo de operação por máquina para o problema com 7 tarefas, 20 máquinas e 72 operações para a tarefa J_3

	O_{31}	O_{32}	O_{33}	O_{34}	O_{35}	O_{36}	O_{37}	O_{38}	O_{39}	O_{310}
M_1	5040 UT	3360 UT	6216 UT	X	X	X	X	X	X	X
M_2	5040 UT	3360 UT	6216 UT	X	X	X	X	X	X	X
M_3	5040 UT	3360 UT	6216 UT	X	X	X	X	X	X	X
M_4	5040 UT	3360 UT	6216 UT	X	X	X	X	X	X	X
M_5	5040 UT	3360 UT	6216 UT	X	X	X	X	X	X	X
M_6	5040 UT	3360 UT	6216 UT	X	X	X	X	X	X	X
M_7	5040 UT	3360 UT	6216 UT	X	X	X	X	X	X	X
M_8	X	X	X	2352 UT	8904 UT	X	X	X	X	X
M_9	X	X	X	2352 UT	8904 UT	X	X	X	X	X
M_{10}	X	X	X	2352 UT	8904 UT	X	X	X	X	X
M_{11}	X	X	X	2352 UT	8904 UT	X	X	X	X	X
M_{12}	X	X	X	X	X	X	X	X	X	X
M_{13}	X	X	X	X	X	X	X	X	X	X
M_{14}	X	X	X	X	X	X	X	X	X	X
M_{15}	X	X	X	X	X	X	X	X	X	X
M_{16}	X	X	X	X	X	X	5376 UT	X	X	X
M_{17}	X	X	X	X	X	X	5376 UT	X	X	X
M_{18}	X	X	X	X	X	X	X	1344 UT	X	X
M_{19}	X	X	X	X	X	4536 UT	X	X	3192 UT	2520 UT
M_{20}	X	X	X	X	X	4536 UT	X	X	3192 UT	2520 UT

Tabela A.11 – Relação de tempo de operação por máquina para problema com 7 tarefas, 20 máquinas e 72 operações para tarefa J_4

	O_{41}	O_{42}	O_{43}	O_{44}	O_{45}	O_{46}	O_{47}	O_{48}	O_{49}	O_{410}	O_{411}
M_1	1536 UT	1037 UT	X	1382 UT	1037 UT	1536 UT	4416 UT	X	X	X	X
M_2	1536 UT	1037 UT	X	1382 UT	1037 UT	1536 UT	4416 UT	X	X	X	X
M_3	1536 UT	1037 UT	X	1382 UT	1037 UT	1536 UT	4416 UT	X	X	X	X
M_4	1536 UT	1037 UT	X	1382 UT	1037 UT	1536 UT	4416 UT	X	X	X	X
M_5	1536 UT	1037 UT	X	1382 UT	1037 UT	1536 UT	4416 UT	X	X	X	X
M_6	1536 UT	1037 UT	X	1382 UT	1037 UT	1536 UT	4416 UT	X	X	X	X
M_7	1536 UT	1037 UT	X	1382 UT	1037 UT	1536 UT	4416 UT	X	X	X	X
M_8	X	X	1114 UT	X	X	X	X	X	X	X	X
M_9	X	X	1114 UT	X	X	X	X	X	X	X	X
M_{10}	X	X	1114 UT	X	X	X	X	X	X	X	X
M_{11}	X	X	1114 UT	X	X	X	X	X	X	X	X
M_{12}	X	X	X	X	X	X	X	X	X	X	X
M_{13}	X	X	X	X	X	X	X	X	X	X	X
M_{14}	X	X	X	X	X	X	X	X	X	X	X
M_{15}	X	X	X	X	X	X	X	X	X	X	X
M_{16}	X	X	X	X	X	X	X	3456 UT	X	X	X
M_{17}	X	X	X	X	X	X	X	3456 UT	X	X	X
M_{18}	X	X	X	X	X	X	X	X	346 UT	X	X
M_{19}	X	X	X	X	X	X	X	X	X	576 UT	499 UT
M_{20}	X	X	X	X	X	X	X	X	X	576 UT	499 UT

Tabela A.12 – Relação de tempo de operação por máquina para o problema com 7 tarefas, 20 máquinas e 72 operações para a tarefa J_5

	O_{51}	O_{52}	O_{53}	O_{54}	O_{55}	O_{56}	O_{57}
M_1	X	13500 UT	X	X	X	X	X
M_2	X	13500 UT	X	X	X	X	X
M_3	X	13500 UT	X	X	X	X	X
M_4	X	13500 UT	X	X	X	X	X
M_5	X	13500 UT	X	X	X	X	X
M_6	X	13500 UT	X	X	X	X	X
M_7	X	13500 UT	X	X	X	X	X
M_8	11340 UT	X	7380 UT	X	X	X	X
M_9	11340 UT	X	7380 UT	X	X	X	X
M_{10}	11340 UT	X	7380 UT	X	X	X	X
M_{11}	11340 UT	X	7380 UT	X	X	X	X
M_{12}	X	X	X	X	X	X	X
M_{13}	X	X	X	X	X	X	X
M_{14}	X	X	X	X	X	X	X
M_{15}	X	X	X	X	X	X	X
M_{16}	X	X	X	7020 UT	X	X	X
M_{17}	X	X	X	7020 UT	X	X	X
M_{18}	X	X	X	X	2160 UT	X	X
M_{19}	X	X	X	X	X	7380 UT	3600 UT
M_{20}	X	X	X	X	X	7380 UT	3600 UT

Tabela A.13 – Relação de tempo de operação por máquina para o problema com 7 tarefas, 20 máquinas e 72 operações para a tarefa J_6

	O_{61}	O_{62}	O_{63}	O_{64}	O_{65}	O_{66}	O_{67}	O_{68}	O_{69}	O_{610}	O_{611}	O_{612}	O_{613}	O_{614}
M_1	X	2700 UT	X	7740 UT	3240 UT	X	X	5040 UT	X	8640 UT	X	X	X	X
M_2	X	2700 UT	X	7740 UT	3240 UT	X	X	5040 UT	X	8640 UT	X	X	X	X
M_3	X	2700 UT	X	7740 UT	3240 UT	X	X	5040 UT	X	8640 UT	X	X	X	X
M_4	X	2700 UT	X	7740 UT	3240 UT	X	X	5040 UT	X	8640 UT	X	X	X	X
M_5	X	2700 UT	X	7740 UT	3240 UT	X	X	5040 UT	X	8640 UT	X	X	X	X
M_6	X	2700 UT	X	7740 UT	3240 UT	X	X	5040 UT	X	8640 UT	X	X	X	X
M_7	X	2700 UT	X	7740 UT	3240 UT	X	X	5040 UT	X	8640 UT	X	X	X	X
M_8	X	X	2520 UT	X	X	X	X	X	5760 UT	X	X	X	X	X
M_9	X	X	2520 UT	X	X	X	X	X	5760 UT	X	X	X	X	X
M_{10}	X	X	2520 UT	X	X	X	X	X	5760 UT	X	X	X	X	X
M_{11}	X	X	2520 UT	X	X	X	X	X	5760 UT	X	X	X	X	X
M_{12}	1260 UT	X	X	X	X	5940 UT	2700 UT	X	X	X	X	X	X	X
M_{13}	1260 UT	X	X	X	X	5940 UT	2700 UT	X	X	X	X	X	X	X
M_{14}	1260 UT	X	X	X	X	5940 UT	2700 UT	X	X	X	X	X	X	X
M_{15}	X	X	X	X	X	X	X	X	X	X	X	X	X	X
M_{16}	X	X	X	X	X	X	X	X	X	X	5400 UT	X	X	X
M_{17}	X	X	X	X	X	X	X	X	X	X	5400 UT	X	X	X
M_{18}	X	X	X	X	X	X	X	X	X	X	X	3240 UT	X	X
M_{19}	X	X	X	X	X	X	X	X	X	X	X	X	2520 UT	2700 UT
M_{20}	X	X	X	X	X	X	X	X	X	X	X	X	2520 UT	2700 UT

Tabela A.14 – Relação de tempo de operação por máquina para o problema com 7 tarefas, 20 máquinas e 72 operações para a tarefa J_7

	O_{71}	O_{72}	O_{73}	O_{74}	O_{75}	O_{76}	O_{77}	O_{78}	O_{79}	O_{710}
M_1	2880 UT	6300 UT	X	11700 UT	X	X	X	X	X	X
M_2	2880 UT	6300 UT	X	11700 UT	X	X	X	X	X	X
M_3	2880 UT	6300 UT	X	11700 UT	X	X	X	X	X	X
M_4	2880 UT	6300 UT	X	11700 UT	X	X	X	X	X	X
M_5	2880 UT	6300 UT	X	11700 UT	X	X	X	X	X	X
M_6	2880 UT	6300 UT	X	11700 UT	X	X	X	X	X	X
M_7	2880 UT	6300 UT	X	11700 UT	X	X	X	X	X	X
M_8	X	X	3600 UT	X	11340 UT	6300 UT	X	X	X	X
M_9	X	X	3600 UT	X	11340 UT	6300 UT	X	X	X	X
M_{10}	X	X	3600 UT	X	11340 UT	6300 UT	X	X	X	X
M_{11}	X	X	3600 UT	X	11340 UT	6300 UT	X	X	X	X
M_{12}	X	X	X	X	X	X	X	X	X	X
M_{13}	X	X	X	X	X	X	X	X	X	X
M_{14}	X	X	X	X	X	X	X	X	X	X
M_{15}	X	X	X	X	X	X	X	X	X	X
M_{16}	X	X	X	X	X	X	4500 UT	X	X	X
M_{17}	X	X	X	X	X	X	4500 UT	X	X	X
M_{18}	X	X	X	X	X	X	X	2340 UT	X	X
M_{19}	X	X	X	X	X	X	X	X	7380 UT	3600 UT
M_{20}	X	X	X	X	X	X	X	X	7380 UT	3600 UT