DESENVOLVIMENTO DE APLICAÇÃO COMPUTACIONAL PARA UM PROBLEMA DE SEQÜENCIAMENTO UTILIZANDO ALGORÍTMO BRANCH AND BOUND

Daniel de Oliveira Mota

MONOGRAFIA SUBMETIDA À COORDENAÇÃO DE CURSO DE ENGENHARIA DE PRODUÇÃO DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A GRADUAÇÃO EM ENGENHARIA PRODUÇÃO

Aprovada por:

_____

Prof. Fernando Marques de Almeida Nogueira, M.Sc.

_____

Prof. Eduardo Breviglieri, D. Sc.

_____

Prof. Clovis Neumann, D. Sc.

JUIZ DE FORA, MG - BRASIL

NOVEMBRO 2007

MOTA, DANIEL DE OLIVEIRA
Desenvolvimento de aplicação Computacional para um problema de Seqüenciamento utilizando algoritmo Branch and Bound [Minas Gerais] 2007
VII, 33 p. 29,7 cm (EPD/UFJF, Graduação, Engenharia de Produção, 2007)
Monografia - Universidade Federal de Juiz de Fora, Departamento de Engenharia de Produção

1. Scheduling
2. Job-Shop
3. “Resource Constrained”
4. Matlab
5. Sequenciamento

I. EPD/UFJF II. Título (série)

DEDICATÓRIA

Dedico esta monografia a todos aqueles que contribuíram nesta jornada na Engenharia de Produção, em especial a minha mãe.

AGRADECIMENTO

Agradeço ao professor Fernando Nogueira, pela dedicação na orientação dessa monografia; a universidade norte americana NCA&T, em especial a Dr. Lauren Davis e Dr. Eui Park que me incentivaram a pesquisa nesta área; e a todos aqueles que contribuíram para o desenvolvimento desse trabalho.

Resumo da monografia apresentada à Coordenação de Curso de Engenharia de Produção como parte dos requisitos necessários para a graduação em Engenharia Produção.

DESENVOLVIMENTO DE APLICAÇÃO COMPUTACIONAL PARA UM PROBLEMA DE SEQÜENCIAMENTO UTILIZANDO ALGORÍTMO BRANCH AND BOUND

Daniel de Oliveira Mota

Novembro/2007

Orientador: Fernando Marques de Almeida Nogueira

Curso: Engenharia de Produção

É apresentada neste trabalho uma proposta de solução de problemas de S*cheduling* de projetos sujeito à recursos restritos (RCSP) utilizando algoritmo *Branch and Bound*. Através da utilização de um seqüenciamento planejado sobre uma das diversas funções objetivos, pode-se obter além de economias substanciais no projeto de produção em termos de utilização dos recursos ou no tempo total de término das atividades inerentes ao projeto. Originalmente, a célula de remanufatura utilizada como estudo é uma oficina de aviões, onde muitos processos podem ocorrer concomitantemente. O modelo apresentado, chamado de "Resourced Constrained", expõe a flexibilidade de se poder realizar tarefas simultâneas para um mesmo *Job*. O problema foi modelado utilizando a ferramenta matemática MATLAB. Com testes de complexidade progressiva, chegou-se a um limite operacional de 7 *Jobs* e 31 atividades, com tempo de execução de 9813 segundos (aproximadamente 3 horas). Gráficos de Gantt foram apresentados com os exemplos testados no trabalho, bem como detalhes técnicos do teste, porém, para maior complexidade do problema abordado, vê-se a necessidade de técnicas mais robustas como utilização de meta-heuristicas.

Palavras-chaves: Scheduling, Resource Constrained, Seqüenciamento, Otimização.

Abstract of monography presented to the coordination of Department of Production Engineering course as a partial fulfillment of the requirements for the undergraduate degree

COMPUTER APLICATION DEVELOPMENT FOR A SCHEDULING PROBLEM USING A BRANCH AND BOUND ALGORITHM

Daniel de Oliveira Mota

November/2007

Advisors: Fernando Marques de Almeida Nogueira

Department: Production Engineering

Is shown in this monography a solution propose for a Resource Constrained Scheduling Problem (RCSP) using the Branch and Bound Algorithm by the utilization of a Scheduling plan under one of the objective functions available substantial economy can be reached in the production project in terms of resources utilization or finalizing tasks within the project. Originally, the remanufacturing approached is an airplane plant, where lots of process happens simultaneously. The model presented exposes the flexibility of realizing simultaneous tasks for the same Job. The problem were modeled in a mathematics computer tool called MATLAB. Progressive complexity tests were realized reaching up to 7 Jobs and 31 activities each as the operational limit. The execution time was 9813 seconds (approximately 3 hours). Gantt charts were also presented showing the examples tested in the paper, as well as technical details, although, for further complexity is necessary to take in consideration more robust techniques, such as meta-heuristics.

Key words: Scheduling, Resource Constrained, Optimization

SUMÁRIO

Capítulo I 1
INTRODUÇÃO 1
1. CONSIDERAÇÕES INICIAIS 1
2. OBJETIVO 1
3. ESCOPO DO TRABALHO 1
4. CONDIÇÕES DE CONTORNO 2
5. JUSTIFICATIVAS 2
Capítulo II 3
REVISÃO BIBLIOGRÁFICA 3
1. SCHEDULING 3
2. PROJETO 11
2.1. SIMPLEX 13
3. PROGRAMAÇÃO INTEIRA 14
3.1. BRANCH AND BOUND 14
Capítulo III 16
DESCRIÇÃO DO PROBLEMA 16
1. APRESENTAÇÃO DO PROBLEMA 16
2. APLICAÇÃO 17
Capítulo III 21
FERRAMENTA DE OTIMIZAÇÃO 21
1. INTRODUÇÃO 21
2. O ALGORITMO 21
3. ANÁLISE DE RESULTADOS 22
4. QUADRO COMPARATIVO DE PERFORMANCE 25
5. GRÁFICOS DE GANTT GERADOS: 26
Capítulo IV 32
CONCLUSÃO 32

Capítulo I
INTRODUÇÃO

1. CONSIDERAÇÕES INICIAIS

Este trabalho apresenta uma proposta de solução para o problema de seqüenciamento, sob a condição de precedência e em geral, compartilhamento de recursos. Este contexto é encontrado em ambientes de produção, principalmente em células de remanufaturas, onde as oficinas estão preparadas para realizar tarefas de naturezas semelhantes, porém de tempos de duração diferentes e em acordo com o reparo a ser realizado.

A formulação escolhida para o a solução do problema foi a utilização de algoritmos de S*cheduling*. Particularmente, nesta será utilizada a formulação do problema de "*Resource Constrained*", um problema de seqüenciamento (*Scheduling)*, onde várias linhas de produção (*Jobs*) disputam os mesmos recursos (*Machines*). Assim, através do modelo e algoritmo proposto obtém-se a solução ótima do problema proposto.

2. OBJETIVO

O objetivo deste trabalho é apresentar uma proposta de solução para o problema de seqüenciamento chamado de "*Resource Constrained*", empregado em células de remanufaturas através de técnicas de *Branch and Bound*.

3. ESCOPO DO TRABALHO

O escopo deste trabalho é apresentar um algoritmo para a solução do problema de seqüenciamento, aplicado a uma célula de remanufatura de aeronaves. O problema utilizado como exemplo foi composto de dois Jobs (duas aeronaves) e seis máquinas (seis diferentes operações). Para exemplo, foram utilizadas duas redes de atividades (relação de precedência entre as máquinas) idênticas, porém, com tempos distintos para cada *Job*.

Considera-se que este exemplo seja genérico o suficiente para que, após a primeira implementação bem sucedida, apenas alterando alguns parâmetros, pode-se aproximar de maneira satisfatória do limite de capacidade do algoritmo, uma vez que este problema apresenta uma infinidade de combinações.

Após a exemplificação, será apresentada também uma tabela com o tempo de execução do problema utilizando de outros parâmetros de número de máquinas e *Jobs* pretendendo chegar ao limite operacional da capacidade do algoritmo.

4. CONDIÇÕES DE CONTORNO

Os exemplos utilizados neste trabalho são teóricos e se valem de simplificações e generalizações que para a aplicação em uma empresa deve ser analisadas com critério de forma a não inviabilizar o projeto de implementação do *Scheduling.*

5. JUSTIFICATIVAS

Para empresas que possuem processos produtivos com custos altos, quer seja fixos (equipamentos, mão de obra, tecnologia de processos) ou variáveis (impostos, matérias primas e insumos em geral), o investimento em sistemas de racionalização da produção representa uma linha de ação que pode trazer um grande retorno financeiro com investimento proporcionalmente baixo.

A aplicação de técnicas de seqüenciamento (S*cheduling*) visa atingir a melhor utilização dos recursos já instalados no sistema e resulta na correta seqüência de utilização dos recursos, otimização e eficiência do sistema por encontrar tempos de produção mais curtos para a finalização de todas as linhas (Jobs).

Segundo PINEDO(1995), a decisão tomada pelas técnicas de *Scheduling* apresenta um importante papel na maioria dos sistemas de produção, de manufatura e também em ambientes de processamento de informações, pois os recursos e as tarefas dentro das organizações são escassos e as técnicas de seqüenciamento lidam, exatamente com a alocação destes recursos no tempo, sendo um processo de tomada de decisão com a meta

Capítulo II
REVISÃO BIBLIOGRÁFICA

1. SCHEDULING

Segundo TUBINO(2000), as atividades de programação da produção estão divididas em três grupos hierarquicamente relacionados: Administração dos estoques, o seqüenciamento e a emissão e liberação das ordens. Especialmente, no caso de um sistema produtivo organizado por processos repetitivos em lote, caracterizado pela produção de um volume médio de itens padronizados em lote, onde cada lote segue uma série de operações que necessita ser programada à medida que as operações anteriores sejam concluídas. Assim, a administração de estoques define a quantidade e o momento em que os itens são necessários, cabendo ao seqüenciamento definir as prioridades na alocação dos recursos.

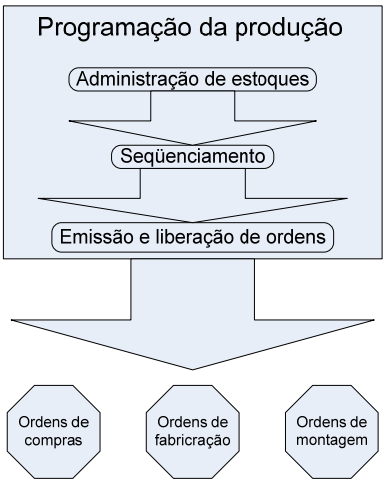

Figura 1 - Hierarquia das funções da programação da produção

Fonte: Tubino (pág.147)

Através de regras de seqüenciamento, pode-se buscar uma organização inicial de modo a planejar a ordem em que os pedidos serão atendidos e quando estarão disponíveis para o cliente.

Tabela 1– Regras de seqüenciamento

Sigla	Especificação	Definição
PEPS	Primeiro que entra primeiro que sai	Os lotes serão processados de acordo com sua chegada no recurso.
MTP	Menor tempo de processamento	Os lotes serão processados de acordo com os menores tempos de processamento no recurso.
MDE	Menor data de entrega	Os lotes serão processados de acordo com as menores datas de entrega.
IPI	Índice de prioridade	Os lotes serão processados de acordo com o valor da prioridade atribuída ao cliente ou ao produto.
ICR	Índice crítico	Os lotes serão processados de acordo com o menor valor de: (data de entrega – data atual) / tempo de processamento
IFO	Índice de folga	Os lotes serão processados de acordo com o menor valor de: $\frac{(\text{data de entrega} - \sum\text{tempo de processamento restante})}{\text{número de operações restante}}$
IFA	Índice de falta	Os lotes serão processados de acordo com o menor valor de: quantidade em estoque / taxa de demanda

Fonte: Tubino (pág. 157)

Segundo CORRÊA e CORRÊA(2004), a maneira como as ordens de produção são seqüenciadas, influencia no desempenho da operação tendo as seguintes repercussões estratégicas:

- Percentual de ordens de produção completadas no prazo
- Tempo médio de operação da ordem
- Níveis de estoques em processo na unidade produtiva
- Níveis de utilização dos recursos

Assim, como se trata de um problema multi-objetivo e complexo, CORRÊA E CORRÊA (2004) afirma não existir uma regra que maximize o desempenho da unidade produtiva em todos os aspectos, podendo, inclusive, uma determinada regra de seqüenciamento ser a melhor para uma condição de disponibilidade de capacidade produtiva, não se manter como a melhor quando se alteram essas condições. Portanto, hoje, as abordagens mais contemporâneas sugerem uma combinação de sistemas computacionais e matemáticas avançadas para a criação de programas que solucionem problemas de seqüenciamento complexos.

Desta maneira, pode-se obter uma série de possíveis programações, porém, devido à complexidade do problema apresentado, faz-se necessária a utilização de um sistema mais robusto e que possa ser avaliado através de uma função objetivo.

Segundo PINEDO(1995), um problema de *Scheduling* lida com a alocação de recursos escassos em tarefas no decorrer do tempo. É um processo de tomada de decisão com a meta de otimizar um ou mais objetivos.

O processo de seqüenciamento encontra-se inserido no macro-processo de planejamento e controle de produção no ponto onde, dado o planejamento de capacidade e de materiais, faz-se necessária a tomada de decisão de como arranjar as ordens de produção para que possa se utilizar os recursos com maior eficiência. Portanto, os pedidos já são emitidos na seqüência definida como ótima para dado cenário. A figura a seguir exemplifica o processo de Planejamento e Controle de Produção, bem como mostra o ponto onde ocorre o *Scheduling* num sistema de manufatura.

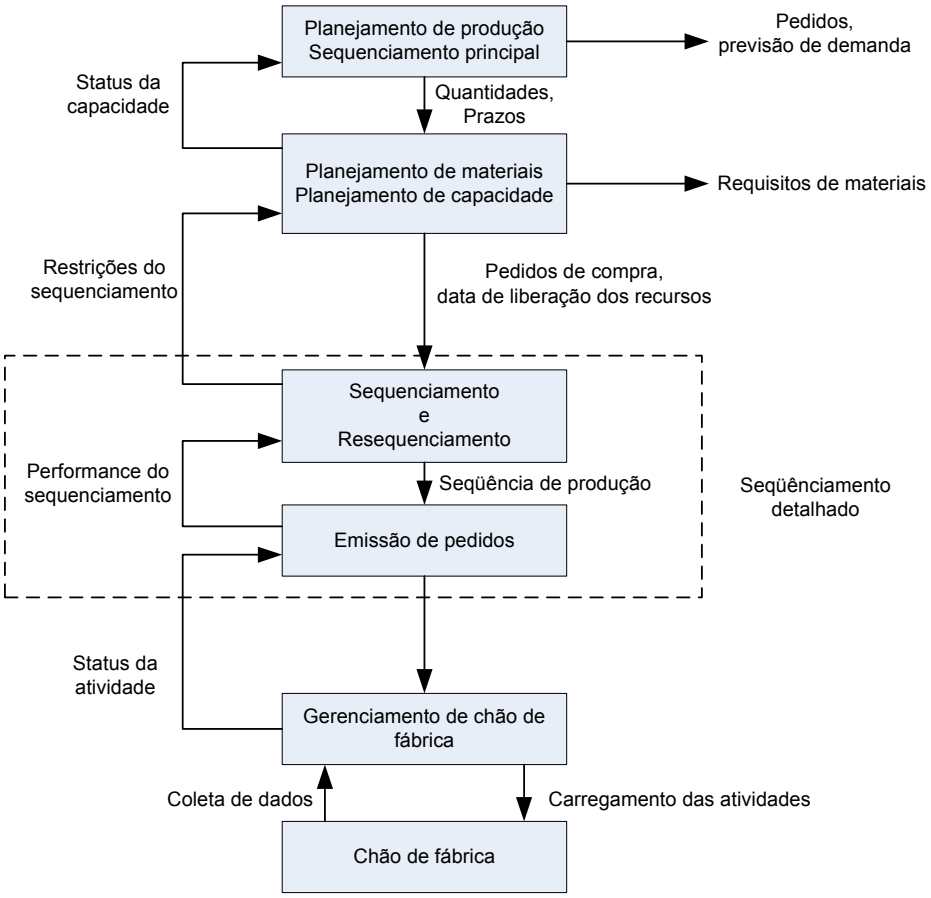

Figura 2 – Diagrama de fluxo de informação em um sistema de manufatura

Fonte: Pinedo (pág. 5)

Já num sistema de serviços, maior dinamismo é dado ao sistema de informações, uma vez que não existe planejamento mestre de produção e a análise da capacidade é feita paralelamente a produção (mesmo que feita antes, não passa de uma previsão). Portanto fatores mercadológicos como aceitação no mercado e preços praticados interferem de maneira muito mais direta no seqüenciamento das tarefas que em modelos de manufatura. A seguir é apresentado um modelo esquemático da utilização de técnicas de seqüenciamento em um sistema de serviços.

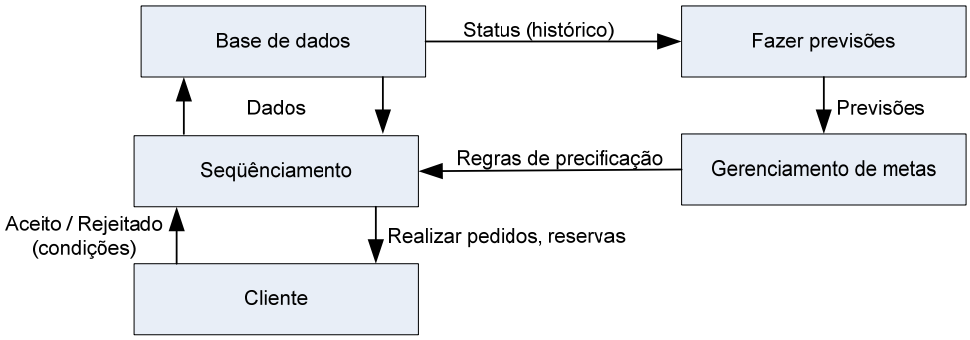

Figura 3 - Diagrama de fluxo de informação em um sistema de serviços

Fonte: Pinedo (pág. 6)

Os termos recursos e tarefas podem assumir diversos papéis: pistas de decolagem em aeroportos, máquinas em um chão de fábrica, operários em uma planta de construção ou até mesmo unidades de processamento em um ambiente computacional. Nestes exemplos, as tarefas são: processos produtivos, decolagem e pousos, estágios em um projeto de construção ou execução de programas de computador. Assim, pode-se ter como objetivos minimizar o atraso de cada tarefa, minimizar o atraso da ultima tarefa, ou ainda iniciar todas elas o mais cedo possível.

Uma vez escolhidos os recursos que serão utilizados e quais as atividades recolherão os recursos, deve-se definir qual o objetivo da otimização, para a medida da qualidade da solução proposta, utiliza-se a função objetivo que melhor se adapte aos anseios dos *Stakeholders* do projeto.

Segundo Pinedo, as principais funções objetivo disponíveis para avaliação da solução são:

1. O menor tempo de finalização (*The Makespan* – C_{max})
2. O máximo de pontualidade (*The maximum Lateness* - L_{max})
3. O número de jobs atrasados (*The Number of Tardy Jobs* - $\sum U_j$)
4. Tempo de finalização total ponderado (*The Total Weighted Completion Time* - $\sum w_j U_j$)
5. O total de pontualidade (*The total Tardiness* - $\sum T_j$)
6. O total de pontualidade ponderada (*The total Weighted Tardiness* - $\sum w_j T_j$)

Na figura a seguir, pode ser analisada a precedência hierárquicas das funções objetivo que podem ser otimizadas segundo as técnicas de S*cheduling*. As setas representam o grau crescente de complexidade do problema sendo as superiores, situações genéricas das imediatamente inferiores.

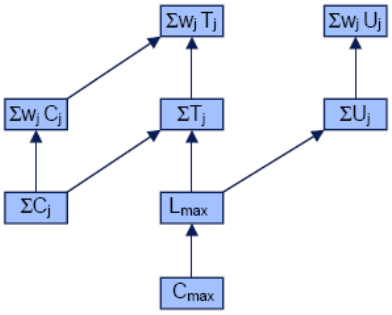

Figura 4 – Hierarquia de complexidade em problemas determinísticos de seqüenciamento(funções objetivo)

Fonte: Pinedo (pág. 27)

As principais literaturas relativas ao assunto aponta como principais as seguintes categorias de problemas de S*cheduling*:

1. Máquinas Simples (1)

   É considerado o caso mais particular de todas as categorias subseqüentes, pois analisa somente o processamento de uma única máquina e diversos Jobs que podem possuir diferentes tempos de processamento.

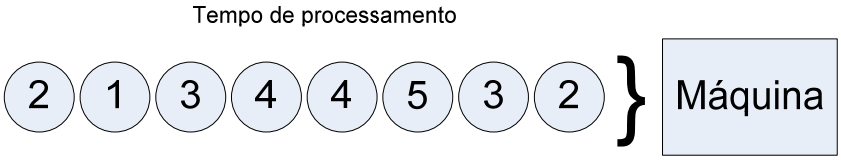

Figura 5 – Máquinas Simples

Fonte: Autor

2. Máquinas Paralelas (P_m)

   É uma primeira generalização do modelo de máquina simples, e um caso particular do Flow Shop. Trata-se da possibilidade de duas máquinas independentes processarem simultaneamente, podendo ou não haver a utilização de ambas, em qualquer ordem.

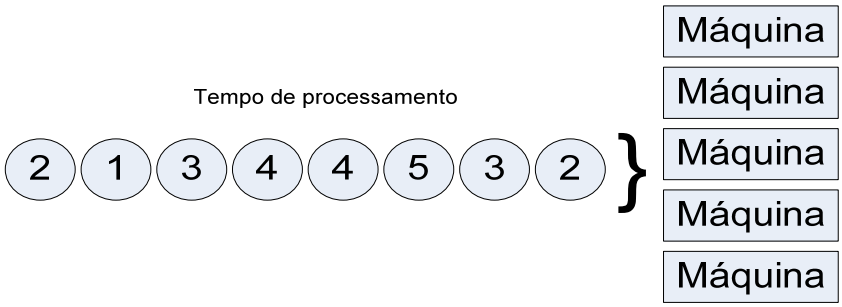

Figura 6 – Máquinas Paralelas

Fonte: Autor

3. Flow Shops (P_m)

Um modelo é considerado Flow Shop, se os Jobs possuem uma seqüência pré-definida de máquinas a ser percorrida durante o processamento e comum a todos os Jobs.

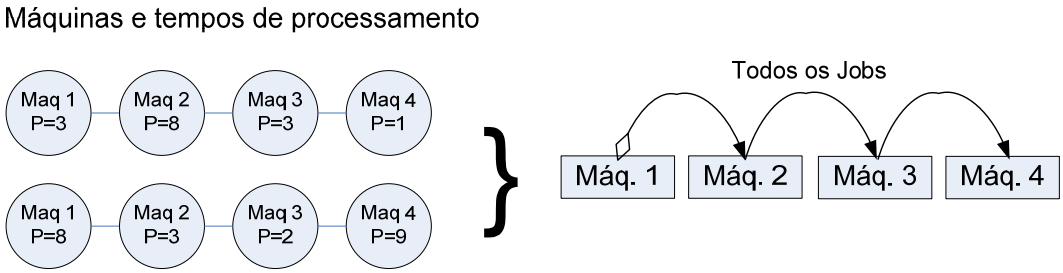

Figura 7 – Flow Shops

Fonte: Autor

4. Job Shops (J_m)

Constitui de um modelo de operação simultânea, porém, sem ordem pré-definida comum a todos os Jobs. Assim, podem percorrer diferentes seqüências de máquinas dado que cada um já possui sua rota.

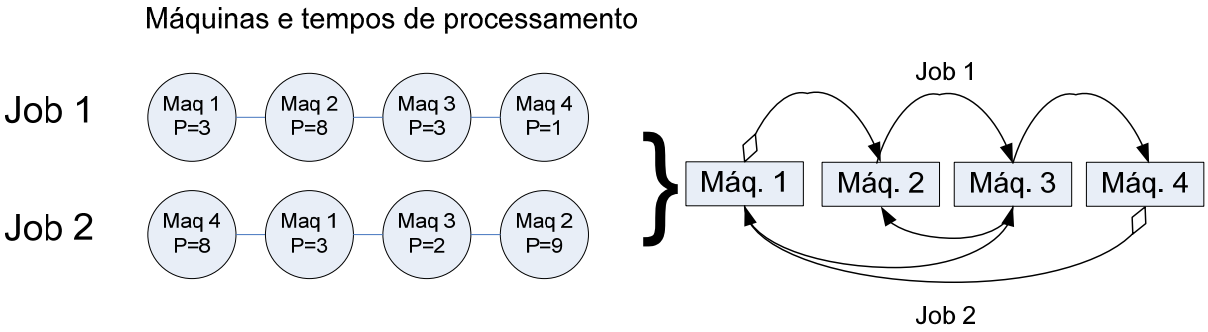

Figura 8 – Job Shops

Fonte: Autor

5. Resource Constrained Scheduling Project (RCSP)

Trata-se do modelo a ser especificamente estudado neste trabalho, pela sua flexibilidade em termos de poder realizar atividades simultâneas dentro de um mesmo *Job* torna-se melhor aplicável a alguns contextos produtivos, pois pode-se modelar cada parte do *Job* que será processada individualmente e ao final, agrupada.

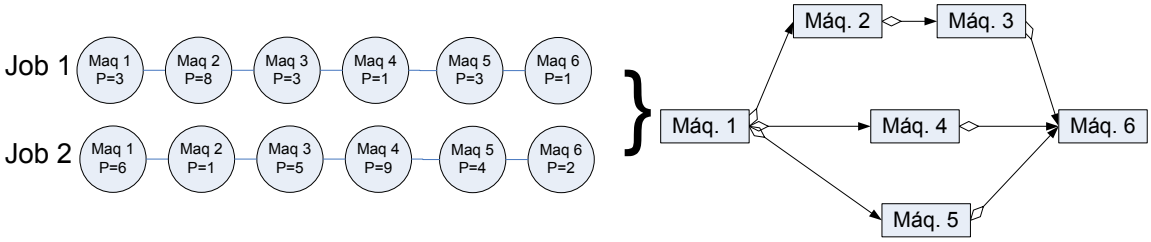

Figura 9 – RCSP

Fonte: Autor

6. Open Shops (O_m)

O modelo Open Shop permite o máximo em termos de flexibilidade, uma vez que a rota do Job será definida de acordo com os critérios de otimização da função objetivo. A entidade chamada *"The Scheduler"* irá se encarregar de definir qual a melhor seqüência de máquinas irá otimizar o processo.

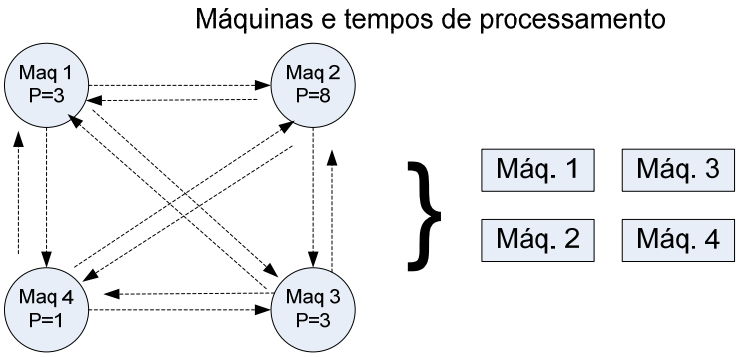

Figura 10 – Open Shops

Fonte: Autor

7. Nível de complexidade

Os modelos de *schedulling* estão arranjados por grau de complexidade da seguinte maneira:

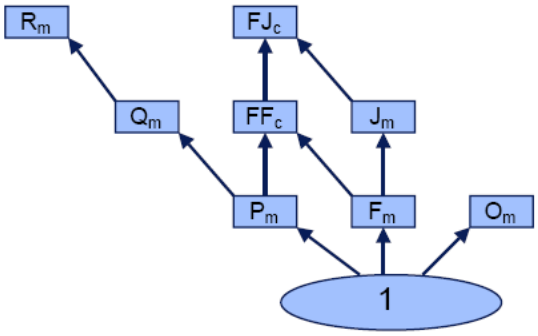

Figura 11 – Hierarquia de complexidade em problemas determinísticos de seqüenciamento (ambientes de produção)

Fonte: Pinedo (pág. 27)

Onde:

1 Máquinas simples

Pm Processamento paralel

Fm Flow-Shop

Om Open-Shop

Jm Job-Shop

FFc Flexible Flow-Shop

FJc Flexible Job-Shop

Qm Máquinas paralelas com diferentes desempenhos

Rm Máquinas diferentes em processamento paralelo

2. PROJETO

Neste trabalho o conceito de projeto abordado será o de um projeto como um tipo de processo em manufatura. Segundo SLACK(1996), um processo de manufatura pode ser dividido em cinco tipos escritos em ordem crescente de volume e decrescente de variedade:

- Processos de projeto;
- Processos de *jobbing;*
- Processos em lotes ou bateladas;
- Processos de produção em massa;
- Processos contínuos;

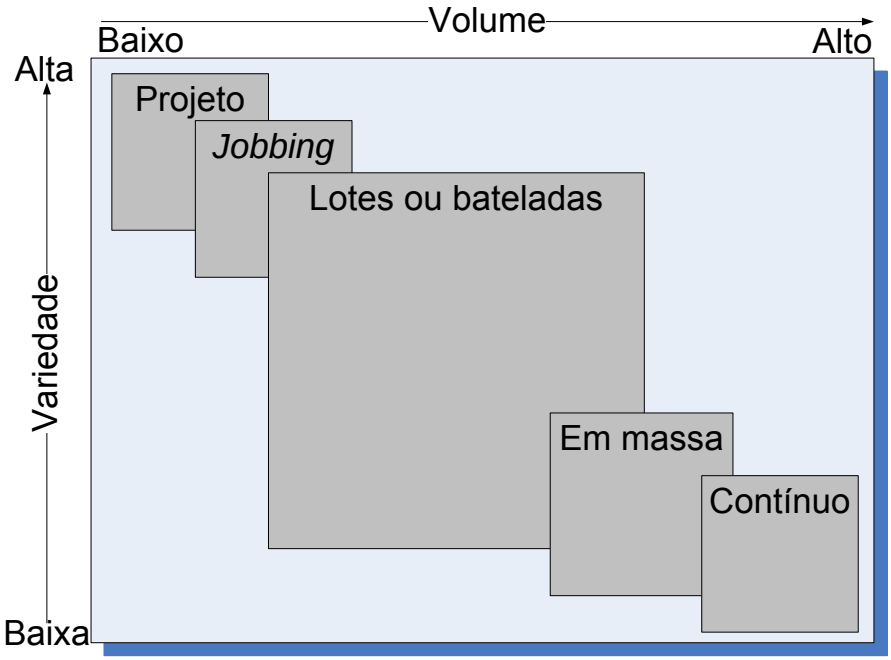

Figura 12 - Tipos de processos em operações de manufatura

Fonte: Slack (pág. 129)

Cada tipo corresponde a uma maneira diferente de organizar a produção que influencia, desde as estratégias adotadas até a maneira como a disposição física das máquinas será realizada.

Dentre as principais características do problema abordado por este trabalho estão o elevado nível de customização do processo, uma vez que cada aeronave poderá conter processos diferentes, portanto, haverá um planejamento prévio de como será abordado cada cenário encontrado na linha de remanufatura. Por outro lado, o compartilhamento dos recursos é característica típica do processo tipo *Jobbing*, já que existe um certo grau de repetição nos processos em geral.

Pode-se, portanto, posicionar o escopo deste trabalho na zona de intersecção entre os tipos Projeto e *Jobbing*, com predomínio do tipo Projeto.

3. PROGRAMAÇÃO LINEAR

O problema geral de programação linear visa otimizar (maximizar ou minimizar) uma função linear de variáveis, chamada de "função objetivo", sujeita a uma série de equações ou inequações lineares, chamadas restrições. A formulação do problema a ser resolvido por programação linear segue alguns passos básicos.

- Deve ser definido o objetivo básico do problema, ou seja, a otimização a ser alcançada. Por exemplo, maximização de lucros, ou de desempenhos, ou de bem-estar social; minimização de custos, de perdas, de tempo. Tal objetivo será representado por uma função objetivo, a ser maximizada ou minimizada;
- Para que esta função objetivo seja matematicamente especificada, devem ser definidas as variáveis de decisão envolvidas. Por exemplo, número de máquinas, a área a serem exploradas, as classes de investimento à disposição etc. Normalmente, assume-se que todas estas variáveis possam assumir somente valores positivos;
- Estas variáveis normalmente estão sujeitas a uma série de restrições, normalmente representadas por inequações. Por exemplo, quantidade de equipamento disponível, tamanho da área a ser explorada, capacidade de um reservatório, exigências nutricionais para determinada dieta etc.

Todas essas expressões, entretanto, devem estar de acordo com a hipótese principal da programação linear, ou seja, todas as relações entre as variáveis devem ser lineares. Isto implica proporcionalidade das quantidades envolvidas. Esta característica de linearidade pode ser interessante no tocante à simplificação da estrutura matemática envolvida, mas prejudicial na representação de fenômenos não lineares (por exemplo, funções de custo tipicamente quadráticas).

O problema geral de programação linear pode ser definido por:

Maximizar (ou minimizar)

$$Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \quad (1)$$

Sujeito a:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \text{ (ou } \geq \text{, ou =)} \quad (2)$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2 \text{ (ou } \geq \text{, ou =)} \quad (3)$$

$$\ldots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m \text{ (ou } \geq \text{, ou =)} \quad (4)$$

$$x_1, x_2, \ldots, x_n \geq 0 \quad (5)$$

Figura 13 – Formulação de um problema de PL

Fonte: Hillier (pág. 34)

3.1. SIMPLEX

Segundo HILLIER(2002), o método simplex é o procedimento geral para resolver problemas de programação linear. Na maioria das vezes, o método é utilizado através de uma implementação computacional devido ao grande número de contas, cabendo somente a pequenos problemas a utilização manual. Trata-se de um algoritmo de processamento sistemático repetitivo seguidamente, até que o resultado desejado seja obtido. Assim, o método simplex pode ser simplificado no seguinte diagrama:

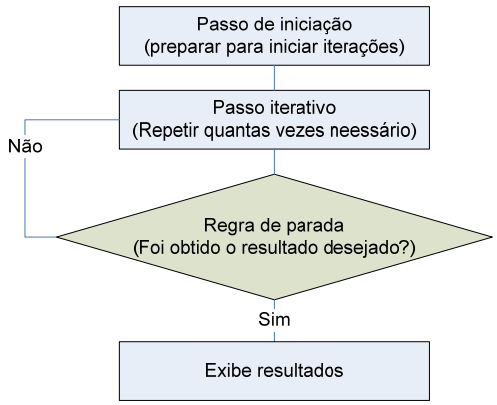

Figura 14 – Estrutura dos Algorítmos

Fonte: Hillier (pág. 45)

Seu funcionamento baseia-se em identificar uma solução básica viável inicial no Passo de inicialização, após feito, o Passo iterativo realiza uma busca de melhores soluções adjacentes, ou seja, todo passo é realizado entre pontos adjacentes para garantir que o algoritmo tenha uma referência anterior de comparação e a Regra de parada que estipula a parada das iterações quando não houver nenhuma solução adjacente melhor.

Cada solução é avaliada através da Função Objetivo que pode buscar o valor máximo (problema de maximização) ou valor mínimo (problema de minimização) e a cada iteração a possível solução é testada nesta função e caso apresente um valor superior (ou inferior, dependendo do tipo de problema) a decisão entre realizar nova iteração ou parar para exibição dos resultados é tomada.

4. PROGRAMAÇÃO INTEIRA

Devido à vasta aplicabilidade que a Programação Linear possui como ferramenta de auxílio à tomada de decisão, em alguns problemas, não se pode simplesmente aplicar a metodologia de sua resolução (Simplex, Método Gráfico ou Analítico), uma vez que além das restrições usuais, o tipo de variável apresenta uma restrição primordial. O fato de se trabalhar somente com variáveis inteiras amplia a gama de aplicações da Programação Linear, uma vez que habilita a mesma a trabalhar com designação de pessoas (não se pode ter meia pessoa) ou até mesmo discretizar a variável tempo, para se trabalhar com dimensões tecnicamente mensuráveis em aparelhos do dia-a-dia (segundos, minutos, horas, dias, etc.).

Por outro lado, perde-se em performance, pois computacionalmente é sensivelmente mais caro, em termos de números de cálculos, a solução de um problema com variáveis inteiras. Além de resolver vários problemas de Programação Linear dentro de um único problema de Programação Linear Inteira, forma-se uma estrutura de árvore para que, baseado nos resultados obtidos, possa-se eliminar os ramos indesejáveis e manter aqueles que apresentam bons resultados, até que se chegue a um resultado ótimo.

Esta técnica genérica de resolução de problemas de Programação Linear Inteira é chamada de *Branch and Bound* e será a utilizada no algoritmo deste trabalho.

4.1. BRANCH AND BOUND

O problema de *Scheduling* em questão será formulado conforme técnicas de *Branch and bound*, pois se trata de um problema de Programação Linear Inteira (dado que parte-se do pressuposto de que o tempo é uma variável discreta). Assim, a solução ótima poderá ser encontrada através de algum procedimento de enumeração. Este, porém, é geralmente impraticável de se realizar exaustivamente sua completa listagem.

Para, então, definir a solução ótima, na técnica de *Branch and bound* deve-se definir um limite superior (melhor valor da função objetivo até o momento), subdivide-se o conjunto de todas as soluções viáveis em diversos subconjuntos e, para cada um, obter um limite inferior para o valor da função objetivo das soluções deste. Para os subconjuntos cujo limite inferior exceda (no caso de minimização) o limite superior previamente estabelecido, serão "podados", ou seja, estarão fora para posteriores análises. Assim, através de um processo iterativo, redefine-se o limite superior e realizam-se novos cortes na árvore de alternativas até que se chegue a um valor mínimo da função (mais uma vez pressupondo minimização), e atinja o resultado ótimo.

No problema em estudo, foi utilizada a abordagem *Branch and Bound* com o intuito de se obter uma solução ótima, com o teste do limite de capacidade do algoritmo, uma vez que dependendo da natureza do problema e da quantidade de conflitos a ser resolvida, esta abordagem é suficiente e apropriada, pois garante a solução ótima, porém, existem casos que devido à configuração do mesmo, parte-se para a utilização de algoritmos genéticos, métodos de meta-heurística (*Simulated Annealing,* Busca Tabu, etc.), pois, apesar de não garantir o resultado ótimo, são mais rápidos, e os resultados sub-ótimos já são suficientes para sua aplicação.

Capítulo III
DESCRIÇÃO DO PROBLEMA

1. APRESENTAÇÃO DO PROBLEMA

Segundo DAVIS, et. al.(2006) a prática de Gerenciamento de Projetos tende para a utilização do seqüenciamento através do caminho crítico. Este método é com maior freqüência apresentado como uma abordagem filosófica que se baseia na Teoria das Restrições que através de implementações práticas propriamente ditas. Assim, quando um recurso se depara com o conflito de poder ser alocado para diversos projetos simultaneamente, os seguintes passos devem ser adotados:

a. Identificar o recurso em conflito
b. Eliminar o conflito, abrindo para teste as possibilidades (árvore decisória)
c. Sujeitar as decisões não conflitantes às diferentes opções geradas no passo anterior
d. Seguir na direção ao próximo recurso
e. Repetir os passos anteriores caso novo conflito se configure

Através desta lógica, pode-se chegar ao seqüenciamento ótimo, porém devido à dimensão dos problemas de S*cheduling*, faz-se necessária a aplicação de um método numérico computacional para sua execução.

A classe de problemas RCSP (Resource Constrained Scheduling Problem), segundo DAVIS, et. al. (2006), foi exaustivamente documentado e é conhecido por ser NP-Dificil (Hall, Kubiak & Sethi, 1991; Hoogeven & van de Velde, 1991). Este trabalho visa à implementação do algoritmo *Branch and Bound* proposto por Demeulemeester e Herroelen (1992). Assim, através da minimização do tempo total de finalização *(Makespan)*, estar-se-á maximizando a disponibilidade dos recursos, portanto, um sistema de produção atrativamente eficiente.

Por se tratar de um problema com *Branch and Bound*, a solução ótima é construída através da análise dos recursos em conflito e a bifurcação do mesmo em alternativas que avaliará o resultado quando seqüenciado diferentes Jobs em todas as possíveis ordens.

Na árvore de busca do *Branch and Bound*, cada nó corresponde a uma atividade onde potencialmente ocorreu um conflito de recursos.

A abordagem de múltiplos *Jobs* e múltiplos recursos sendo disputados torna a implementação mais genérica, pois poder-se-á adaptá-la, através de ajustes de parâmetros, para qualquer tipo de problema da classe RCSP.

2. APLICAÇÃO

Uma aplicação da teoria discorrida até este ponto, será apresentada em forma de um modelo que retrata uma situação prática onde o problema de *Scheduling* pode trazer aumento da produtividade e de utilização dos recursos.

O problema inicial proposto será de seqüenciar dois *Jobs* contendo um total de seis máquinas em cada um deles, porém, 4 delas são concorrentes. Por não se tratar um problema de Job-Shop clássico, o grafo será composto conforme a figura a seguir:

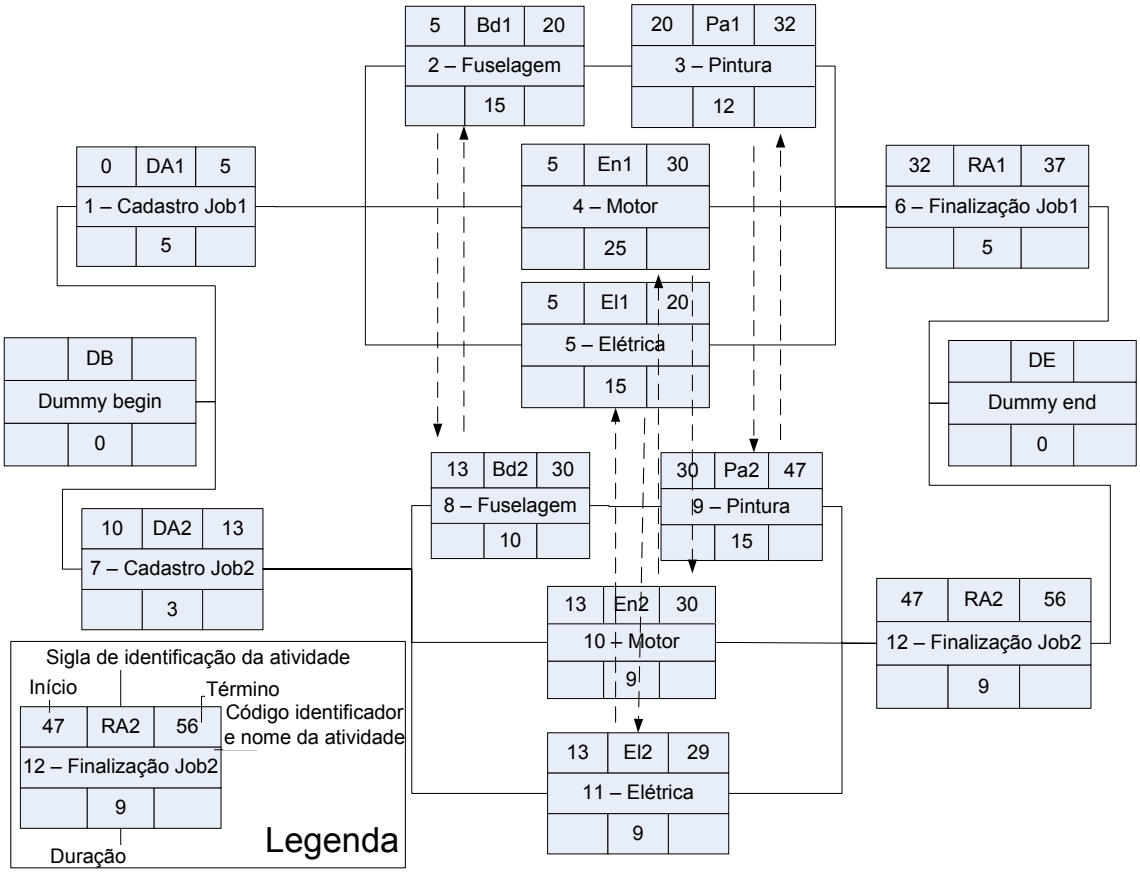

Figura 15 – Modelo do exemplo

Fonte: Davis (2006)

Percebe-se que dentro do mesmo *Job*, existem máquinas que podem funcionar simultaneamente. Isto pode ser facilmente compreendido, uma vez que numa aeronave, a pintura da fuselagem pode ser preparada simultaneamente com a instalação do sistema de refrigeração, ou fixação dos assentos.

A modelagem do problema, que se apresenta como de Programação Linear Inteira, é estabelecida da seguinte forma:

- Função Objetivo: minimizar o período total de operações dentro da oficina (*Makespan*);

- Restrições de Precedência: monta o grafo relativo a cada *Job*, indicando a precedência entre as atividades;
- Restrições Disjuntivas: analisa em qual seqüência a ocupação das máquinas compartilhadas deve realizar (relação de precedência entre os Jobs);
- Atividades fantasmas: Serão acrescentadas ao modelo duas variáveis fantasmas (*Dummy*): uma no início (Y_{db}) e uma ao término (Y_{de}) de forma a se montar um grafo com único início e único final;

Matematicamente:

Tabela 2 – Modelo utilizado

Minimizar		
Y_{de}		(6)
Sujeito a:		
Restrições Iniciais:		
$Y_{ik} \geq 0$		(7)
Restrições de precedência:		
$Y_{ik} - Y_{ih} \geq P_{ih}$	se O_{ih} precede O_{ik}	(8)
Restrições disjuntivas:		
$Y_{pk} - Y_{ik} + K(1-B_{ipk}) \geq P_{ik}$	$B_{ipk} = 1$, se O_{ik} precede O_{pk}	(9)
$Y_{ik} - Y_{pk} + K(B_{ipk}) \geq P_{pk}$	$B_{ipk} = 0$, caso contrário	(10)
Onde:		
i, p são aeronaves (*Jobs*)		
k, h são atividades (máquinas)		
O_{ik} é a operação da aeronave i na atividade k		
Y_{ik} é o tempo que a aeronave i inicia a operação na atividade k		
P_{ik} é a duração da operação da aeronave i na atividade k		
B_{ipk} é a variável binária que indica se a aeronave i ou p inicia primeiro sua operação na atividade k		
K é um número grande que garanta as restrições disjuntivas		

Fonte: Autor

Para que se possa arranjar as equações acima citadas de forma adequada à solução do problema, pode-se utilizar de recursos matemáticos para modelar o mesmo. A entrada de dados é dividida em Matriz de Adjacência, que é uma matriz representativa de um grafo direcional e que cada valor representa um arco entre os nós que foram padronizados neste trabalho como sendo a origem do arco, o índice da linha da matriz e o destino do arco, o

índice da coluna. Assim, a figura abaixo representa a Matriz de Adjacência do exemplo em estudo.

Origem (To) \ Destino (From)	1	2	3	4	5	6
1	0	1	0	1	1	0
2	0	0	1	0	0	0
3	0	0	0	0	0	1
4	0	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

Figura 16 – Matriz de adjacência (JO)

Fonte: Autor

Outra matriz de entrada de dados é a Matriz de tempo de Processamento, que abriga em suas células a duração do processamento dos Jobs (colunas) em função da máquina (linhas).

Machines \ Jobs	J1	J2
1	5	3
2	15	10
3	12	15
4	25	9
5	15	9
6	5	9

Figura 17 – Matriz de Tempo de Processamento (JM)

Fonte: Autor

Finalmente, a ultima matriz de entrada de dados é a Matriz de Conflitos que delimita quais as máquinas serão compartilhadas durante o processo de seqüenciamento. Esta matriz torna o sistema mais flexível, uma vez que durante a parametrização pode-se definir se o seqüenciamento poderá contar com processamentos independentes ou se serão compartilhados.

Máquinas Compartilhadas
Máquina
3
4
5
6

Figura 18 – Matriz de conflitos (Shared)

Fonte: Autor

Capítulo III
FERRAMENTA DE OTIMIZAÇÃO

1. INTRODUÇÃO

Para a análise e comparação dos resultados atingidos por este trabalho, foi elaborada uma ferramenta de otimização que é composta do algoritmo desenvolvido em MATLAB integrada a uma planilha de Excel facilitando a análise dos resultados e a parametrização do problema a ser resolvido.

2. O ALGORITMO

O algoritmo desenvolvido foi basicamente um tratamento de dados para que fossem montadas as matrizes adequadas e inseridas no solver TORSCHE que por sua vez realizou a solução, sobre a condição de ser um problema de variáveis inteiras (tempos de processamento e as variáveis binárias).

Esquematicamente, foi aplicado o seguinte procedimento:

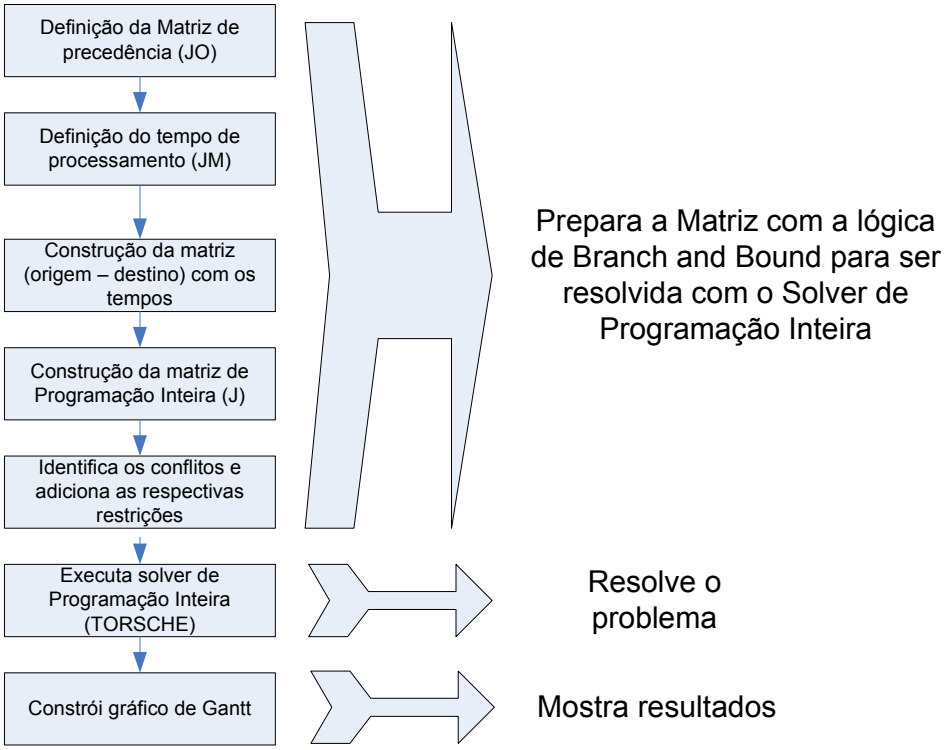

Figura 19 – Lógica do algoritmo

Fonte: Autor

3. ANÁLISE DE RESULTADOS

A análise dos resultados se dá em processos de validação do modelo e observação do resultado.

Para uma validação adequada do modelo, duas matrizes são geradas dinamicamente para que sejam confrontados os dados de entrada com a matriz gerada pelo sistema. Uma delas é a Matriz Origem-Destino, a qual identifica qual o Job, Origem, Destino e Tempo de Processamento. Esta matriz pode ser lida como a tradução da Matriz de Adjacência citada anteriormente.

JOB	ORIGEM	DESTINO	TEMPO DE PROCESSAMENTO
1	1	2	5
1	2	3	0
1	2	5	0
1	2	6	0
1	3	4	15
1	4	7	12
1	5	7	25
1	6	7	15
1	7	8	5
2	1	2	0
2	2	3	3
2	2	5	3
2	2	6	3
2	3	4	10
2	4	7	15
2	5	7	9
2	6	7	9
2	7	8	0

Figura 20 – Matriz de origem-destino

Fonte: Autor

Outra matriz utilizada para a validação do modelo é a Matriz da Programação Inteira, esta é a matriz de entrada na função de solução da programação inteira e pode ser utilizada para uma checagem final do funcionamento do algoritmo em fase de preparação. Nesta matriz, as equações Conjuntivas e Disjuntivas são representadas bem como a função objetivo.

Esta matriz contém as informações necessárias para a solução do problema. Cada linha representa uma restrição que podem ser Conjuntivas ou Disjuntivas, conforme o modelo matemático e as colunas representam as variáveis de decisão envolvidas no problema e as quais representarão a solução do problema. Assim, os campos com valor -1

representa a atividade de destino referente aquela restrição, os com valor 1 representa a atividade de origem. O Valor 1000 ou -1000 correspondem a atuação do valor K conforme explicado na modelagem matemática, caracterizando o *Branch and Bound* das restrições disjuntivas.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Ydb	Y1,1	Y1,2	Y1,3	Y1,4	Y1,5	Y1,6	Y1,7	Y1,8	Y2,1	Y2,2	Y2,3	Y2,4	Y2,5	Y2,6	Y2,7	Y2,8	Yde	Bin3	Bin4	Bin5	Bin6
0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	-1	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0
-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0
0	0	0	-1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	-1000	0	0	0
0	0	0	1	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1000	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	-1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	-1000	0	0
0	0	0	0	1	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1000	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	-1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	-1000	0
0	0	0	0	0	1	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1000	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	-1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	-1000
0	0	0	0	0	0	1	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1000
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figura 21 – Matriz da programação inteira (J)

Fonte: Autor

Uma vez resolvido o problema, a Matriz Solução é gerada, com os valores referentes ao momento de início de cada atividade para que se tenha o seqüenciamento ótimo do modelo estudado. As unidades das grandezas na coluna de “Início de Execução” estarão na mesma grandeza da Matriz Tempo de Processamento, podendo ser dias, horas, minutos, etc.

Atividade	Início
Ydb	0
Y1,1	0
Y1,2	12
Y1,3	13
Y1,4	28
Y1,5	12
Y1,6	12
Y1,7	40
Y1,8	45
Y2,1	0
Y2,2	0
Y2,3	3
Y2,4	13
Y2,5	3
Y2,6	3
Y2,7	36
Y2,8	36
Yde	45
Bin3	0
Bin4	0
Bin5	0
Bin6	0

Figura 22 – Matriz Solução

Fonte: Autor

Uma outra forma de exposição dos resultados gerados pelo algoritmo é através do Gráfico de Gantt, no qual as barras de cores semelhantes representam o momento de ocupação do mesmo Job, os eixos são: tempo e índice da máquina.

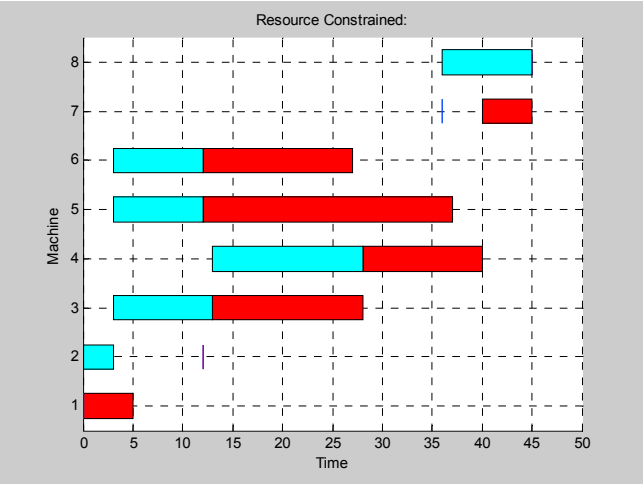

Figura 23 – Gráfico de Gantt relativo ao seqüenciamento dos 2 Jobs exemplificado

Fonte: Autor

4. QUADRO COMPARATIVO DE PERFORMANCE

Os testes foram realizadas em ambiente doméstico, em um computador portátil com processador AMD Turion 64, 1Gb de memória RAM, e sobre plataforma Windows XP. Assim, computadores dedicados e com maior capacidade pode melhorar um pouco a performance do algoritmo, porém, dificilmente aumentará sensivelmente a capacidade do mesmo.

Tabela 3 – Comparação de Performance

Número de *Jobs*	Número de Atividades por *Job*	Número de Conflitos	Tempo gasto Processamento (segundos)
2	31	17	0,0150000
3	31	51	0,3430000
4	31	102	2,9530000
5	31	170	50,828000
6	31	255	322,34400
7	31	357	9813,6000

Fonte: Autor

O gráfico a seguir demonstra o crescimento exponencial do tempo de processamento gasto do algoritmo com o aumento do numero de *Jobs*, conforme a tabela mostrada anteriormente, os testes foram realizados mantendo constante o número de atividades por *Job*, para criar um parâmetro de comparação.

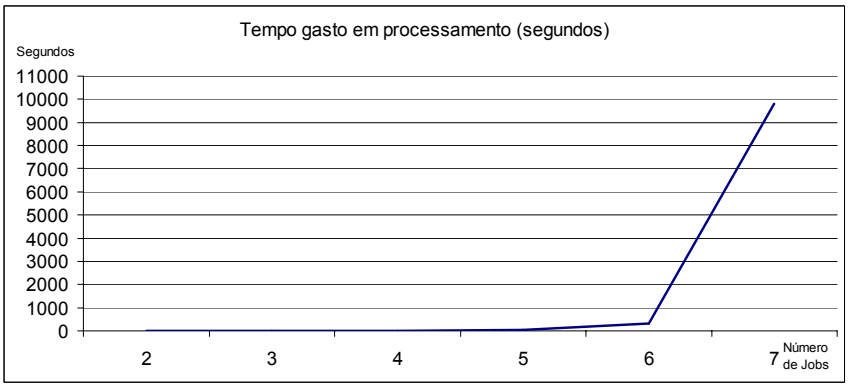

Figura 24 - Curva de tempo de processamento x número de Jobs

Fonte: Autor

5. GRÁFICOS DE GANTT GERADOS:

Segundo SLACK, & OUTROS (1996), o Gráfico de Gantt é uma ferramenta de programação inventada por H. L. Gantt, em 1917, que representa o tempo como uma barra no gráfico. A seguir serão apresentados os resultados do teste de performance representado no quadro acima, o grau crescente de complexidade, aqui representado, resulta em um tempo exponencialmente maior de processamento.

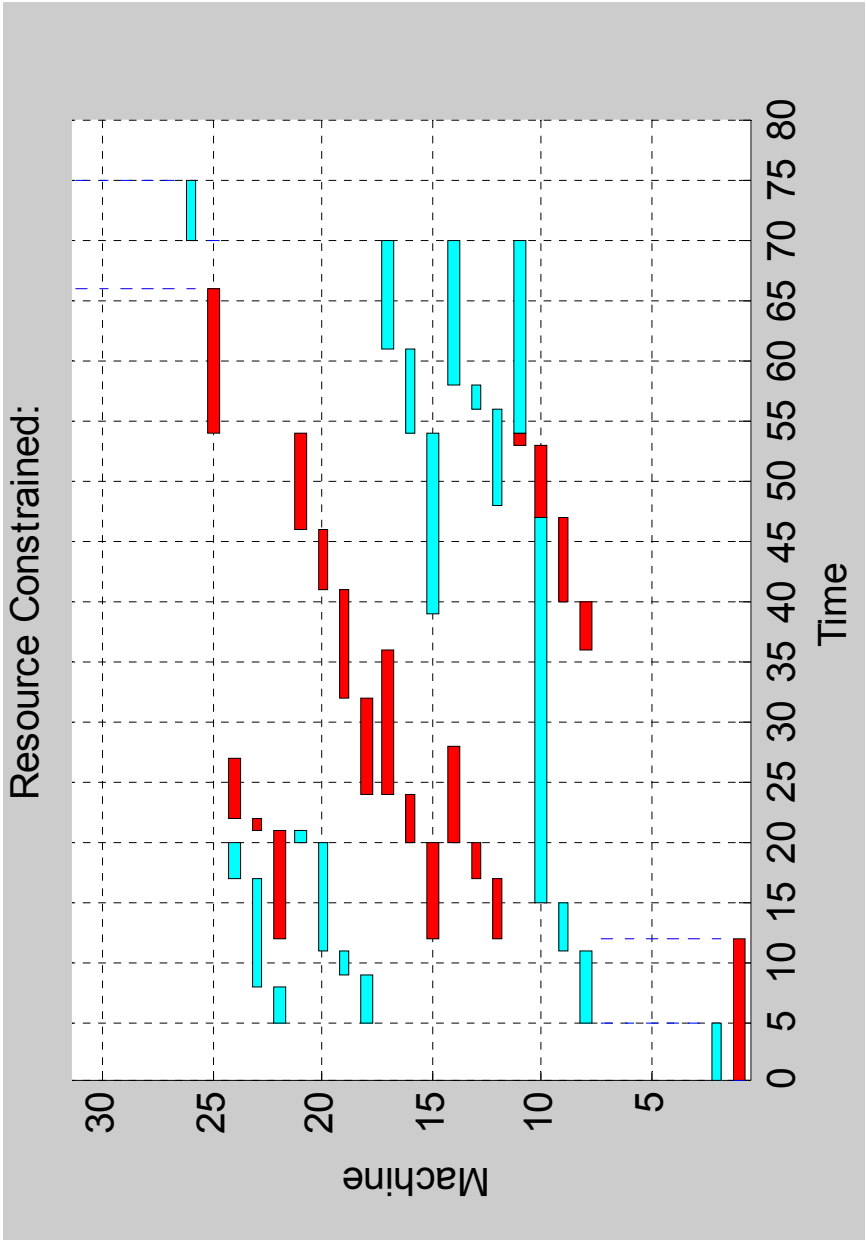

Figura 25 – 2 Jobs

Fonte: Autor

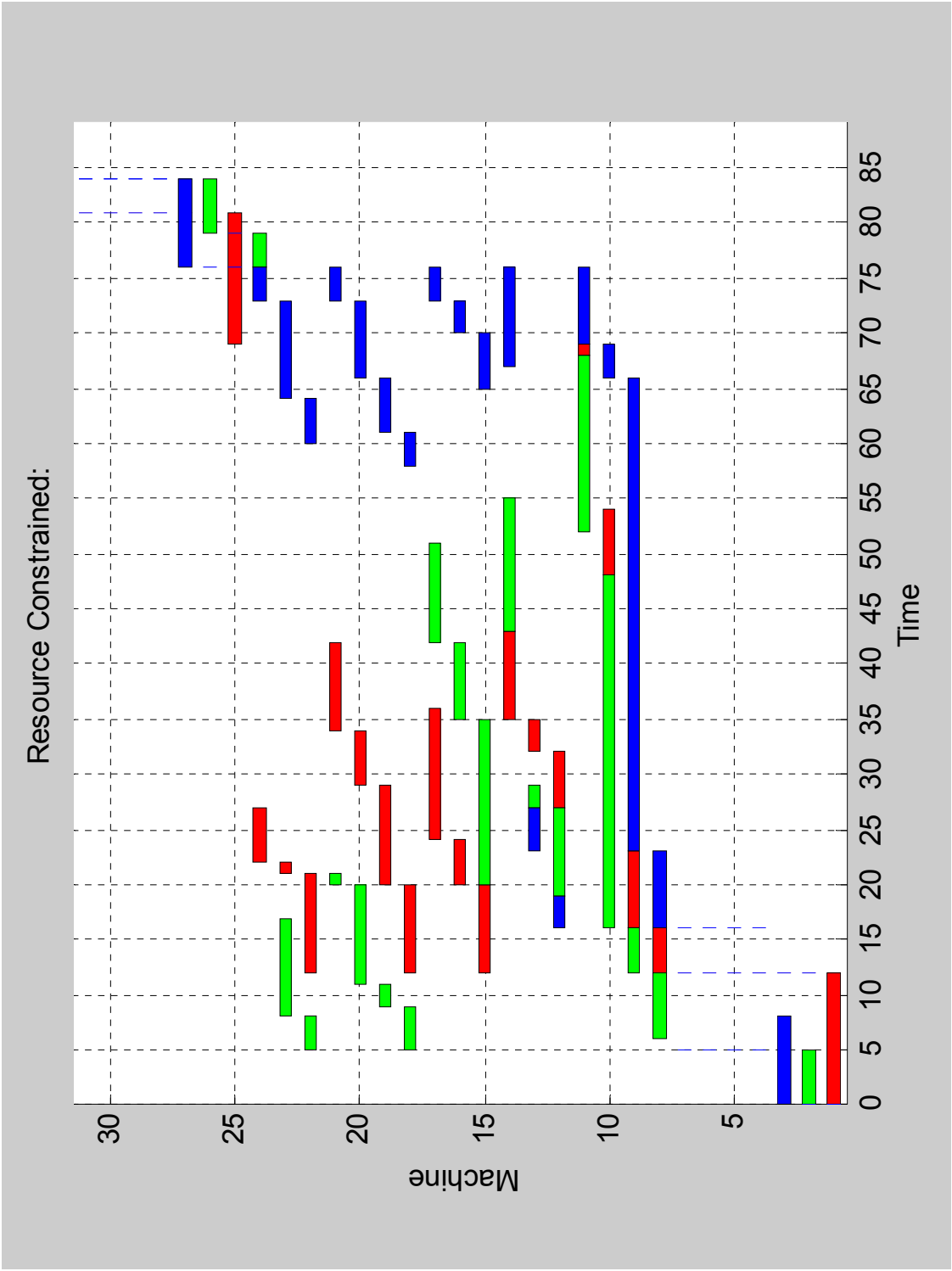

Figura 26 – 3 Jobs

Fonte: Autor

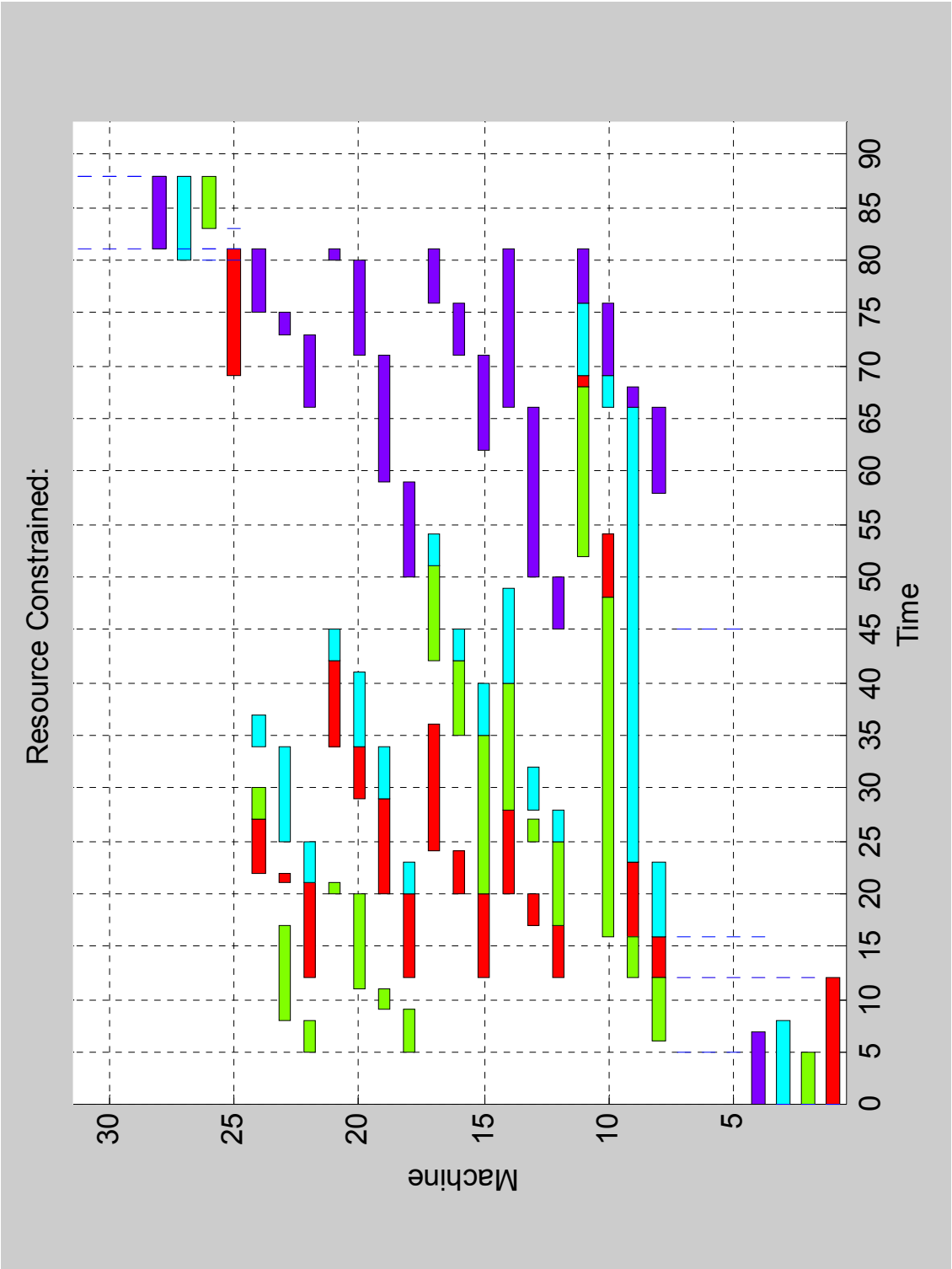

Figura 27 – 4 Jobs

Fonte: Autor

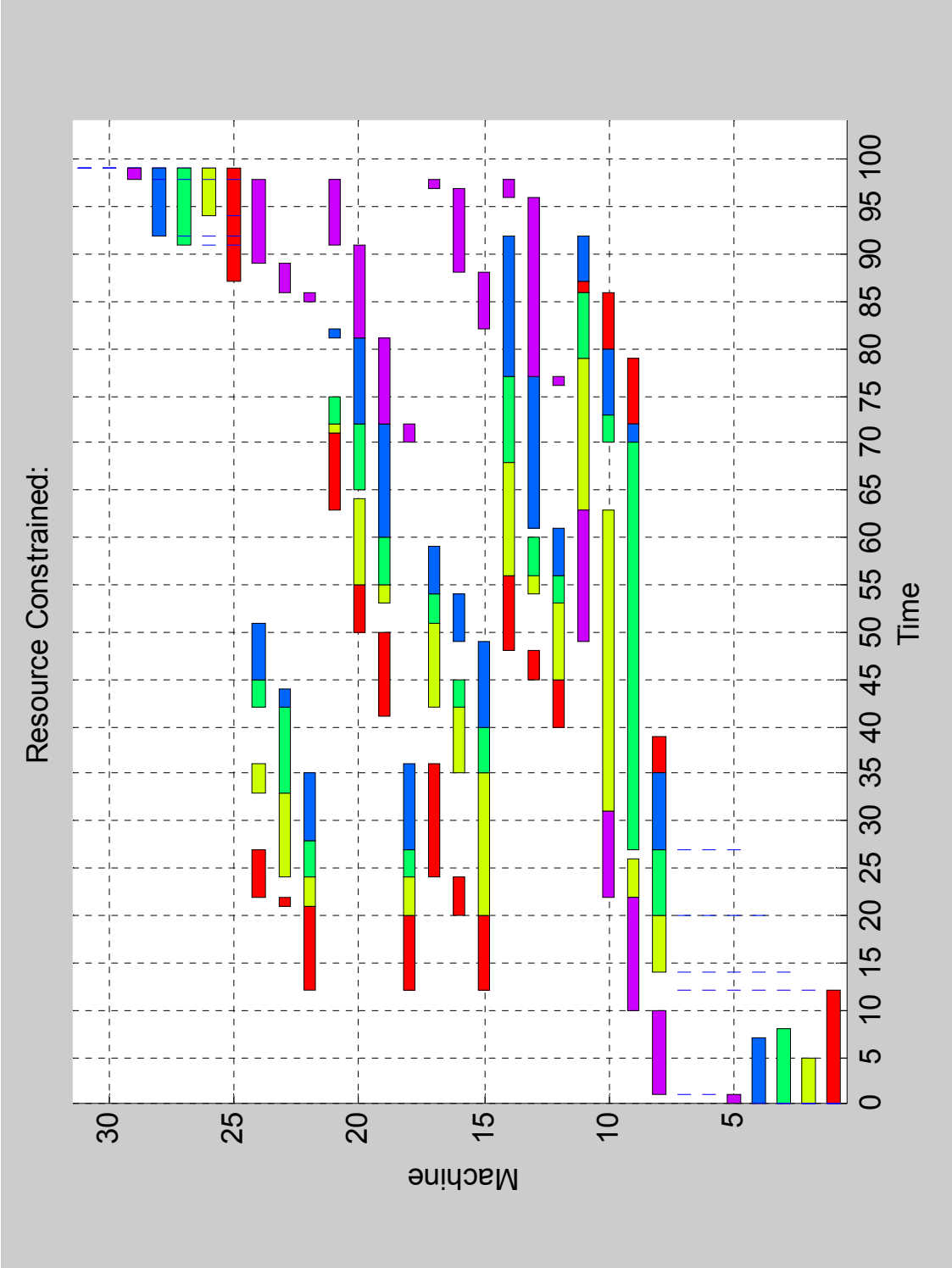

Figura 28 – 5 Jobs

Fonte: Autor

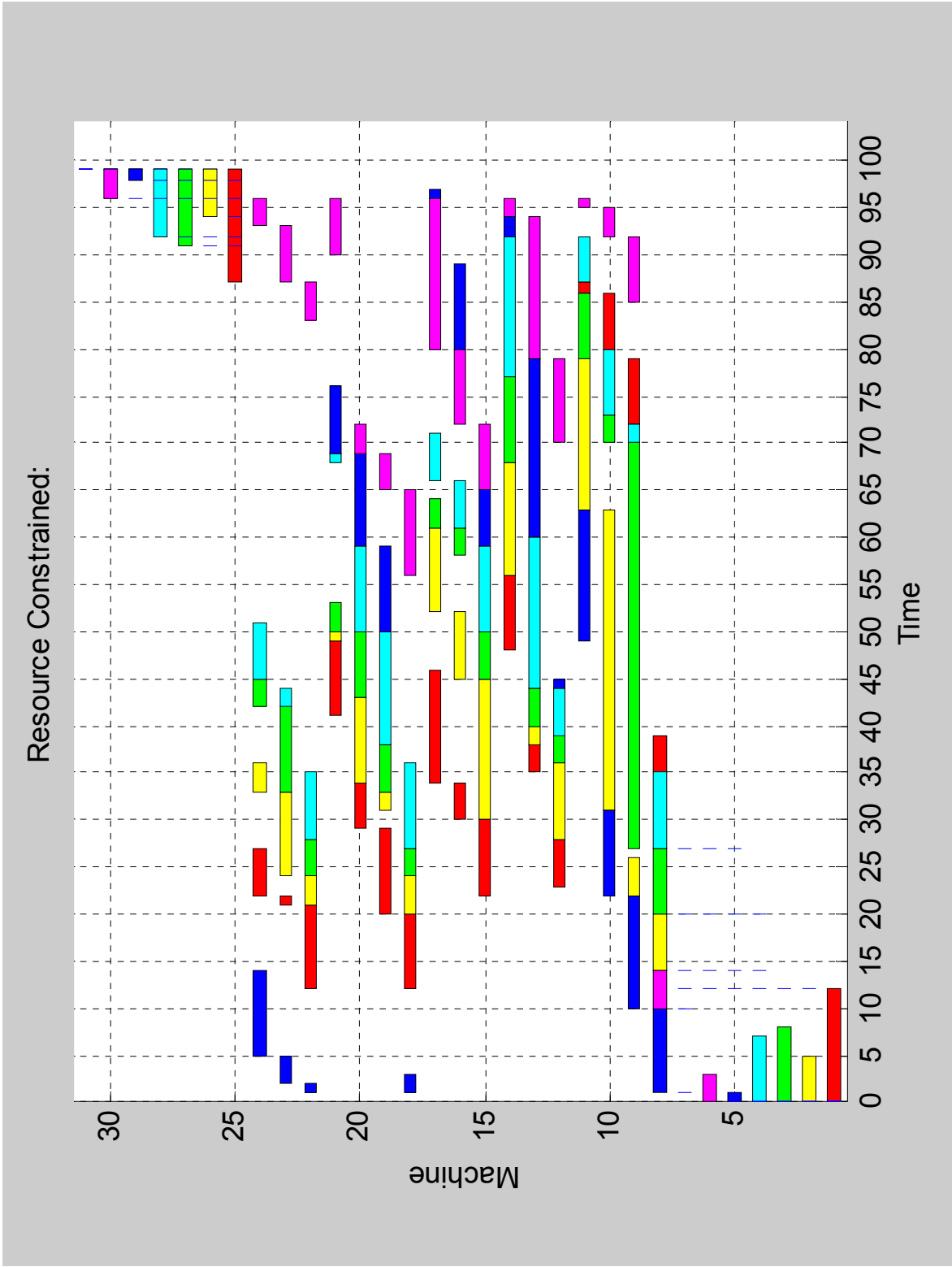

Figura 29 – 6 Jobs

Fonte: Autor

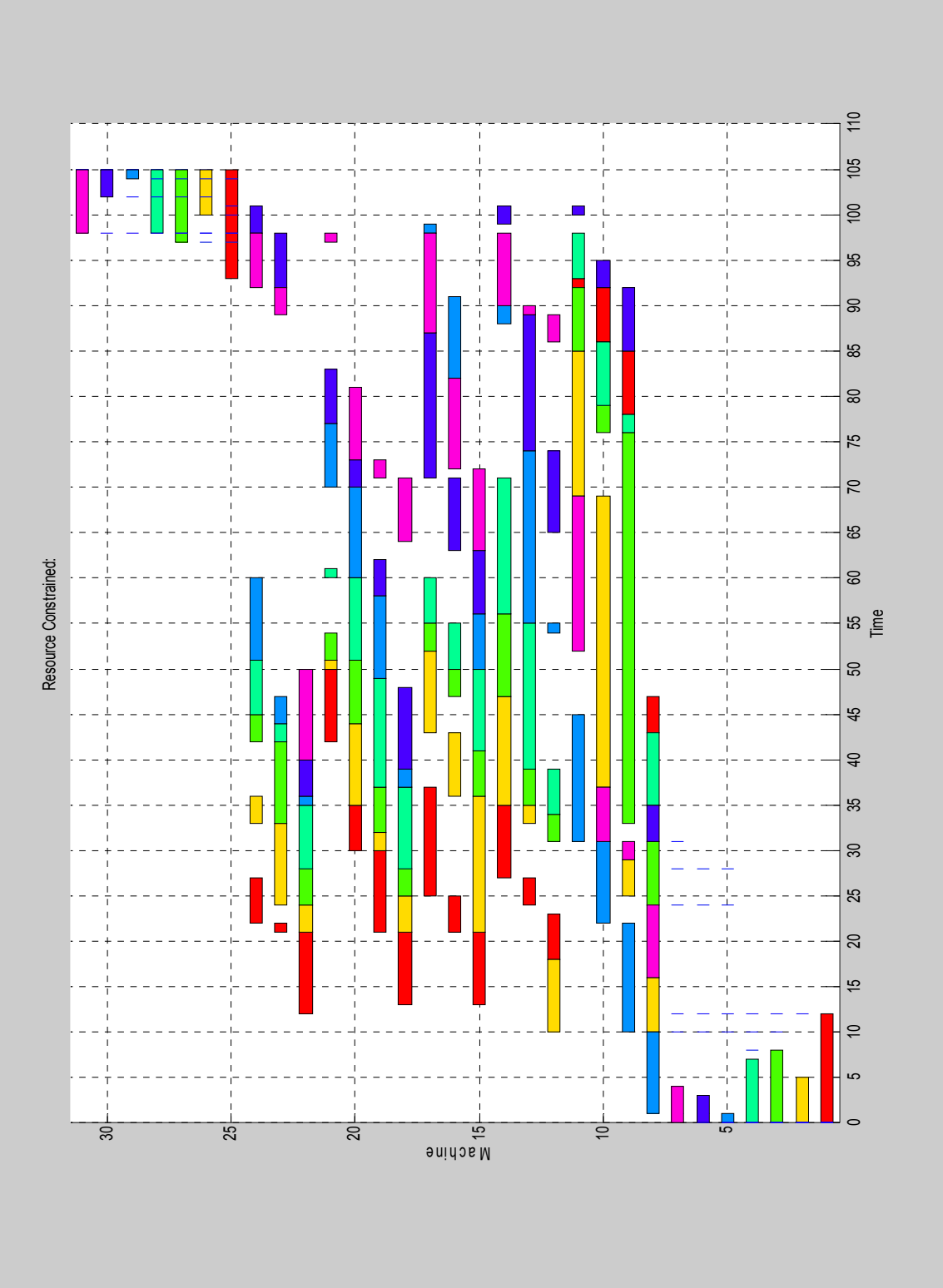

Figura 30 – 7 Jobs

Fonte: Autor

Capítulo IV
CONCLUSÃO

O presente trabalho apresentou uma metodologia de solução de problemas de Scheduling utilizando a técnica de Branch and Bound.

A limitação em 6 *Jobs* demonstrada no quadro comparativo depende dos tempos de processamento em cada atividade para cada *Job* portanto, o real limite poderá ser testado com a utilização de tempos compatíveis com o sistema a ser modelado e *pode* ser contornada se mudada a modelagem do problema, quando pensada em grupos de atividades, ao invés de simplesmente atividades isoladas, ou inserindo no problema critérios e restrições extras que simplifiquem a localização da solução ótima.

Considerando um problema de programação de tarefas em diversas máquinas e sujeitas a restrição de capacidade, com o objetivo de se minimizar o tempo de finalização de todas as tarefas, a abordagem utilizada permite a obtenção de resultados ótimos através da enumeração implícita do espaço de soluções, porém, como desvantagem principal, consome-se um elevado tempo de processamento.

O seu desempenho foi computacionalmente satisfatório quando se tratando de pequenos problemas, porém, ao se aumentar o número de Jobs e máquinas o tempo cresce exponencialmente chegando ao ponto de não poder ser resolvido e forçando a utilização de meta-heurísticas sob o custo de perder-se a garantia de solução ótima.

REFERÊNCIAS BIBLIOGRÁFICAS:

CORREA, H. L. & CORREA, C. A., 2004, *Administração da Produção e Operações*. São Paulo: Atlas

DAVIS L., SAMANLIOGLU F., STANFIELD P., DAVIS J. L.,2006, Department of Industrial Engineering and Systems Engineering, NCA&T, Systemized Critical Chain Project Management with Aplication to Remanufacturing. pp.1 - 6

DEMEULEMEESTER E., HERROELEN W., 1992, A Branch and Bound procedure for the Multiple Resource-Constrained Project Scheduling Problem. Management Science, Vol. 38, No. 12 (Dec., 1992),pp. 1803 – 1818

HILLIER, F.S; LIEBERMAN, G. J., 2002, *Introduction to Operations Research*. Seventh Edition. McGraw Hill.

MATLAB, Matlab for windows, versão 7.0: Matrix Laborathory. Michigan, MI, The MathWorks, Inc. Conjunto de programas 2 DVD-ROM

PINEDO, Michael., 1995, *Scheduling - Theory, Algorithms, and Systems.* 2 ed. Prentice-Hall.

SLACK, N. & OUTROS, 1996, *Administração da Produção*. Editora Atlas.

STIBOR, M., KUTIL, M.,2006, Torsche Scheduling Toolbox: List Scheduling. In *Proceedings of Process Control 2006* [CD-ROM]. Pardubice: University of Pardubice

TUBINO, Dalvio Ferrari, 2000, *Manual de planejamento e controle da produção*. 2. ed. – São Paulo: Atlas.