UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA QUÍMICA
ÁREA DE CONCENTRAÇÃO: SISTEMAS DE
PROCESSOS QUÍMICOS E IMFORMÁTICA

Estudo dos Conflitos de Armazenagem Intermediária em Flowshops

POR: Lázara Lane Oliveira
ORIENTADORA: Profª. Dra. Maria Teresa M. Rodrigues

Tese submetida à comissão de Pós-Graduação da Faculdade de Engenharia Química - UNICAMP como parte dos requesitos necessários para a obtenção do grau de Mestre em Engenharia Química.

Fevereiro/1994
Campinas-SP

FICHA CATALOGRAFICA ELABORADA PELA
BIBLIOTECA CENTRAL - UNICAMP

Oliveira, Lazara Lane
Estudo dos conflitos de armazenagem intermediaria em flowshops /
OL4e Lazara Lane Oliveira. -- Campinas, SP : [s.n.], 1994.

Orientador : Maria Teresa Moreira Rodrigues
Dissertacao (mestrado) - Universidade Estadual de Campinas.
Faculdade de Engenharia Quimica.

1. Administracao da producao. 2. Producao - Sequenciamento.
3. Armazenamento. I. Rodrigues, Maria Teresa Moreira.
II. Universidade Estadual de Campinas. Faculdade de Engenharia
Quimica. III. Titulo.

20. CDD - 658.5
-658.53
-658.785

Indices para catalogo sistematico:

1. Administracao da producao 658.5
2. Producao - Sequenciamento 658.53
3. Armazenamento 658.785

Esta versão corresponde a redação final da Tese de Mestrado defendida pela Engenheira Lázara Lane Oliveira e aprovada pela comissão julgadora em 28 /02/1994

Prof(a). Dr(a). Maria Teresa Moreira Rodrigues

Tese defendida e aprovada em 28 de fevereiro de 1994 pela banca examinadora constituida pelos professores:

Prof[a] Dr[a] Maria Teresa Moreira Rodrigues

Prof. Dr. João Alexandre Ferreira da Rocha Pereira

Prof. Dr. Sérgio Pérsio Ravagnani

Ao José Antônio,
minha mãe: Alina e
meu pai: Leônidas
(em memória).

A prof.ª Dra. Maria Teresa pela sua super orientação na elaboração deste trabalho e acima de tudo pela sua amizade que fez com que a realização deste se tornasse ainda mais interessante.

Ao José Antônio Duarte Reis pelo carinho, apoio, ajuda e pelo incentivo durante a realização desta tese.

A minha amiga Ana Bernadete Brasil que compartilhou comigo de toda esta fase de mestrado me ajudando sempre.

A minha mãe Alina e aos meus irmãos que sempre torceram por mim.

A todos os outros amigos pela ajuda, amizade e incentivo.

RESUMO

É cada vez maior o emprego de plantas bateladas em indústrias químicas de processamento de produtos de alto valor ou que seguem caminhos similares de produção. Nos tipos de processos em que o fluxo de processamento é unidirecional para todas as tarefas ou produtos a serem fabricados é chamado "flowshop". Neste tipo de processo a armazenagem intermediária é de vital importância para garantir a continuidade do mesmo. Esta tese desenvolve o problema de programação de produção de flowshops que envolve o sequenciamento e a alocação das tarefas. O sequenciamento é realizado utilizando uma técnica de inteligência artificial denominada Branch-and-Bound. É feito um estudo dos trabalhos existentes que tratam este assunto e dos conflitos existentes quando a armazenagem intermediária é limitada e a demanda é simultânea, envolvendo decisões de alocações.

Relação de Variáveis

Relação de variáveis

A(j,k)-Termo que representa o tempo em que a tarefa entra ou sai de uma unidade de armazenagem (+1:entra e -1:sai)

Ak(K)- Número de tanques de armazenagem disponíveis na unidade de processamento k

a_{ij} - Tempo de transferência da tarefa i do processador j

B - limitante utilizado na técnica BAB

BAB - Técnica "Branch and Bound"

C_{ij} - Instante de término da tarefa i no processador j

CP_{ij} - Instante em que a tarefa i está pronta para deixar o processador j

CS_{ij} - Instante em que o processador j está pronto para recewber a tarefa (i+1)

F - " Flowtime " ou tempo de residência da tarefa no processador

Relação de variáveis

FIS - Armazenagem intermediária limitada (Finite Intermediate Storage)

LB - Limitante inferior (LB) utilizado na técnica BAB

M - Número de processadores ou máquinas

M^* - Instante de término do processamento de todas as tarefas (makespan)

MIS - Armazenagem intermediária mista

N - Número de tarefas

NIS - Sem armazenagem intermediáaria (No Intermediate Storage)

r_{ij} - Tempo de residência da tarefa i no processador j

s_{ikj} - Tempo de estabelecimento ou tempo necessário para que o processador j possa receber a tarefa k após liberar a tarefa i

Relação de variáveis

Sint - Instante em que a tarefa entra na armazeangem

Sout - Instante em que a tarefa sai da armazeangem

t_{ij} - Tempo de processamento da tarefa i no processador j

UIS - Armazenagem intermediária ilimitada (Unlimited Intermediate Storage)

Z - Número de tanques de armazenagem

ZW - Nenhuma espera (zero wait ou no wait)

CAPÍTULO 1

Introdução

O processamento em plantas bateladas está se tornando cada vez mais importante dentro da Indústria de Processamento Químico, e é muito utilizado quando múltiplos produtos de alto valor comercial que seguem um caminho similar de produção devem ser fabricados na mesma planta. As indústrias de Química Fina e Farmaucêuticas são alguns exemplos que caracterizam este tipo de processo. Desde que o tempo e os recursos devem ser divididos entre os produtos, há a necessidade de sistematizar um problema de Programação de Produção das operações para se utilizar as instalações de uma maneira eficiente.

A fabricação de um determinado produto é feita através de uma sucessão de etapas, definida pela receita de processamento de cada produto. Naqueles casos em que cada etapa é realizada em um tipo de equipamento distinto, chamado "processador", e a sucessão de etapas, e portanto de equipamentos, for a mesma para a fabricação de todos os produtos, o fluxo de matéria na planta é unidirecional e esta estrutura de processamento unidirecional é chamada "flowshop".

Cada produto a ser fabricado constitui então uma tarefa a ser completada, e a execução de cada tarefa exige a execução de uma sequência de "operações", entendendo-se por operação a transformação físico-química que deve ocorrer em cada equipamento na rota de produção. Assim o problema de "Sequenciamento e Alocação de tarefas em flowshops" significa

definir a ordem em que os produtos (ou tarefas) devem ser produzidos, bem como os instantes de tempo em que cada operação deve ser executada em cada equipamento (ou processador). Como no caso da estrutura de processamento flowshop, todos os produtos obedecem a mesma rota de produção, dá-se especial atenção às sequências de permutação, que são aquelas em que as operações necessárias a execução de cada tarefa são realizadas na mesma sequência em todos os processadores.

O objetivo da Programação de produção é determinar a ordem e os instantes nas quais os produtos devem ser processados em cada um dos estágios ou equipamentos da planta otimizando algum critério de custo ou desempenho economicamente adequado. Este último pode ser a minimização do custo de processamento ou do tempo requerido para completar o processamento de todos os produtos por exemplo. A estrutura de processamento dominante na indústria química e tema de estudo desta tese é denominada flowshop, como já se viu.

Devido a natureza descontínua de processamento em plantas químicas bateladas, a armazenagem é um componente de importância fundamental. Geralmente a utilização do equipamento e a produtividade são baixos nestes processos de estado não-estácionário. Assim, a armazenagem intermediária pode ajudar reduzir tempos ociosos nestes estágios pela liberação destes para processar outras tarefas e então aumentar a utilização do equipamento e a produtividade de um processo batelada multiproduto.

Um problema de programação de produção envolve dois subproblemas:

. Determinar a sequência de processamento das tarefas que otimize o critério de desempenho do sistema escolhido;

. Determinar os instantes em que os eventos devam ocorrer durante o processamento, em especial os instantes em que as tarefas estejam prontas em cada processador, chamado "Completion Time".

Neste trabalho tem-se como objetivo implementar uma ferramenta para permitir a ordenação de um dado número de tarefas que se deseja executar em uma estrutura do tipo flowshop. A ferramenta tradicional escolhida é proveniente da área de inteligência artificial e é denominada técnica " Branch and Bound " (BAB). A seleção da melhor sequência em que as tarefas devem ser processadas na máquinas ou processadores é feita através de sucessivas ramificações e sondagens de cada nó que tem associado a ele uma estimativa do custo total constituída de duas parcelas : a primeira relativa ao custo associado a sequência parcial representada no nó e a segunda, que constitui uma estimativa de custo para completar a sequência total de produtos. Uma planta batelada real envolve um número de recursos disponíveis tais como: tempo, armazenagem intermediária, mão-de-obra, utilidades, etc, os quais são limitados. É desejável que a função de estimativa associada a cada nó sondado pela técnica BAB reflita adequadamente a

disponibilidade destes recursos compartilhados. Neste trabalho, será explorada a influência da armazeanagem intermediária na função de custo " tempo total de completamento das tarefas " denominado "makespan". Como se verá, a dificuldade em incorporar o custo relativo à armazenagem intermediária na função de estimativa do BAB se deve à natureza oportunista deste custo.

Neste trabalho procurou-se então explorar a problemática existente quando em um processo descontínuo flowshop, a armazenagem é reduzida pois em geral se espera que a redução na disponibilidade de tanques de armazenagem provoque um aumento substancial no valor do tempo total requerido para se produzir todas as tarefas.

Em suma, nesta tese tem-se dois objetivos:

- Apresentar o problema de programação da produção em unidades químicas multiproduto com limitação na armazenagem intermediária e as técnicas utilizadas na abordagem deste problema;
- Comparar os resultados obtidos no sequenciamento de tarefas, em termos do tempo total de execução das tarefas (makespan), usando a técnica Branch and Bound e, utilizando procedimentos heurísticos, evidenciando o caráter oportunista da armazenagem intermediária.

CAPÍTULO 2

Descrição e Métodos de Resolução de Flowshops

2.1. Introdução

Existem basicamente duas abordagens para resolver o problema de programação da produção em flowshops com mais de duas unidades de processamentos :

- Técnicas exatas, usando esquemas de Enumeração ou programação matemática, naqueles casos em que a formulação matemática do problema for possível;

- Técnicas Heurísticas baseadas no uso de regras heurísticas diretamente ou simplificações das técnicas exatas.

As técnicas exatas envolvem tipicamente soluções de problemas-NP (não descritos polinomialmente) através da programação dinâmica, programação inteira linear ou métodos " Branch and Bound " [Baker, 1974].

Técnicas Heurísticas são utilizadas em problemas nos quais respostas ótimas ou exatas não são requeridas ou no caso onde uma solução rápida é necessária. São estratégias que dão soluções próximas das "ótimas" para um grande número de problemas.

Neste trabalho foi utilizado a técnica "Branch and Bound" que será apresentado na seção 2.5.1.1 para sequenciar as tarefas em um flowshop. Esta técnica consiste em fazer uma " enumeração " inteligente, procurando usar conhecimentos e/ou estimativas que permitam reduzir as dimensões da árvore de busca

onde cada nó sondado representa uma sequência parcial. A avaliação de cada nó é feito com base no cálculo de uma função custo que serve como discriminação entre as diversas soluções parciais. Esta função de custo é composta de duas parcelas:

i) a 1.ª relativa ao custo da sequência parcial no nó e

ii)uma parcela referente ao custo mínimo que se incorpora para completar a solução.

A soma destas duas parcelas (custo mínimo do nó): i+ii é chamado limitante inferior.

A utilização de uma estimativa incapaz de incorporar as características do problema, ou seja, que não reflita de maneira adequada o custo das tarefas ainda não sequenciadas faz com que se tenha um maior número de caminhos igualmente promissores durante a construção da sequência. Assim o custo estimado é calculado de forma rápida mas pode conduzir a um número maior de nós sondados. Por outro lado, a utilização de uma estimativa capaz de incorporar todas as características do problema, refletindo de forma mais "real" o custo das tarefas ainda não sequenciadas traz o inconveniente de aumentar o tempo de cálculo da estimativa de custo aproximando-se de uma simulação das soluções candidatas . Então, uma estimativa "fina" reduz a árvore e em contrapartida leva a um aumento no tempo necessário para realizar a simulação. O ideal, então, é conseguir balancear de forma adequada estes dois pontos: a

utilização de uma estimativa que reflita de forma adequada as características do problema fazendo com que se tenha um número razoável de nós sondados mas que não seja tão minuciosa a ponto de se ter um tempo muito grande de cálculo da função custo. Quando a armazenagem intermediária é ilimitada o problema de minimização do tempo total de execução das tarefas pode ser feito através do recurso de diferentes maneiras de estimar este custo, tais como as chamadas heurísticas " Full Machine Based Bound "(FMBB), Single Machine Based Bound "(SMBB) entre outras [Baker,1974]. Quando a armazenagem intermediária é limitada, o aumento do "makespan" resulta do fato que a armazenagem é solicitada, mas não havendo armazenagem disponível, ocorre o bloqueio do equipamento ou processador. Neste caso podem ser adotadas duas estratégias:

i) bloquear o processador fazendo-o atuar como armazenagem e

ii) atrasar o início da operação até sincronizar os dois eventos: término da operação e liberação do processador.

Como se vê, a demanda de armazenagem é um evento que surge apenas durante a alocação, não existindo até o momento, uma estimativa de custo capaz de incorporar o impacto sobre o makespan resultante das estratégias citadas. A armazenagem intermediária tem então a característica de custo do tipo oportunista, isto é, pode ou não ocorrer a demanda deste

recurso durante o desenvolvimento da programação de produção. O impacto sobre o valor do makespan deve ser feito com a utilização de uma estimativa razoável. Para exemplificar tome-se o caso em que mais de uma tarefa necessita ser enviada para um mesmo tanque de armazenagem, supondo que apenas uma deve ser enviada. Qual será o impacto sobre o valor do makespan estimado no nó? Neste caso se percebe então como a função de custo se torna mais complexa, uma vez que todos estes problemas devem ser levados em conta na busca.

Neste trabalho, foi desenvolvido um estudo dos conflitos gerados pela limitação da armazenagem intermediária, como uma etapa fundamental para o desenvolvimento futuro de estratégias para estimar adequadamente o makespan. Neste capítulo serão apresentadas algumas definições básicas, comentários a respeito dos modos de resolução do problema de sequenciamento em flowshop, critérios de desempenho utilizados e por fim, as várias principais abordagens usadas para resolver o problema.

2.2. Características gerais de um processo Flowshop

Neste ítem, são apresentadas, em linhas gerais, as características deste tipo de processo bem como os critérios de

otimização mais frequentes.

Um flowshop é caracterizado por um fluxo de operações unidirecional. A estrutura geral é formada por M estágios consecutivos sendo cada tarefa formada por M operações. Cada uma das operações deve ser realizada em um só estágio obedecendo uma direção única de fluxo de operações. Em cada estágio pode haver um número variável de processadores. Para que cada uma das tarefas seja completada é necessário que seja executado um conjunto de operações e cada uma destas operações seja executada por um processador diferente. Um flowshop contém uma ordem natural de processadores. Os processadores são numerados de 1 a M e as operações da tarefa i são numeradas na forma (i,1), (i,2), ..., (i,M). Isto significa que a j-ésima operação de alguma tarefa precede sua k-ésima operação, então o processador alocado para desenvolver a operação j possui numeração inferior ao processador alocado para desenvolver a operação k (j<k). O caso mais simples é aquele em que cada estágio é formado por apenas um processador caracterizando um flowshop puro, que é o tipo utilizado neste trabalho. Neste caso estágio e processador se confundem sendo normalmente empregada a terminologia "processador". A FIG. 2.1 exemplifica um flowshop "puro", no qual todas as tarefas requerem uma operação em cada máquina.

As condições que caracterizam um flowshop são [Baker,1974]:

1. Cada tarefa requer M operações e cada operação requer uma diferente máquina;

2. Os tempos de estabelecimentos para as operações são independentes da sequência e são incluídos nos tempos de processamentos bem como os tempos de transferências entre bateladas e armazenagem;

3. Descrições das tarefas são conhecidas à priori;

4. M diferentes máquinas são continuamente disponíveis;

5. Não é permitida a preempção de operações ou seja uma vez iniciado a operação de uma determinada tarefa esta não pode ser interrompida até que seu processamento seja terminado.

Por causa da complexidade dos flowshops, estes podem ser resolvidos usando métodos enumerativos (exatos) ou métodos heurísticos que dão resultados aproximados.

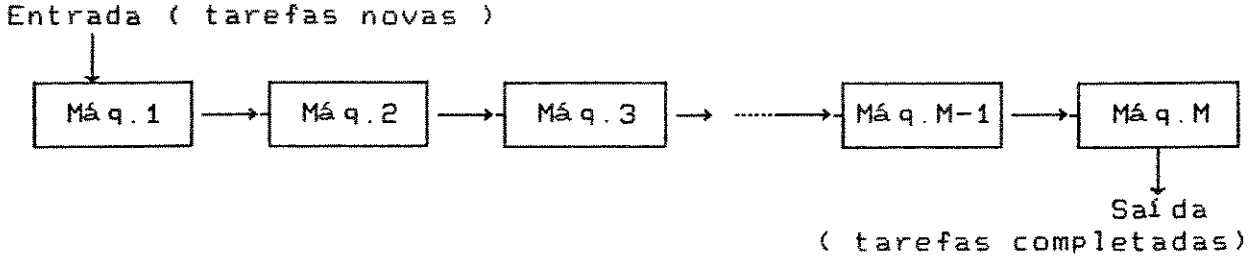


FIG.2.1 - Exemplo de um flowshop puro.

A programação de produção das operações envolve

duas tarefas interdependentes:

1. A determinação da melhor sequência na qual o grupo de tarefas deveriam ser produzidas;

2. Cálculo do instante de término das tarefas ("Completion Times") de todas as tarefas em todas as unidades.

2.3 Critérios de desempenho

O sequenciamento de tarefas deve ser realizado utilizando um critério de desempenho para caracterizar e discriminar entre diferentes soluções. Um dos critérios mais frequentemente empregados na determinação da sequência ótima é o do tempo total de processamento das tarefas, chamado "makespan" [Baker, 1974]. Outros critérios muitos frequentemente empregados na otimização são a minimização de :

i) tempo médio de residência ($\bar{F}$)

$$\bar{F} = (1/N) \sum_{i=1}^{N} Fi;$$

ii)atraso médio ($\bar{T}$)

$$\bar{T} = (1/N) \sum_{i=1}^{N} Ti;$$

iii)tempo de residência máximo (F_{max})

$$F_{max} = \max_{i \leq 1 \leq N} (Fi)$$

e outros.

Onde:

F_i = (Flowtime ou tempo de residência) quantidade de tempo em que uma tarefa permanece no sistema = $C_{iM} - r_i$;

r_i = Instante em que a tarefa i está pronta para ser executada.

Todas estas medidas de desempenho do sistema são chamadas de medidas regulares de desempenho e são sempre funções dos términos das tarefas, isto é [Rodrigues, M.T.M., 1992]:

$$Z = f (C_{1M}, C_{2M}, \ldots, C_{NM}).$$

2.4. Armazenagem intermediária

Durante anos, as operações contínuas tem prevalecido nas indústrias de processamento químico. Nos últimos anos, tem se tido um renovado interesse em processos bateladas para situações de processamentos envolvendo produtos complexos e de alto valor ou de múltiplos produtos pelo compartilhamento do uso dos equipamentos de processo. Plantas bateladas são mais flexíveis frente a pequenas mudanças nas

condições de operações do que as plantas contínuas.

Quando se trata de um processo batelada multiproduto, isto quando a rota de produção é a mesma para todos os produtos, a flexibilidade inerente contrasta com o problema adicional de estabelecer a capacidade real de produção em um intervalo de tempo. Dadas as dimensões do problema de sequenciamento e sua natureza combinatorial [Rodrigues,M.T.M.,1992], tem se dado ênfase na literatura ao problema de sequenciamento puro em flowshops, isto é, decidir apenas a sequência em que os produtos devem ser produzidos, sendo relegado a um segundo plano os problema de compartilhamento de armazenagem intermediária e outros recursos. Apesar de não existirem até o presente, estudos que comprovem esta afirmativa, tem-se creditado à utilização da armazenagem intermediária à redução de tempo ociosos nos processadores pela liberação deles para o processamento de outras bateladas e assim em projetos de plantas bateladas, geralmente oferece a melhor solução para aumentar a razão flexibilidade /custo. A armazenagem intermediária pode também ser usada para minimizar os efeitos de variações nos parâmetros do processo, para moderar os efeitos de falhas de equipamentos e para isolar intermediários associados com diferentes produtos entre outras coisas. E daí a sua ausência/presença pode afetar significativamente a programação de produção das operações.

2.4.1. Políticas de armazenagem intermediária

A capacidade da armazengem intermediária é medida em termos do número de unidades e não do tamanho físico da armazenagem, desde que é geralmente assumido que cada unidade pode manter temporariamente uma batelada de produtos.

As políticas de processamento das tarefas estão intimamente associadas à disponibilidade ou não de armazenagem intermediária, bem como a natureza dos produtos a serem produzidos e a seus intermediários presentes na planta.

As regras que governam a transferência de bateladas entre estágios de processamentos podem ser classificadas dentro de quatro políticas de armazenagem intermediária. Estas políticas são:

1)Unlimited Intermediate Storage (UIS);

2)Finite Intermediate Storage (FIS);

3)No Intermediate Storage (NIS) e

4)Zero Wait ou No Wait (ZW OU NW).

Na política UIS, os produtos intermediários são removidos das unidades tão logo tenham sido processados. É

disponível um número ilimitado de tanques de armazenagem intermediário para estocar cada produto parcialmente completo.

TAB 2.1 - Tempos de Processamentos

PROCESSADORES	TAREFAS			
	1	2	3	4
1	2	4	5	6
2	4	4	2	4
3	6	4	5	2

Aplicando os dados referentes a TAB. 2.1 tem-se a seguinte Carta de Gantt para o flowshop UIS representado na FIG.2.2, cuja sequência é 1-2-3-4.

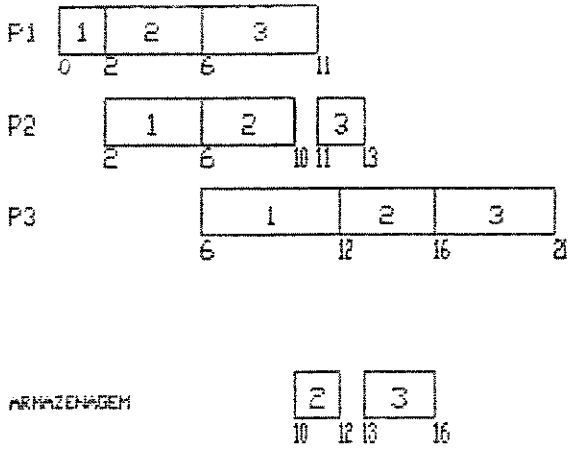


FIG. 2.2 - Modo UIS, dados referentes a TAB. 2.1

Na política NIS, os produtos intermediários podem ser removidos do processador j somente se o processador j+1 estiver disponível para processamento. Se o processador j+1

está comprometido, então o produto intermediário deve esperar no processador no qual ele acabou de sofrer o processamento. O modo NIS é utilizado quando não existe armazenagem intermediária disponível, os produtos intermediários são estáveis e não há restrição quanto à utilização do processador como tanque de armazenagem. A Carta de Gantt, referente ao exemplo anterior (mesma sequência), correspondente ao modo NIS é mostrado na FIG. 2.3.

Na política ZW há uma forte restrição: ou o produto intermediário é altamente instável de tal modo que deve ser completado sem interrrupção entre as operações.

Este modo é exemplificado na Carta de Gantt correspondente a FIG. 2.4.

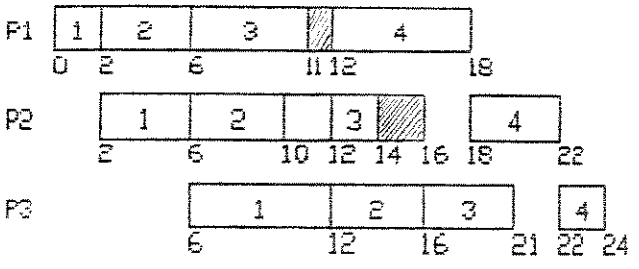


FIG. 2.3 - Modo NIS, dados referentes a TAB. 2.1

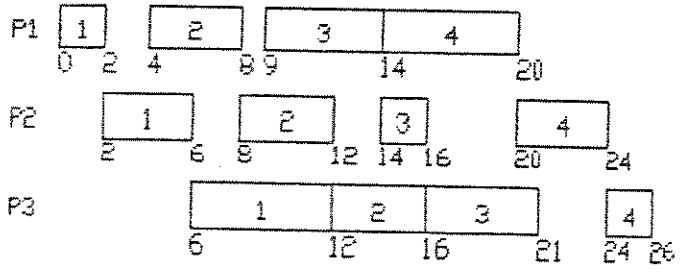


FIG. 2.4 - Modo ZW, dados referentes a TAB. 2.1

O modo FIS difere do UIS, apenas pelo fato que o número de tanques intermediários disponíveis é limitado.

2.5. Métodos de Resolução de Programação de Tarefas em Flowshops

A dificuldade de se resolver problemas de programação de produção de plantas bateladas reside na sua característica combinatorialmente explosiva [Rodrigues, M.T.M, 1992]. Os métodos exatos conhecidos para resolver o problema de sequenciamento são exponenciais no tempo, isto significa que o tempo computacional requerido para resolver este problema aumenta exponencialmente com o tamanho do problema. Uma notável exceção é o algoritmo de Johnson, 1954 [Reklaitis, 1985], o qual é um algoritmo polinomial no tempo para o problema UIS de duas unidades. O algoritmo de Johnson é um algoritmo polinomial pois o problema de sequenciamento de N tarefas em 2 processadores sob o critério do makespan não é NP. Em face de sua característica polinomial, o algoritmo de Jonhson tem servido como base para o desenvolvimento de heurísticas de sequenciamento em flowshops UIS com um número de processadores

superiores a dois [Knopf, 1982 ; Reklaitis, 1985], embora não se tenha tido bons resultados.

A maior parte dos problemas de sequenciamento em flowshops ocorre em sistemas com um número de processadores superiores a dois conduzindo a problemas da classe NP (não polinomiais) [Reklaitis, 1985]. A bem conhecida dificuldade computacional em resolver exatamente os problemas NP-completos (não polinomiais no tempo) tem levado ao desenvolvimento de soluções boas (subótimas) utilizando um esforço computacional razoável.

O problema de programação de produção de plantas bateladas que tem recebido um grande interesse em aplicações nos processos químicos é a minimização do "makespan" (tempo total de completamento das tarefas) admitindo apenas " sequências de permutação". Esta aproximação significa que somente são admitidas soluções onde cada unidade processa diferentes tarefas na mesma sequência. No caso de processos químicos, as operações de cada tarefa são normalmente executadas na mesma ordem em todos os equipamentos, o que equivale a dizer que uma vez iniciado produção de um dado produto, seu processamento não é suspenso de tal forma que a ordem de processamento é sempre a mesma. Isto se deve em parte à interconexão entre equipamentos e a natureza físico-química dos produtos e intermediários manipulados. Estas sequências são chamadas sequências de permutação. A vantagem de se considerar

somente "sequências de permutação" é que simplifica e reduz o espaço de busca da solução dos problemas flowshops para a determinação de apenas uma única sequência de produtos. Então, em um problema de N tarefas o espaço de busca se reduz a N! soluções ao invés de $(N!)^M$ caso não fosse considerado esta simplificação.

Para resolver este problema de programação, dois outros subproblemas interligados necessitam ser considerados : o sequenciamento das tarefas e a determinação dos tempos necessários para se completar o processamento destas em todos os processadores (Completion Times) e . A determinação do subproblema de cálculo dos "Completion times" se estingue com o cálculo detalhado da programação de produção de operação para uma dada sequência de tarefas. O subproblema de sequenciamento é solucionado com a determinação da sequência de produtos ótima que dá o mínimo tempo necessário para se completar o processamento de todas as tarefas (makespan).

2.5.1. Métodos de busca

A enumeração explícita de todos as soluções do problema é, para problemas grandes (N≥6) impraticável pois o espaço de soluções cresce combinatorialmente com a dimensão do problema (N!). Uma alternativa é então recorrer a métodos de busca controlada para minimizar uma função de custo escolhida.

Estes métodos funcionam através da aplicação de duas atividades: a)busca, que se concentra nas maneiras possíveis de exploração da estrutura em árvore e b) controle, que se concentra em selecionar dentre as várias alternativas de caminhos que podem ser percorridos na busca da solução ótima, aquela que efetivamente será escolhida.

Os métodos de busca controlada permitem que as atividades de busca e controle sejam realizadas utilizando diferentes tipos de cálculos, bem como a incorporação de informações acerca da provável evolução do critério de otimização empregado, auxiliando na escolha do caminho mais eficiente na busca da solução ótima e permitindo que sejam descartadas, ao menos temporiamente, soluções ineficientes.

A solução é construída pelo método, no sentido que a ordenação e alocação das tarefas são decididas consecutivamente em cada nó da árvore.

Os problemas de sequenciamento tem sido tradicionalmente resolvidos através de uma técnica da área de Inteligência Artificial denominada Branch and Bound (BAB). A utilização desta técnica na Programação de Produção, especialmente nos problemas de sequenciamento de tarefas consiste em decidir sucessivamente qual a próxima operação a ser executada e quando executá-la, com base em heurísticas para estimar o valor do critério de otimização. Devido a utilização praticamente unânime dos métodos tipo BAB para o sequenciamento

em flowshops, esta técnica é discutida com maiores detalhes na seção a seguir.

2.5.1.1. A Técnica Branch and Bound (BAB)

O método " Branch and Bound " é uma estratégia geral proposta para encurtar enumerações. a abordagem consiste de dois procedimentos fundamentais: "Branch " que é o processo de ramificar um problema grande em vários outros subproblemas e " Bound ", que é o processo de cálculo de um limitante inferior que reflete a estimativa de custo da solução completa.

O procedimento " Branch " ou de ramificação troca o problema original por :

1. um grupo de novos problemas que são mutualmente exclusivos e exaustivos oriundos do original ;

2.versões parciais resolvíveis do original e

3. problemas menores que o original.

Como um exemplo do procedimento " Branch ", tem-se P^{o} denotando um problema de sequenciamento de uma única máquina contendo N tarefas. O problema P^{o} pode ser dividido em N subproblemas P_1^1 , P_2^2, ... ,P_N^N, pela fixação da primeira posição na sequência. Então P_i^1 é o mesmo problema, mas com a tarefa 1 fixada na primeira posição e daí por diante. Claramente, estes problemas são menores que o P^{o} original porque somente (N - 1) posições permanecem para serem sequenciadas e obviamente que P_i^1

é uma versão resolvida de P^0. Adicionalmente, o grupo de subproblemas P_i^1 é uma ramificação mutuamente exclusiva e é um subproblema exaustivo de P^0 no sentido que, se cada P_i^1 é resolvido, o melhor das N soluções representará uma solução

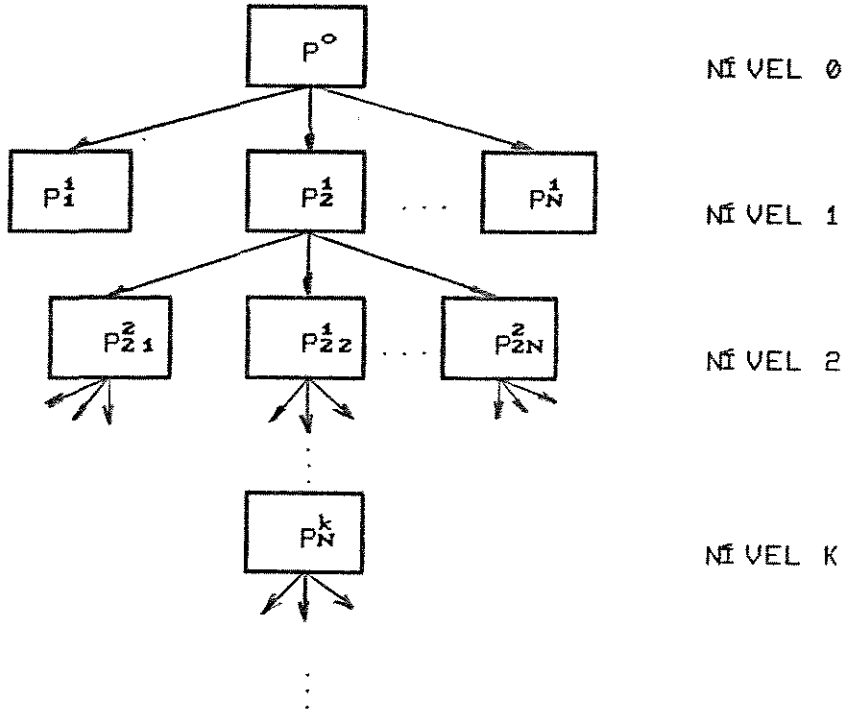


FIG.2.5 - A Técnica Branch.

ótima de P^0. Contudo o P_i^1, no primeiro nível satisfaz as condições (1), (2) e (3) acima. Assim, cada um dos problemas podem ser divididos em outras ramificações, conforme é verificado na FIG. 2.5. P_2^1 pode ser dividido em P_{21}^2, P_{23}^2, P_{24}^2, . . . , P_{2N}^2. Em P_{21}^2, as tarefas 2 e 1 ocupam as duas primeiras posições na sequência daquela ordem e em P_{23}^2, as tarefas 2 e 3 ocupam as duas primeiras posições, ou seja, a partição do

segundo nível, P^{2}_{2i}. Assim, em cada nível k, cada subproblema contém k posições fixadas e podem ser divididos em (N-k) subproblemas, os quais corresponderão ao nível (k+1). Se este procedimento " Branch ", ou de ramificação fosse executado totalmente, haveria N ! subproblemas no nível N, cada um correspondendo a uma solução distinta do mesmo problema original. Em outras palavras, a perseguição exaustiva da árvore " Branch " seria equivalente a uma enumeração completa de todas as sequências. A função do processo " bound" ou limitação, é de providenciar uma forma de encurtar ou reduzir esta enumeração.

A estrutura do tipo árvore conforme a FIG.2.5 é composta de nós que representam soluções parciais do problema e ramos que ligam nós de níveis sucessivos. Durante o processo de busca se terá na árvore diversos caminhos incompletos e um custo total associado a cada nó. O caminho ótimo é encontrado através da ramificação do nó com menor custo, gerando nós em um nível de profundidade da árvore imediatamente posterior. Este procedimento é aplicado ciclicamente até que a solução final seja obtida.

A utilização eficiente de um método deste tipo apresenta uma grande limitação: o cálculo da função custo asssociado a cada nó. Esta função custo é normalmente composta das duas parcelas já citadas anteriormente:

i)a primeira relativa ao custo associado à solução parcial representada no nó e

ii)a segunda relativa a uma estimativa de custo mínimo remanescente necessário para completar a solução parcial do nó.

A primeira parcela é fácil de ser calculada a partir dos dados do problema. A segunda parcela, que é uma estimativa de natureza oportunista é mais complexa e implica na utilização de heurísticas para o cálculo de um custo mínimo associado ao resto do caminho em direção à solução final. Esta segunda parcela é um custo futuro e deve ser calculada com base em uma função de estimação que represente de forma fiel o modelo do problema pois, desta forma, o custo estimado se aproxima bastante do custo real. É claro que deve haver bom senso no sentido que a estimativa empregada não se aproxime demasiadamente de uma "simulação" do caminho a ser ainda percorrido a partir do nó, pois assim pode se ter um aumento do custo computacional (tempo) necessário para se calcular esta parcela, aproximando esta busca a uma enumeração completa.

Ao custo mínimo calculado com base na soma das duas parcelas citadas dá-se o nome de limitante inferior ("Lower Bound") no qual é baseado, por sua vez o controle da enumeração. Os métodos do tipo BAB apresentam uma série de características adequadas para a solução de problemas de programação de produção em flowshops.

2.5.1.2. Funções de Estimação de Limitantes

A escolha dentre as tarefas não-sequenciadas da próxima tarefa a integrar a sequência parcial é normalmente feita com base em estimativas do custo ou desempenho do sistema calculados com base nas duas parcelas já citadas (seção 2.5.1.1): uma parcela referente ao custo ou desempenho do sistema associado ao conjunto de tarefas sequenciadas e a outra parcela referente à estimativa do custo ou desempenho do sistema associado às tarefas não-sequenciadas. Quanto melhor for a estimativa do comportamento futuro da sequência, em outras palavras, quanto mais a função de estimação empregada se aproximar das hipóteses do modelo do problema em estudo, maior será a probabilidade de se escolher rapidamente o melhor caminho na busca de uma solução.

A utilização eficiente de métodos de busca na solução de problemas exige o cálculo de um limitante inferior com base em funções ou procedimentos de estimativas de custo adequadas e que reflita o problema estudado. Uma boa estimação permite rejeitar muitos dos caminhos prováveis de soluções. Além disso a eficiência é também resultado de uma decisão entre o compromisso de uma estimativa rigorosa do custo de cada caminho (ou solução parcial) e o número de caminhos sondados. Portanto a parcela de estimativa deve incorporar de forma realista o custo associado as tarefas ainda não-sequenciadas, pois senão se

terá um maior número de caminhos promissores na construção da sequência das tarefas.

A primeira parcela da função custo representa o tempo total necessário para se completar o processamento de todas as tarefas (makespan) da sequência parcial, calculado com base no instante de término das tarefas $C_{i,j}$, que por sua vez reflete a distribuição temporal das tarefas já sequenciadas e, portanto representa o tempo efetivamente comprometido na execução de uma determinada sequência parcial. No caso mais simples em que o tempo de processamento já inclui os tempos de transferência e preparação do equipamento, este tempo pode ser calculado pela expressão (2.1):

$$C(i,j) = \max [\, C(i-1,j) ; C(i,j-1) \,] + t_{ij} \qquad (2.1)$$

Onde:

$C_{i,j}$ = instante em que o tempo em que a tarefa com posição na sequência é terminada no processador j.

t_{ij} =tempo de processamento da tarefa i no processador j.

A segunda parcela reflete o custo mínimo estimado do caminho a ser ainda percorrido. No caso da armazenagem intermediária ser limitada, a não disponibilidade de tanques pode provocar um aumento do limitante inferior, assim, a estimação desta parcela tanto será melhor quanto mais cedo for possível quantificar o impacto sobre o limitante inferior desta limitação de tanques de armazenagem. No entanto, esta parcela

não depende exclusivamente de dados do problema, mas do desenvolvimento da programação de produção. Neste sentido, o custo decorrente da limitação de tanques é chamado "custo oportunista", pois sua previsão antecipada não é, até o momento possível. Uma das funções estimadoras mais usadas até o momento, quando se adota o critério do makespan, tem sido a heurística FULL MACHINE BASED BOUND. Nesta função estimadora calcula-se o impacto mínimo sobre o valor do makespan estimado em cada nó, admitindo-se que os processadores nunca estarão ociosos e que a armazenagem disponível é infinita. A expressão para o cálculo dos limitantes inferiores (LB) para o flowshop UIS [Baker, 1977] pela heurística Full Machine Based Bound é dado por:

$$Bj = C(k,j) + \sum t_{i,j} + \min_{i \in U} \left\{ \sum_{l=j+1}^{M} t_{il} \right\} \quad 1 \leq j \leq M-1 \quad (2.2)$$

$$B_M = C(k,M) + \sum_{i \in U} t_{iM}$$

$$LB = \max_{i \leq j \leq M} \{ Bj \}$$

Onde:

LB = limitante inferior ("Lower Bound")

Neste trabalho a função de estimação foi utilizada para o cálculo do tempo total de término das tarefas (makespan), o que implica em admitir que, para o conjunto de tarefas não sequenciadas a armazenagem intermediária disponível é limitada.

Como esta hipótese não é verdadeira, pode ocorrer que em um ou vários instantes futuros, a demanda de armazenagem pode ultrapassar a armazenagem disponível. Neste caso, deve ser tomada uma decisão de alocação.

A árvore de busca é formada por nós e ramos. A cada nó corresponde uma solução parcial do problema, e cada solução parcial tem associada um custo. Este custo tem como objetivo auxiliar na decisão de qual nó será "explodido", gerando ramificações que conduzem a novas soluções parciais. No exemplo a seguir é mostrado a aplicação de um método do tipo "Branch and Bound" com busca pelo melhor (BAB Best First) ao sequenciamento de tarefas em um flowshop UIS.

TAB. 2.2 - Tempos de Processamentos

TAREFA	PROCESSADOR		
	1	2	3
1	7	4	15
2	1	8	7
3	13	5	4
4	10	11	5

Na TAB. 2.2 são apresentados os tempos de processamentos das tarefas em todas as máquinas e a FIG.2.6 dá a árvore de busca gerada usando a heurística "Full Machine Based

Bound" (FMBB).

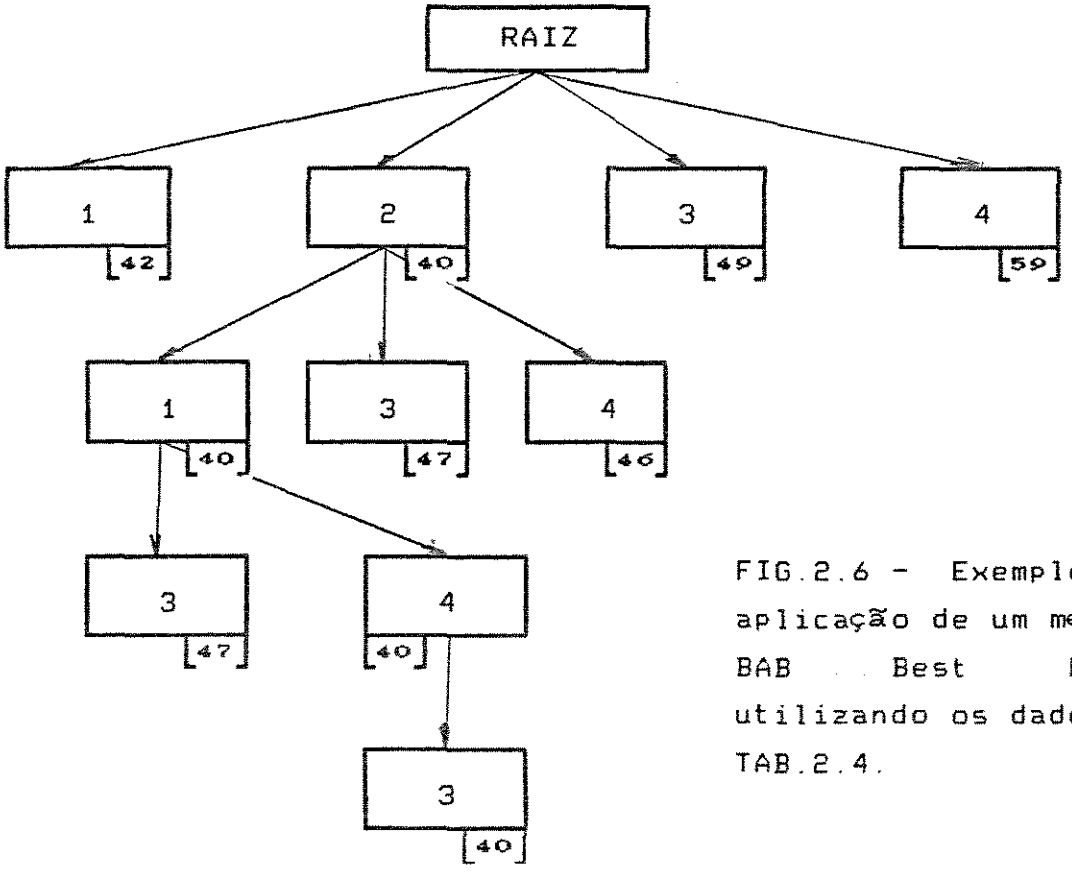


FIG.2.6 - Exemplo de aplicação de um método BAB Best First utilizando os dados da TAB.2.4.

2.5.2. Algoritmo de Johnson

Para o caso particular de sequenciamento de N

tarefas em um flowshop de dois processadores com armazenagem intermediária infinita entre os estágios (UIS) adotando-se o critério do makespan, é utilizado o algoritmo de Johnson.

Este é um algoritmo connstrutivo cujas etapas são mostradas a seguir:

Etapa 1 - Encontre o $\min_i \{ t_{i1}, t_{i2} \}$,

onde:

i - tarefa a ser processada

t_{i1} - tempo de processamento da tarefa i no processador 1

t_{i2} - tempo de processamento da tarefa i no processador 2

Etapa 2a - Se o tempo de processamento mínimo ocorre no processador 1 esta deve ser sequenciada o mais cedo possível e retirada da lista de tarefas não sequenciadas.

Etapa 2b - Se o tempo de processamento mínimo ocorre no processador 2 a tarefa em questão deve ser sequenciada o mais tarde possível e retirada da lista de tarefas não sequenciadas.

Para ilustrar o algoritmo considere-se o problema de cinco tarefas mostrado na TAB. 2.3. O trabalho de determinação da sequência ótima em cinco estágios usando o algoritmo de Johnson é mostrado na TAB. 2.4. Em cada estágio o tempo de processamento mínimo entre as tarefas ainda não

sequenciadas devem ser identificadas. A sequência encontrada é 3-1-4-5-2.

TAB. 2.3 - Tempos de Processamentos

	TAREFAS				
PROCESSADORES	1	2	3	4	5
1	3	5	1	6	7
2	6	2	2	6	5

TAB. 2.4 - Sequenciamento gerado pelo algoritmo de Johnson.

ESTÁGIO	TAREFAS NÃO-SEQUENCIADAS	t_{ik} MÍNIMO	POSIÇÃO	SEQUÊNCIA PARCIAL
1	1,2,3,4,5	t_{31}	3 = [1]	3 x x x x
2	1,2,4,5	t_{32}	2 = [5]	3 x x x 2
3	1,4,5	t_{11}	1 = [2]	3-1 x x 2
4	4,5	t_{52}	5 = [4]	3-1-x 5-2
5	4	$t_{41} = t_{42}$	4 = [3]	3-1-4-5-2

2.5.3. Regras Heurísticas para o Sequenciamento das Tarefas

A abordagem Branch and Bound discutida anteriormente e desenvolvidada neste trabalho oferece duas inevitáveis desvantagens, as quais são típicas de métodos que tem por objetivo a obtenção de soluções exatas . Primeiro, a necessidade de alto esforço computacional para problemas de grande dimensão. Segundo, mesmo para problemas de pequena dimensão não é garantido que a solução possa ser obtida

rapidamente, desde que a extensão da enumeração parcial (número de nós sondados) depende dos dados do problema. Algoritmos heurísticos evitam estes dois inconvenientes: eles podem obter soluções para problemas grandes com esforço computacional limitado, e suas necessidades computacionais são praticáveis para problemas de um dado tamanho. A desvantagem das aproximações heurísticas é, de fato, que elas não garantem otimalidade e, é difícil julgar a qualidade da asolução obtida.

Adiante são descritas, de forma resumida, duas heurísticas representativas de rapidez que foram utilizadas neste trabalho como forma de comparação com os resultados obtidos com a técnica BAB.

2.5.3.1. Heurística de Dannembring (RA - Rapid Acess)

Esta heurística transforma um problema flowshop de de M estágios em um subproblema de dois estágios através das seguintes equações:

$$t_{i1} = \sum_{j=1}^{M} (M - j + 1) \, t_{i,j} \qquad (2.3)$$

$$t_{i2} = \sum_{j=1}^{M} j \, t_{i,j} \qquad (2.4)$$

É utilizado o Algoritmo de Johnson para resolver

estes dois novos subproblemas (M=2).

2.5.3.2. Heurística de Campbell-Dudek-Smith (CDS)

Esta heurística é semelhante à anterior. Ela gera (M-1) problemas artificiais de dois estágios utilizando as seguintes expressões para o cálculo dos tempos de processamentos em cada estágio:

$$t_{i1}^{k} = \sum_{j=1}^{k} t_{i,j} \qquad (2.5)$$

$$t_{i2}^{k} = \sum_{j=M-k+1}^{M} t_{i,j} \qquad (2.6)$$

Para ilustrar a aplicação da Heurística de CDS considere os dados apresentados na TAB. 2.5. Neste caso fazendo i variar de 1 até 4 tem-se através das Eqs. (2.5) e

TAB. 2.5 - Tempos de Processamentos

	TAREFAS			
PROCESADORES	1	2	3	4
1	5	4	2	6
2	3	6	3	5
3	8	2	6	4
4	3	1	2	6

(2.6) a transformação de um problema de 4 tarefas e 4 processadores em 3 novos subproblemas de 4 tarefas e 2 processadores conforme mostrado na TAB. 2.6. Assim estes novos subproblemas são resolvidos utilizando o Algoritmo clássico de Johnson e sendo escolhida a sequência de menor valor do makespan.

TAB. 2.6 - Exemplo de aplicacação da Heurística de CDS.

k = 1	k = 2	k = 3
$t^1_{11} = t_{11} = 5$	$t^2_{11} = t_{11} + t_{12} = 8$	$t^3_{11} = t_{11} + t_{12} + t_{13} = 16$
$t^1_{12} = t_{14} = 3$	$t^2_{12} = t_{23} + t_{24} = 11$	$t^3_{12} = t_{12} + t_{13} + t_{14} = 14$
$t^1_{21} = 4$	$t^2_{21} = 10$	$t^3_{21} = 12$
$t^1_{22} = 1$	$t^2_{22} = 3$	$t^3_{22} = 9$
$t^1_{31} = 2$	$t^2_{31} = 5$	$t^3_{31} = 11$
$t^1_{32} = 2$	$t^2_{32} = 8$	$t^3_{32} = 11$
$t^1_{41} = 6$	$t^2_{41} = 11$	$t^3_{41} = 15$
$t^1_{42} = 6$	$t^2_{42} = 10$	$t^3_{42} = 15$

2.6 — Abordagens para a resolução das políticas UIS, NIS, ZW e FIS mostradas na literatura

Para o caso do flowshop com dois estágios sem limitação na oferta de tanques de armazenagem intermediária (chamado tecnicamente de N/2/F, ou seja sequenciamento de N tarefas em 2 estágios, otimizando o tempo total de processamento das tarefas) foi proposto o algoritmo de Johnson (1954). O algoritmo de Johnson é um algoritmo "construtivo", isto é, um algoritmo em que as soluções do problema (definir a sequência de tarefas) é construída a partir de determinadas Regras, as quais foram mostrada em formas de etapas na seção 2.5.2.

A vantagem deste algoritmo é que ele é polinomial com o crescimento de N, evitando a explosão combinatória. A ordem na qual os produtos são processados na segunda unidade é a mesma que na primeira (sequências de permutação).

Para o caso N/M/F com M>2 não existem soluções construtivas e ocorre a explosão combinatória. Para contornar este problema recorre-se frequentemente a regras heurísticas para problemas de sequenciamento de grande porte (N≥12; M≥3) abdicando, nestes casos, da otimalidade da função custo.

Ainda para o modo UIS existem resultados satisfatórios de uma variedade de problemas usando técnicas Branch and Bound [Baker, 1974;]. Quando uma solução exata não

é necessária, mais de 100 regras de sequenciamento para problemas flowshops tem sido avaliada por [Panwalker e Islander, 1977].

Para o caso NIS, [Suhami e Mah, 1981] apresentaram um "Branch and Bound " para a solução deste problema

O sistema de armazenagem ZW foi estudado por [Wismer,1972], [Gupta,1976], [Panwalker e Woolman, 1979], pesquisados por [Gonzales, 1976] e [Graham et al, 1979]. Nestas referências mostra-se que o problema NW pode ser transformado em um "problema de caixeiro viajante", um problema não polinomial de otimização. Este problema também foi abordado pela técnica BAB [F. Carl Knopf, 1984].

No FIS existem relatados apenas dois trabalhos :

1. [Dutta e Cunningham, 1975]; que envolve uma formulação de programação dinâmica de um problema flowshop makespan de dois estágios. Neste estudo os tempos de transferência foram considerados negligenciáveis sendo desenvolvido um algoritmo exato usando programação dinâmica. Devido ao elevado tempo computacional gasto para resolver o problema foram desenvolvidosdois métodos aproximados e apresentados por estes autores.

2. [Papadimitrius e Kanellakis,1980]; examinaram o problema de sequenciamento em flowshops de dois processadores sob o critério do makespan. Neste artigo os autores examinaram

a estrutura-NP do FIS e NW e propondo uma heurística para resolver o flowshop FIS com dois processadores. As hipóteses usadas neste trabalho foram as mesmas de [Dutta e Cunningham,1975]. A heurística consiste em programar a produção primeiramente do processo sem usar armazenagem (NIS) e então usar armazenagem para reduzir o tempo ocioso. Os resultados mostraram para este trabaho que este procedimento esteve sempre dentro de 50% do ótimo.

Todos os trabalhos descritos até então não podem ser aplicáveis para situações reais onde os tempos de transferência entre unidades não podem ser desprezados.

Um flowshop de dois estágios generalizado com tempos de transferência entre unidades, não-negligenciáveis e com armazenagem intermediária finita (FIS) foi desenvolvido por [F. Carl Knopf, 1984] para o flowshop com dois processadores. O modo de operação dos processadores é semi-contínuo e os tanques podem ser compartilhados entre os produtos sem necessidade de preparação e limpeza dos tanques.

Neste trabalho, foram discutidas as complexidades de um problema flowshop FIS, onde uma solução exata requer um esquema de enumeração. Knopf sugere um método de resolução em que se utiliza a técnica BAB com auxílio de simulaçõespara avaliação dos nós sondados. A forma de resolver este problema FIS segue as seguintes etapas:

- estima-se um limitante inferior para uma dada sequência representada por um nó avaliado pela técnica BAB. É estimado um limitante superior inicial pela utilização do algoritmo de Johnson modificado e

- se este limitante inferior for maior que o superior avalia-se uma nova sequência representada por um novo nó sondado pela técnica BAB. se este último for menor, torna-se o novo limitante superior e a busca continua. Este limitante inferior é calculado como o tempo necessário para se completar o processamento de todas as tarefas.

Este trabalho de [F. Carl Knopf, 1984] trouxe duas contribuições principais para a programação de problemas FIS:

1. O conceito de acoplagem de simulação para melhor abordar o caráter oportunista deste tipo de política FIS com "Branch and Bound" em ordenação para geração de Completion Times mínimos;

2. Regra Heurística para capacitar um boa programação a ser determinada pelos problemas FIS. A modificação do Algoritmo de Johnson foi mostrado ser bastante eficiente para os problemas estudados.

O mesmo sistema FIS de duas unidades com armazenagem intermediária finita foi estudado por [Wiede e Reklaitis, 1987]. Foi desenvolvido um algoritmo para calcular os completion times, sob certas hipóteses para os sistema FIS,

processando N tarefas:

1. todas as tarefas são produzidas na mesma ordem em cada unidade com não-preempção de produto;

2. cada unidade não pode processar simultaneamente mais do que uma tarefa no tempo ;

3. cada tarefa programada não pode ser processada mais do que uma vez em cada unidade de processamento.

4. há Z tanques de armazenagem disponíveis com uma capacidade finita entre as unidades de processamentos;

5. o tempo de transferência de e para a armazenagem é negligenciável e

6. cada tanque de armazenagem pode armazenar somente um produto de cada vez.

Primeiramente foi estudado o caso N/2/M com Z tanques disponíveis entre os estágios (Parte I), posteriormente o estudo foi ampliado para um número de processadores superior a 2 (Parte II e III). A estratégia empregada pelos autores consistiu em sequenciar N tarefas considerando a política UIS. Em seguida a solução do problema era compatibilizada para ter em conta a limitação na oferta dos tanques. Esta compatibilização foi feita através do atraso adequado do início da operação de uma dada tarefa (ou tarefas) na hipótese de haver um conflito de demanda do recurso, isto é, quando a demanda, em um determinado instante superar a oferta de recurso. Por último, o valor do makespan é avaliado.

Os algoritmos desenvolvidos nestes três trabalhos sucessivos são descritos a seguir:

Parte I :

Etapa 1) A sequência é obtida utilizando um algoritmo de mínimo makespan sob o modo ZW ou utilizando a Regra de Johnson (mínimo makespan sob o modo UIS);

Etapa 2) Toma j = 0 (j é a tarefa sequenciada);

Etapa 3) Faz j = j+1 e se j>N, para. Se não, vá para a etapa 4;

Etapa 4) Obtém-se Completion Times para a tarefa j nso processadores 1 e 2 operando sob o modo UIS;

Etapa 5) Checa-se os Completion Times da tarefa sequenciada com relação à disponibilidade de tanques de armazenagem Z, através da sequência de processamento. Se a tarefa não requer armazenagem, vai para a etapa 3, se não, esta posição não é adequada e procede-se para a etapa 6;

Etapa 6) Obtém-se uma programação UIS modificada para a tarefa j na unidade 1 buscando não exceder o número de tanques de armazeangem, Z, então procede-se para a etapa 3.Isto é feito realizando o atraso do início desta referida tarefa para se desfazer a necessidade de armazenagem.

Parte II:

Etapa 1) Como na Parte I, é encontrada a sequência para o modo UIS ou ZW;

Etapa 2) Toma-se j = 0;

Etapa 3) Faz-se j = j + 1. Se j) N, para. Se não procede para a etapa seguinte;

Etapa 4) Para a sequência UIS assumida inicialmente é calculada os Completion Times para j em todos os processadores k (k = 1,...,M);

Teste de disponibilidade de armazenagem. Checa se há armazenagem disponível para a demanda, se houver: continua na etapa 3 e se não, vai para a próxima etapa;

Etapa 6) Encontra-se um modo UIS modificado atrasando-se o início do processamento desta referida tarefa.

O procedimento descrito na Parte II foi modificado para acomodar a configuração MIS geral (blocos de modos UIS, ZW, NIS) da Parte III.

De acordo com os resultados obtidos, este último algoritmo pode satisfatoriamente servir como uma ferramenta prática de programação, com bons resultados dentro da provável precisão dos dados de tempos de processamentos. É importante salientar que nestes três últimos trabalhos descritos (Partes I, II e III), os resultados foram apresentados somente em forma de comparações entre os tempos de computação.

[Ku e Karimi, 1988] também propuseram uma abordagem de programação de N tarefas através de um sistema de processamento de M-estágios em série, com uma única unidade por estágio. O sistema em questão foi operado sob uma combinação das políticas UIS, FIS e NIS. As principais hipóteses

admitidas pelos autores foram:

Foi assumido neste trabalho :

1. todos os produtos são produzidos na mesma ordem em cada unidade de processamento (sequência de permutação);

2. operação não preemptiva, ou seja, uma vez iniciado o seu processamento, este não pode ser iterrompido e iniciado mais tarde;

3. uma unidade não pode processar mais do que um produto no tempo, nem um produto pode ser processado por mais de uma unidade simultaneamente;

4. a última unidade de processamento opera sob o modo UIS e

5. os tempos para transferência de bateladas entre unidades e armazenagem são negligenciáveis.

Similarmente, os tempos de estabelecimentos e limpezas requeridos de unidades e armazenagem quando mudam de um produto para outro foram também considerados negligenciáveis.

Este autores estudaram o caso da armazenagem intermediária inter-estágios e desenvolveram uma modelagem matemática como um problema MILP (Mixed Linear Programming - Hong-Ming Ku e Karimi, 1988). Os autores afirmam que a estratégia MILP é preferível à estratégia BAB, já que existem pacotes codificados para resolver problemas MILP, enquanto que a estratégia BAB deve ser codificada. No entanto, deve-se ressaltar dois pontos: todos os pacotes MILP recorrem à

estratégia BAB para satisfazer as restrições de integralidade que representam as decisões de alocação; a estratégia BAB permite uma maior interação com o usuário na modificação das restrições do problema incorporando ou desprezando informações. Neste último caso a estratégia MILP exige uma nova modelagem matemática do problema. No entanto eles reconhecem que a MILP consome um tempo de computação muito alto. Tentando contornar este problema Ku e Karimi tentaram simplicar esta formulação relaxando as suas restrições de integralidade degenerando assim a solução com relação a solução obtida original.

Para o caso da armazenagem compartilhada entre estágios a formulação matemática do problema se torna mais difícil uma vez que a restrição de factibildade de utilização da armazenagem disponível depende de uma análise global da demanda de armazenagem e não apenas de uma análise de demanda entre dois estágios consecutivos. Por esta razão não existe até o presente a formulação de uma abordagem que utiliza ferramentas de programação matemática.

Em um trabalho posterior de [Rajagopalan e Karimi, 1989] assumiram os tempos de transferência e estabelecimento não negligenciáveis para o cálculo do instante de término das tarefas em Processos multiprodutos em série com armazenagem mista. As demais hipótese do processo assumidas neste trabalho foram as mesmas de [Wiede e Reklaitis, 1987] e [Ku e Karimi, 1988], considerando a armazenagem

inter-estágio, FIG. 2.7.

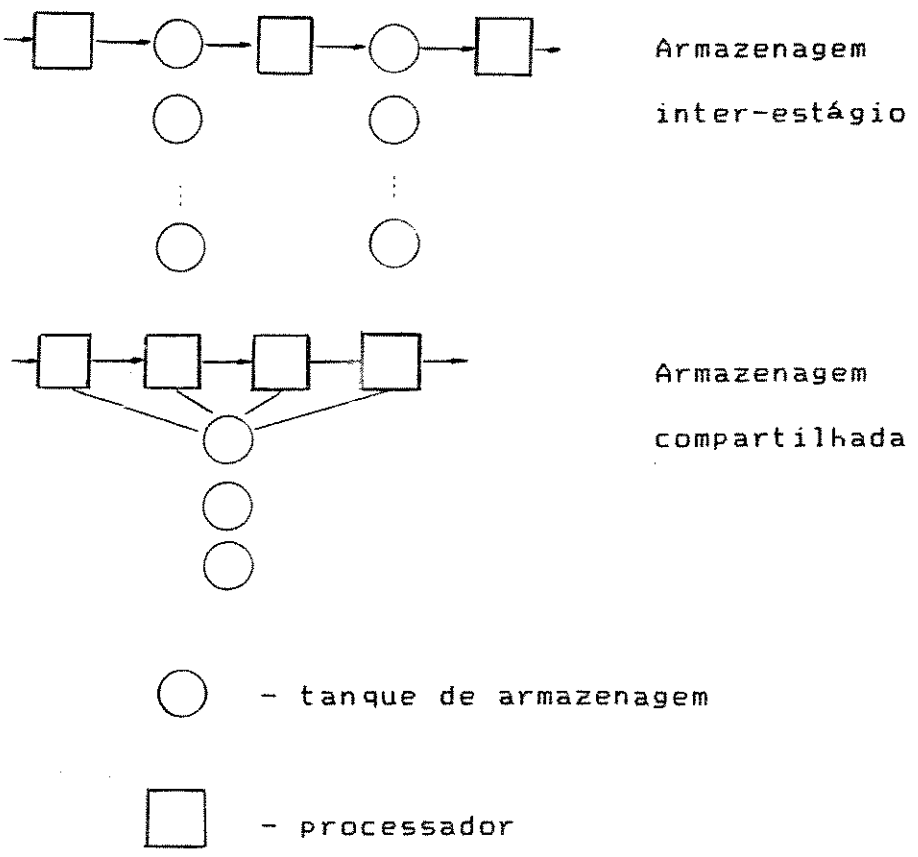


FIG. 2.7 - Armazenagem compartilhada e inter-estágio

A importância deste trabalho foi o fato de ser o primeiro a analisar a programação de flowshops MIS seriais com tempos de transferências e estabelecimentos não negligenciáveis. Foram desenvolvidas relações de recorrências baseadas na suposição de um bloco ZW. Assim duas ou mais tarefas estando dentro deste bloco e necessitando de armazenagem no mesmo

instante dá origem a um conflito. Uma vez que não é possível alocar todas as tarefas que requisitem armazenagem ocorre um atraso do início do processamento de uma destas eliminando assim este conflito. As tarefas estando fora deste bloco aguardam no próprio processador caso necessite de armazenagem e esta não esta não é disponível.

Eles desenvolveram exclusivamente relações de recorrências para o cálculo dos "completion times" a serem incorporadas a abordagens de solução do problema de programação de produção em flowshops com sequências de permutação.

No trabalho posterior de [Ku e Karimi, 1990], foram definidos dois tipos de políticas FIS : ZW/FIS e NIS/FIS. Foram comparadas duas estratégias de alocação de armazenagem : uma baseada na prioridade do produto e outra na prioridade do evento. Foi também comparado o desempenho de um algoritmo desenvolvido neste trabalho com aquele de [Wiede e Reklaitis , 1987]. Finalmente, usando os dois algoritmos FIS, foi desenvolvido um algoritmo geral para determinação da programação de produção em um processo de multiprodutoss em série com armazenagem limitada e uma mistura arbitrária de políticas de armazenagem. As deamis hipóteses admitidas no trabalho de [Ku e Karimi, 1990] foram as mesmas de [Wiede e Reklaitis, 1987].

Foram mostrados neste trabalho [Ku e Karimi, 1990] resultados enfatizando a superioridade do algoritmo ZW/FIS e NIS/FIS deles sobre o de [Wiede e Reklaitis, 1987] para as

mesmas combinações N x M.

[Leisten, 1990] tratou o problema de armazenagem limitada utilizando heurísticas para resolver os problemas NP-completo (não-polinomiais ou não descritos por um polinômio). Estas heurísticas foram adaptadas de problemas UIS ou de ZW ou foram desenvolvidas somente para problemas discutidos em seu trabalho.

Um dos mais recentes trabalhos na literatura, desenvolvido por [Das, Cummings e Le Van, 1990] mostra um estudo da aproximação "Simulated Annealing " para o problema de sequenciamento, para a minimização de makespans em plantas bateladas multiprodutos, sob a hipótese de uma "sequência de permutação". Neste trabalho foi realizado um estudo comparativo de diferentes versões do algoritmo " Simulated Annealing " para a minimização do makespan de diversos problemas flowshops (UIS,NIS,ZW,FIS e MIS). Os autores também fazem uma comparação com o desempenho da Heurística IMS de [Rajagopalan e Karimi, 1989].

" Simulated Annealing " é um algoritmo para otimização de multivariável de um único objetivo, baseado no método Monte-Carlo para simulação de sistemas físicos. A teoria de Markov necessita de uma rigorosa base matemática para o "Simulated Annealing" Este provê uma solução aproximada para os problemas NP-Completos. Na prática, ele mantém uma solução polinomial no tempo, a qual, em geral, é subótima para um

problema polinomial no tempo [Das, Cummings e Le Van, 1990] .

As soluções do "Simulated Annealing" foram comparadas com soluções exatas obtidas por uma completa busca enumerativa para tamanhos de problemas acima de oito produtos. Com base nos resultados apresentados neste trabalho os autores fazem perceber que o "Schedule Annealing" de Aartes de Van Laahoven, uma das versões do " Simuladed Annealing" foi superior ao " schedule annealing " exponencial (outra versão). Quanto as duas outras versões : o Algoritmo Metrópolis para aceitação de movimentos foi melhor que o Algoritmo Glauber. Cada das quatro versões deste foi significativamente melhor que a heurística IMS utilizada, embora cada uma destas tenha examinado uma maior porcentagem do total de busca do que a heurística mencionada. Com o aumento do tamanho do problema notou-se que a porcentagem de espaço de busca total pelas quatro versões decresceu.

O " Simulated Annealing " mostrou altos tempos de CPU, que não foi devido somente ao tamanho do problema, mas também devido a complexidade computacional do algoritmo de cálculo dos completion times sendo considerado. Daí, estes autores desenvolveram um " simulated annealing " mais competitivo que a heurística IMS sob o ponto de vista de menor esforço computacional. Este algoritmo rápido teve a qualidade de solução deteriorada comparada com a anterior mas mesmo assim continuou superior a IMS e sem degradação com o tamanho do

problema.

A natureza polinomial da solução dada com a utilização deste algoritmo pode ser usado para resolver grandes problemas e constitui uma técnica para a programação de processos de plantas bateladas de multiprodutos.

CAPÍTULO 3

Planejamento de Produção em Flowshops

3.1 – Introdução

O problema de programação de produção de plantas bateladas envolve recursos tais como tempo, armazenagem intermediária, mão-de-obra, utilidades, etc. Neste trabalho é admitido que o recurso armazenagem intermediária é compartilhada entre os vários estágios. Como já mencionado em seções anteriores, é considerado o problema geral onde todos as tarefas seguem o mesmo caminho de processamento. Cada estágio tem uma única unidade batelada. A planta tem um número finito de tanques de armazenagem, os quais podem ser usados temporariamente para armazenar bateladas de tarefas oriundas de qualquer estágio.

Foi desenvolvido, um algoritmo que constrói o perfil de armazenagem compartilhada para um dado problema flowshop com armazenagem ilimitada (UIS) e finita (FIS).

3.2 – Considerações

Neste trabalho foi considerado que a armazenagem intermediária é compartilhada entre os múltiplos estágios e as unidades de processamentos podem também manter, temporiariamente tarefas completadas, já que os produtos intermediários foram considerados estáveis. A política FIS foi dividida em dois tipos ZW/FIS e NIS/FIS, conforme é discutido na seção a seguir.

3.2.1 — A Política FIS

As definições clássicas de políticas de armazenagem intermediária [Papadimitrious e Kanellakis,1980; Dutta e Cunningham, 1975] assumem que a armazenagem está localizada entre estágios de processamentos consecutivos e não permitem que certas unidades de processamentos mantenham bateladas de produtos completas , temporariamente . Na indústria de processamento químico batelada, no entanto, se permite armazenagem dividida entre múltiplos estágios e as unidades de processamentos podem armazenar bateladas processadas, temporariamente. O trabalho de [Ku e Karimi,1990] propõe a classificação da armazenagem intermediária em duas categorias: inter-estágio e compartilhada. Conforme mostrado na FIG. 3.1 na armazenagem compartilhada, múltiplos processadores podem usar a mesma armazenagem nos diferentes tempos enquanto que na armazenagem inter-estágio os estágios de processamentos consecutivos só podem utilizar o tanque de armazenagem entre duas unidades consecutivas.

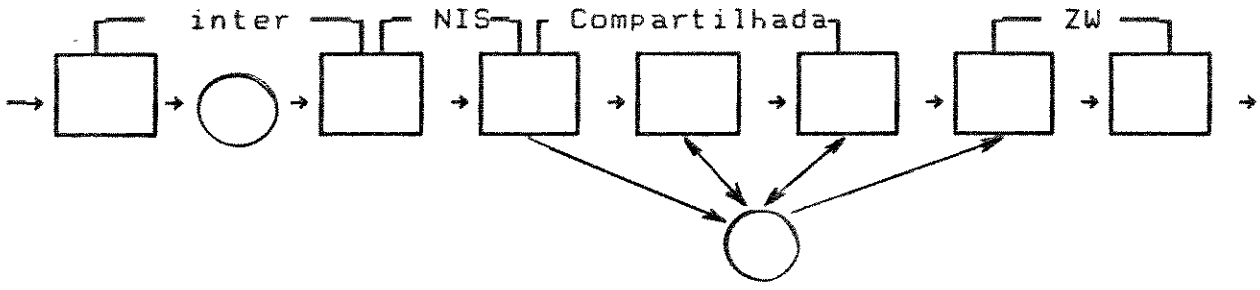
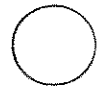


FIG. 3.1 - Um processo de multiprodutos em série com política MIS.

Onde:

 representa um tanque de armazenagem e

 representa o processador .

Seguindo os mesmos autores [Ku e Karimi,1990], a política FIS pode ser dividida em dois tipos:

-ZW/FIS, as tarefas em um bloco ZW devem ser processadas com "zero wait", ou seja elas devem ser transferidas para o processamento seguinte ou para a armazenagem tão logo ela tenha sido processada;

-política NIS/FIS, dentro do bloco NIS, a tarefa após ser processada pode ser mantida na própria unidade que a produziu durante um certo tempo antes de ser transferida fora.

Neste trabalho foram utilizadas as duas definições de políticas FIS : ZW/FIS para os casos onde se tem produtos

instáveis ou determinadas limitações no processo impedindo que o processador se comporte como tanque de armazenagem e NIS/FIS, quando o processador pode ser utilizado como tanque de armazenagem . As FIGs. 3.2 e 3.3 apresentam para uma mesma sequência o emprego desta duas políticas: ZW/FIS e NIS/FIS.

3.2.2 - Estratégias de Alocação

Quando uma unidade de armazenagem é compartilhada, dois ou mais produtos podem requerer armazenamento simultaneamente. Neste caso segundo a decisão tomada de ocupação da armazenagem, pode se provocar reflexos diferentes sobre o valor do makespan.

Uma estratégia para evitar o conflito de simultaneidade é enviar as tarefas de acordo com a ordem cronológica da necessidade para a armazenagem. Esta estratégia é denominada estratégia de prioridade do evento. A outra estratégia é aquela em que existe uma preferência de se enviar uma tarefa ao invés de outra ou prioridade do produto. Não existem até o presente qualquer conclusão ou evidência acerca da estratégia ótima com respeito à otimalidade da função objetivo. Como é comprovado em [Ku e Karimi, 1990], a estratégia de prioridade do produto é simples, eficiente e dá bons resultados e porisso foi utilizada no desenvolvimento dos algoritmos deste trabalho.

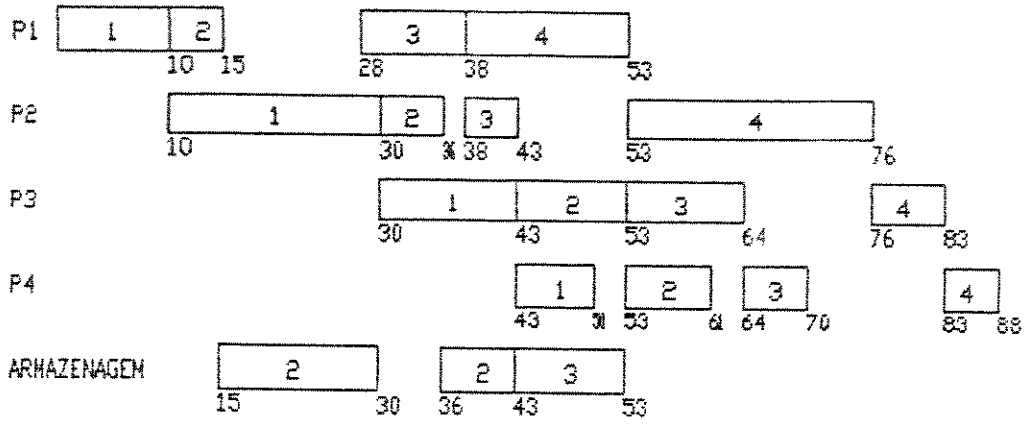


FIG. 3.2 – Modo ZW/FIS para a sequência: 1-2-3-4

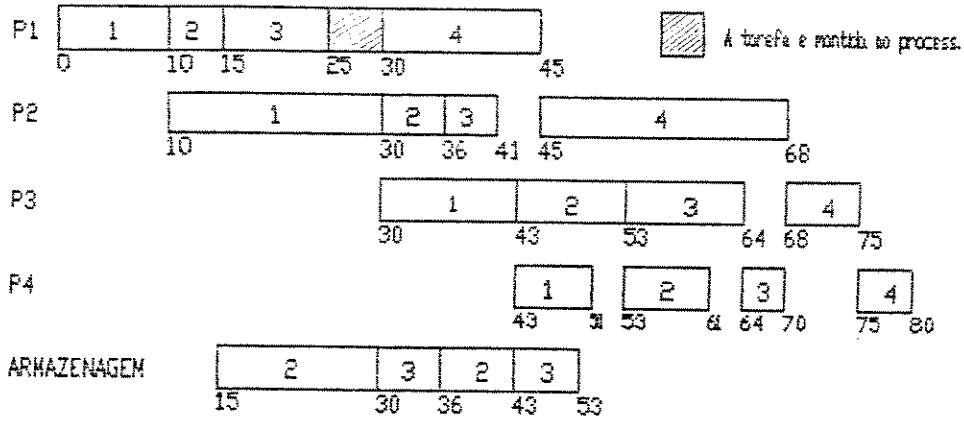


FIG. 3.3 – Modo NIS/FIS, sequência: 1-2-3-4

3.3 – Hipóteses

Foram admitidas as seguintes hipóteses para o processamento de N tarefas em uma planta multiproduto estrita (flowshop):

1. Todos os produtos são produzidos na mesma ordem em cada unidade de processamento (Sequências de Permutação).
2. A operação é não-preemptiva, isto é, uma operação não pode ser interrrompida até que seja completada.
3. Um processador não pode processar mais do que uma tarefa por vez e nem uma uma tarefa pode ser processada por mais de um processador por vez.
4. Uma unidade de armazenagem pode manter somente uma tarefa por vez.
5. A política de armazenagem do último estágio de processamento é UIS e todos as tarefas são disponíveis para processamento no tempo zero.
6. Os tempos requeridos para transferência de tarefas de um processador para outro ou de um processador para um tanque de armazenagem ou de um tanque de armazenagem para o próximo processador são considerados.

3.4 — Cálculo do instante de término das tarefas (Completion Time)

Para o cálculo dos instantes de término das tarefas propriamente dito é necessário utilizar expressões analíticas e/ou relações de recorrências. Então, o problema de determinação do Completion Time para os processos seriais pode

ser definido como:

Dada a sequência de tarefas em todas as unidades de processamentos e dados:

1. tempos de processamentos t_{ij};
2. tempos de transferências a_{ij};
3. tempos de estabelecimentos s_{ijk};
4. número de unidades de armazenagem;
5. política de armazenagem;
6. política de gerenciamento dos conflitos na demanda de recursos compartilhados; derivar relações de recorrências para C_{ij}, onde: C_{ij} é o instante no qual o produto i começa a ser transferido da unidade j.

3.4.1. — O modo UIS

Neste caso a armazenagem intermediária é ilimitada e o cálculo do "Completion Time" é mais simples bastando definir o instante de início da operação e imediatamente o instante final é também conhecido.

3.4.1.1 — O algoritmo de cálculo do Completion Time para o modo UIS

O algoritmo pemite desenvolver para um dado sistema o perfil de armazenagem compartilhado, ou seja , ele dá o número de tanques necessários para que este sistema funcione

como um UIS, mantendo um mínimo "makespan".

As relações utilizadas para o cálculo do instante de término das tarefas são dadas a seguir:

$$C_{i,j} = t_{i,j-1} + a_{i,j} \qquad (3.1)$$

$$CS_{i,j} = C_{i,j} + s_{i,i+1,j} \qquad (3.2)$$

$$CI_{i,j} = CS_{i-1,j} \qquad (3.3)$$

$$CIE_{i,j} = CI_{i,j} + a_{i,j-1} \qquad (3.4)$$

$$CF_{i,j} = CIE_{i,j} + t_{i,j} \qquad (3.5)$$

Onde:

$a_{i,j}$ = tempo de transferência da tarefa i fora do processador j;

$t_{i,j}$ = tempo de processamento da tarefa i no processador j;

$s_{i,k,j}$ = tempo de estabelecimento ou de preparação necessário para que o processador j possa receber a tarefa k após liberar a tarefa i.

Então para um caso genérico de duas tarefas (1 e 2) a serem processadas em duas máquinas se teria a Carta de Gantt correspondente a FIG. 3.4.

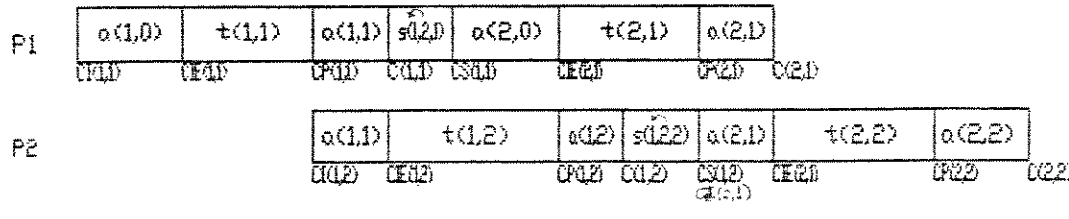


FIG. 3.4 — Instante de término das tarefas

3.4.2 — O sistema NIS/FIS

Neste caso estão disponíveis Z tanques de armazenagem disponíveis para todas as unidades de processamentos.

3.4.2.1 — O algoritmo de Completion Time para o NIS/FIS

Como já discutido anteriormente, a estratégia de prioridade do evento foi utilizada como estratégia de alocação das tarefas para a armazenagem.

Supondo que o processamento da tarefa i está para ser iniciado em uma unidade de processamento j, e C_{ij} definido como no início desta seção, então $CP_{ij} = C_{ij} + (r_{ij} - t_{ij})$, onde r_{ij} e t_{ij} são o tempo de residência e o tempo de processamento da tarefa i na unidade j respectivamente. No modo NIS/FIS, a unidade de processamento pode manter uma tarefa completada temporiariamente como se fosse um tanque de armazenagem e então $(r_{ij} - t_{ij})$ representa o tempo no qual a unidade de processamento j mantém a tarefa i antes dela ser transferida.

Portanto, antes da tarefa i poder começar a ser processada na unidade de processamento j, duas condições devem ser satisfeitas.

Primeira: ela deve terminar o seu processamento na unidade (j-1) e deve ser transferida para a unidade j e;

Segunda: A unidade de processamento j deve estar pronta para recebê-la.

Assim, para o sistema de M processadores, as relações básicas para o cálculo do instante de término da tarefa i são dadas pelas expressões :

$$C_{ij} = \max [CP_{i(j-1)}, CS_{(i-1)j}] + a_{i(j-1)} + t_{ij} \quad i=1,N ; j=2,M \quad (3.6)$$

$$CP_{ij} = \max [CP_{i(j-1)}, CS_{(i-1)j}] + a_{i(j-1)} + r_{ij} \quad i=1,N ; j=2,M \quad (3.7)$$

onde:

CP_{ij} = instante em que a tarefa i está pronta para deixar o processador;

C_{ij} = instante em que a tarefa i começa a ser transferida do processador j;

CS_{ij} = instante em que o processador j está pronto para receber a tarefa (i + 1);

r_{ij} = tempo de residência da tarefa i no processador j.

Então:

$$CP_{ij} = C_{ij} + (r_{ij} - t_{ij}) \quad (3.8)$$

$$CS_{ij} = C_{ij} + a_{ij} + s_{i(i+1)j} \quad (3.9)$$

Com a utilização destas equações pode se realizar a programação das tarefas, uma por uma, utilizando as Eqs. (3.6), (3.7) e (3.8) isto é, primeiro deve se fazer a programação da tarefa 1 em todos os processadores j =2,M; então fazer o mesmo para a tarefa 2 em todos os processadores e daí

por diante.

Cada r_{ij} inclui um tempo de espera que depende da disponibilidade da armazenagem e do processador seguinte. Se $i=1$ ou $j=M$, é claro que nenhuma espera está envolvida e $r_{ij} = t_{ij}$. De outra forma três situações podem ocorrer, conforme descrito abaixo e mostrado na FIG. 3.5.

1. O processador $(j+1)$ está pronto para receber o produto i, então este deve começar a ser transferido imediatamente;
2. O processador $(j+1)$ não está pronto mas a unidade de armazenagem está disponível, isto é, a unidade de armazenagem já está livre para receber o produto i. Então, nestes dois instantes $r_{ij} = t_{ij}$.
3. Quando nem a unidade $(j+1)$ está livre e nem a armazenagem intermediária está disponível, então a unidade j deve manter o produto i até que um dos dois torna-se livre. Desta forma é encontrado o tempo mais cedo na qual uma unidade de armazenagem está disponível. Se este tempo é mais cedo que $CS_{(i-1)(j+1)}$, então o produto é transferido para a armazenagem e por outro lado não. Contudo para se determinar o valor de r_{ij} para um dado i e j é primeiramente calculado C_{ij} usando a Eq. (3.8) e, então é usado um dos seguintes casos:

Caso 1 : Se $i=1$ ou $j=M$ ou $C_{ij} \geq CS_{(i-1)(j+1)}$, então $r_{ij} = t_{ij}$.

Caso 2: Mesmo se $C_{ij} < CS_{(i-1)(j+1)}$ e se uma unidade de armazenagem está livre para receber o produto i em C_{ij}, o

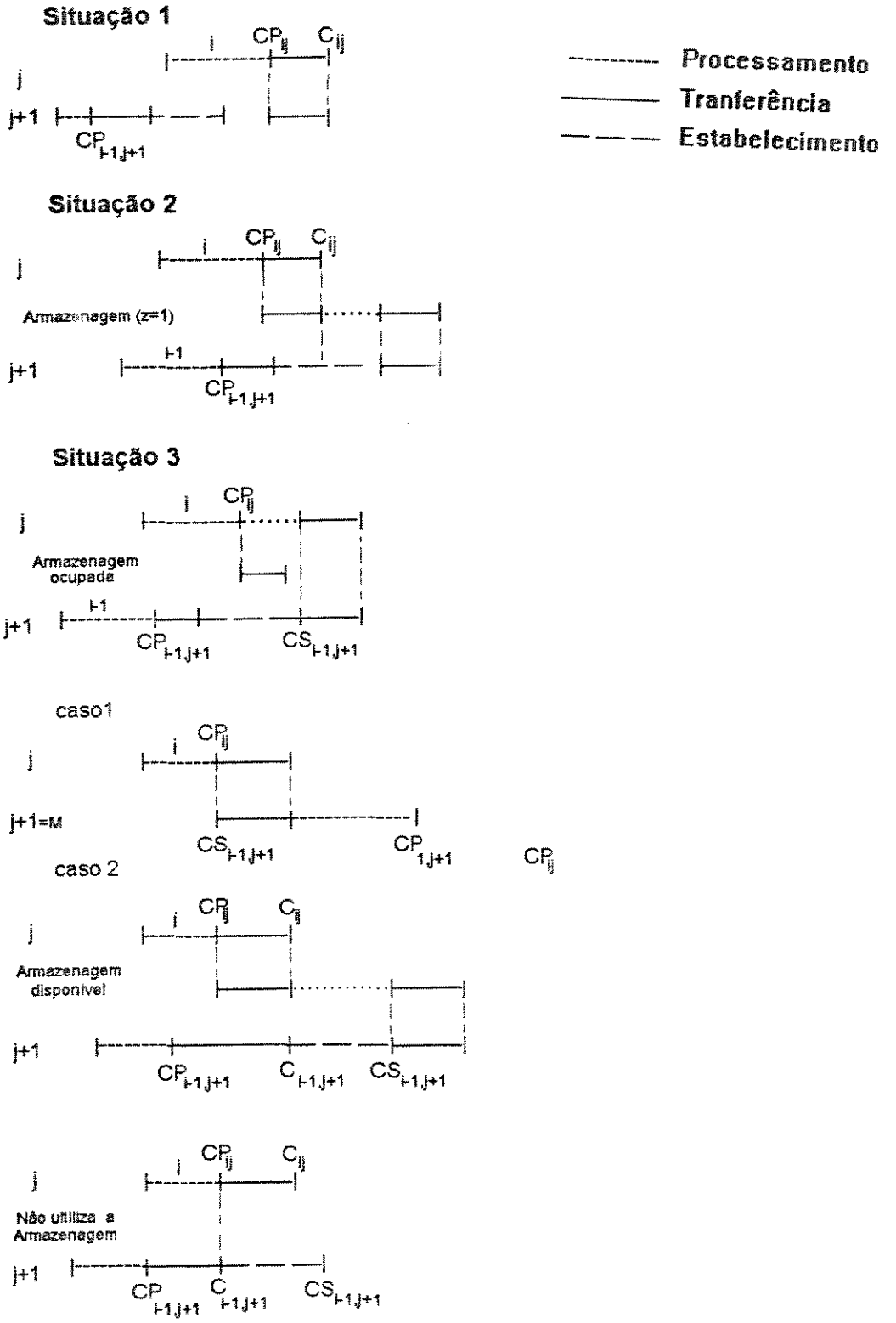


FIG. 3.5 - Cálculo do instante de término das tarefas

produto i seria transferido para ela em $CP_{ij} + a_{ij}$. Neste caso não se deve utilizar a armazenagem caso as transferências desta para a armazenagem e da armazenagem para o processador exceda o tempo $CS_{(i-1)(j+1)}$ ou seja se :

$$C_{ij} + a_{ij} \geq CS_{(i-1)(j+1)} \qquad (3.10)$$

Se acontecer isto a tarefa i permanece na unidade j até que o processador j+1 fique livre ou seja até o momento $CS_{(i-1)(j+1)}$, então $r_{ij} = CS_{(i-1)(j+1)} - C_{ij} + t_{ij}$.

Caso 3: Se a Eq. (3.10) não é obedecida ou seja é mais conveniente utilizar a armazenagem na duração de tempo [C_{ij} , $CS_{(i-1)(j+1)} + a_{ij} + s_{ij}$]. Esta duração de tempo inclui ambos : os tempos de transferências (entrada e saída) e também o tempo de estabelecimento para a unidade após liberar a tarefa i. Se uma unidade de armazenagem é disponível para esta duração então $r_{ij} = t_{ij}$. O teste de disponibilidade de armazenagem é mostrado na secção a seguir.

Caso 4 : Se não existe nenhuma possibilidade de se enviar a tarefa para a armazenagem , então ela deve esperar no processador j até que o processador $(j+1)$ torna-se livre, então $r_{ij} = CS_{(i-1)(j+1)} - C_{ij} + t_{ij}$.

3.4.2.2 — Teste de verificação da disponibilidade de armazenagem

O objetivo deste teste de checagem de disponibilidade de armazenagem é verificarar se uma unidade de

armazenagem está disponível para estocar uma tarefa em um intervalo de tempo aqui denominado [Sint,Sout].

Para verificar a disponibilidade de armazenagem foi definido um termo A(j,k) que representa o tempo em que a tarefa entra em uma unidade de armazenagem ou sai da mesma.

O valor de A(j,k) pode ser : +1 se a tarefa entra no tanque de armazenagem no tempo V0(j,k) e; -1 se o tanque de armazenagem torna-se pronto para receber a próxima tarefa no tempo V0(j,k).

Onde:

V0(j,k) = Tempo em que a unidade de armazenagem recebe uma nova tarefa ou torna-se pronta para receber uma próxima tarefa;

j = Processador;

k = Número de vezes que a armazenagem é requisitada.

O termo Z(k) representa a mudança no número de unidades de armazenagem, as quais estão disponíveis em um dado tempo (V0(j,k)), e é dado por :

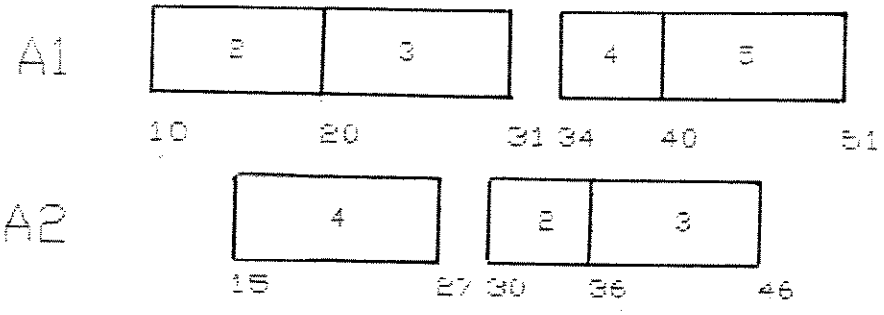
$$Z(k) = AK(j) + A(j,1) + A(j,2) + A(j,3) + \ldots + A(j,k).$$

Onde:

AK(j) = Número de tanques de armazenagem disponíveis na unidade de processamento j.

Uma unidade de armazenagem é possível para uma duração de tempo [Sint,Sout] se $Z \geq 1$ para todos este

intervalo. Para exemplificar, consideramos um sistema onde a política UIS determina a necessidade de dois tanques de armazenagem ou Z(k) = 2.0, conforme Fig.3.6 . Neste caso uma tarefa necessita de armazenagem no intervalo de tempo [28,33] e [31,34]. Conforme o teste de verificação de disponibilidade de armazenagem tem-se que no intervalo de [31,34] é possível de se enviar a tarefa para a armazenagem enquanto que no intervalo [28,33] não.



Vφ(j,k) = { 10,15,20,20,27,30,31,34,36,36,40,40,46,51 }

A(j,k) = {1, 1,-1, 1,-1, 1,-1, 1,-1, 1,-1, 1,-1,-1 }

Z(k) = { 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 0 }

{ Sint,Sout } = [28,33] Nenhuma armazenagem disponível

{ Sint,Sout } = [31,34] Armazenagem disponível

FIG. 3.6 - Verificação da disponibilidade da armazenagem intermediária.

Onde:

A1 e A2 são os dois tanques de armazenagem disponíveis.

3.4.3 — O sistema ZW/FIS

A característica principal da política ZW/FIS é que as tarefas devem ser programadas tão cedo quanto possível. Assim que seja terminado o processamento de uma tarefa, esta deve ser imediatamente enviada para a próxima unidade de processamento ou para a unidade de armazenagem. Geralmente são produtos instáveis que não podem esperar durante algum tempo até que seja possível o próximo processamento.

Como no algoritmo NIS/FIS, as tarefas são programadas, uma a uma, utilizando as Eqs.(3.6), (3.7) e (3.8) em todos os processadores j, j = 1,M. Neste caso, tem-se também que $CP_{ij} = C_{ij}$, como não é permitida armazenagem na da unidade de processamento. É feita também a checagem da armazenagem todas as vezes que ela for requisitada. Quando a armazenagem for solicitada e, simultaneamente, não houver tanque disponível e o processador seguinte estiver ocupado, então o instante de início da tarefa é atrasado até compatibilizar o processamento com a restrição ZW.

3.5 — Comparações de resultados com a literatura

Foram feitas várias comparações dos resultados obtidos utilizando o algoritmo para a programação da produção

de flowshops FIS, desenvolvido neste trabalho com alguns exemplos encontrados na literatura. A seguir serão apresentadas comparações entre a técnica BAB desenvolvida neste trabalho e os artigos utilizados para comparações :

A) "Determination of Completion Times for Serial Multiproducts Processes -1", Wiede e Reklaitis, 1987.

DADOS :

N = 4, M = 2 e

Z = 1

TAB. 3.1 - Tempos de processamentos

	TAREFAS			
PROCESSADORES	1	2	3	4
1	10	10	5	30
2	20	10	15	5

RESULTADOS :

Sequência proposta pelos autores: 1-2-3-4.

Sequência proposta no trabalho: 1-3-2-4.

Em ambos os casos foi considerado os tempos de transferência e estabelecimento iguais a zero. Foi gerado a sequência considerando Z=1 (1 tanque de armazenagem

intermediário) e as As FIGS. 3.7 e 3.8 mostram resultados para a política UIS e FIS utilizando o algoritmo deste trabalho e o resultado para a política FIS obtida pelos autores em questão

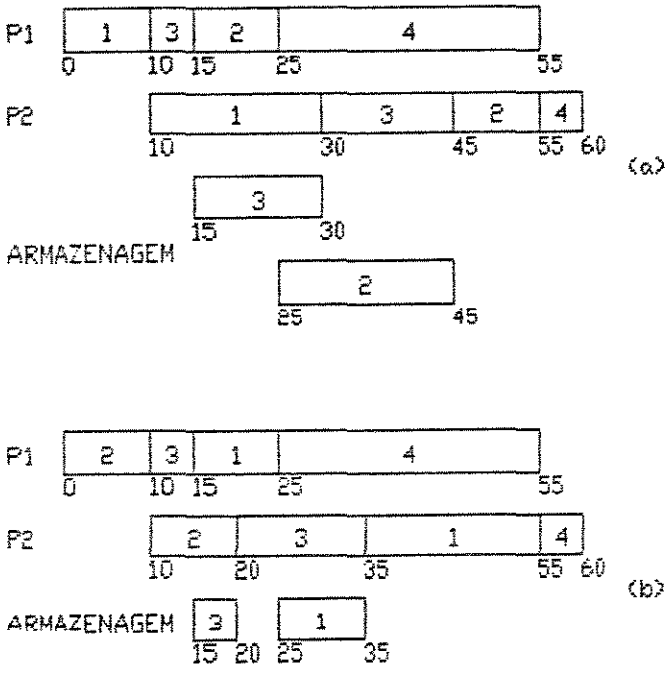


FIG. 3.7 - Solução para um problema utilizando a técnica BAB desenvolvida neste trabalho.

(a) -Modo UIS e (b) - modo FIS

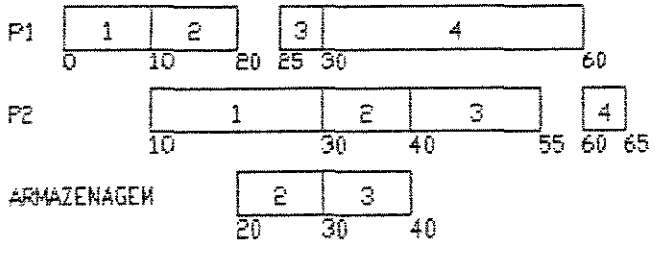


FIG. 3.8 - Solução dada pelo algoritmo subótimo de Wiede e Reklaitis para o mesmo exemplo referente a FIG.3.7

Neste Artigo em questão, o problema é solucionado com a utilização de algoritmos subótimos. A comparação é então feita entre a solução dada por este algoritmo e a solução dada pela utilização do método BAB desenvolvido neste trabalho.

Nota-se através das FIGs. 3.7 e 3.8 que o algoritmo desenvolvido nesta tese apresentou um desempenho melhor do que o de [Wiede e Reklaitis, 1987] para o mesmo problema, obtendo uma sequência de makespan igual a 60 contra 65 deste autores. Isto pode significar que esta sequência apresentada pelo artigo de [Wiede e Reklaitis,1987] não é uma sequência melhor do que a resultante deste trabalho e, portanto não pode ser tida como uma melhor sequência possível para o caso FIS de um tanque de armazenagem disponível.

B) " Completion Time Algorithms for Serial Multiproduct Batch Processes with Shared Storage", de Hong-Ming Ku e Karimi, 1990.

B.1) Exemplo utilizando a política ZW/FIS e estratégia de prioridade do evento (seção 2.6) apresentado neste artigo:

DADOS :

N=5; M=5, Z=1

TAB.3.3 - Tempos de Procesamentos

	TAREFAS				
PROCESSADORES	1	2	3	4	5
1	3	1	4	5	9
2	4	3	3	12	5
3	6	3	7	6	3
4	8	2	10	3	4
5	12	2	5	6	4

RESULTADOS :

A sequência proposta pelos autores é 1-2-3-4-5.

Sequência proposta no trabalho 1-2-3-4-5

Em ambos os casos, os tempos de transferências e estabelecimentos foram considerados iguais a zero. Os resultados são mostrados nas FIGS. 3.9 e 3.10.

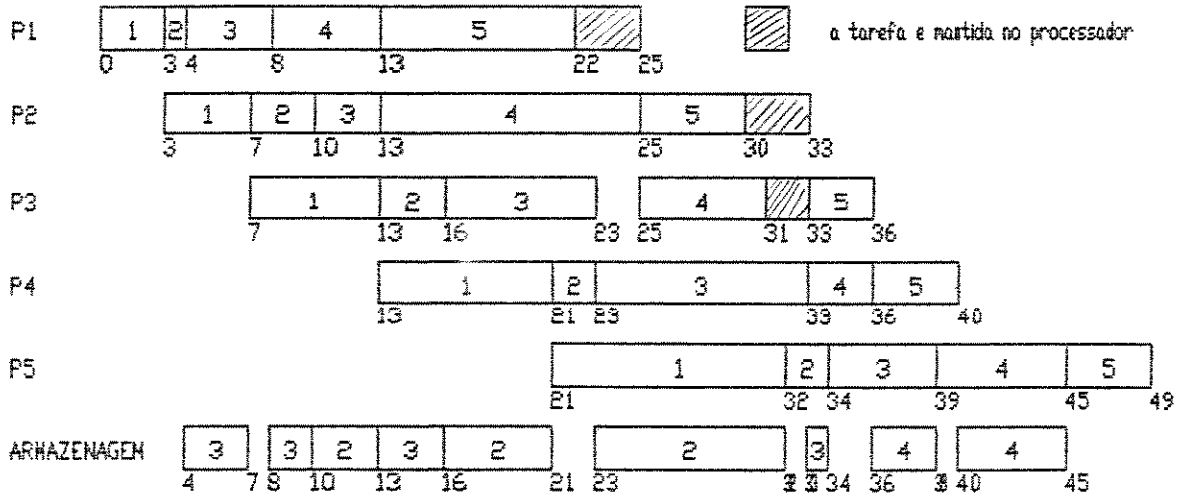


FIG. 3.9 - Solução dada pela técnica BAB, desenvolvida neste trabalho sob o modo ZW/FIS para um problema exemplo.

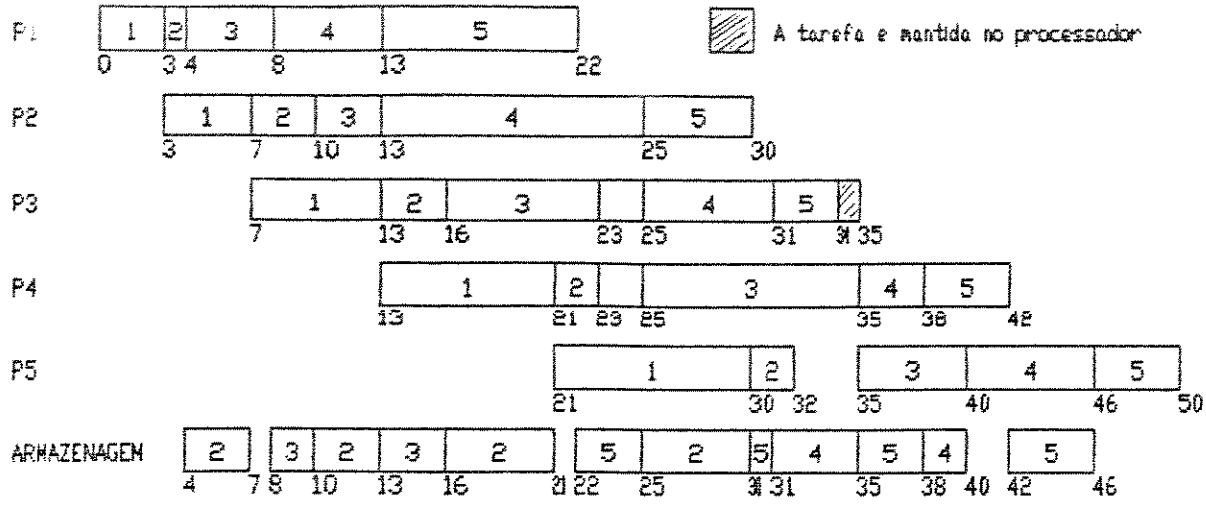


FIG. 3.10 - Solução dada por Hong-Ming Ku e Karimi para o problema exemplo referente a FIG.3.9.

Neste exemplo específico para o caso de estratégia ZW/FIS foi notado que com os mesmos dados, o algoritmo desta tese utilizando a técnica BAB apresentou para a mesma sequência de Hong-Ming Ku e Karimi um resultado ainda melhor do que o deles em termos de mínimo makespan. Isto se deu pela utilização de um tanque de armazenagem de forma diferente (FIGS. 3.9 e 3.10). Daí pode se perceber a importância da decisão de ocupação da armazenagem de uma forma ou de outra.

B.2) Exemplo utilizando as políticas NIS/FIS e ZW/FIS

DADOS :

N=4 ; M=4 e Z=1.

TAB. 3.4 - Tempos de processamentos.

	TAREFAS			
PROCESSADORES	1	2	3	4
1	10	5	10	5
2	20	8	5	4
3	13	10	11	7
4	7	8	6	5

B.2.1) Política NIS/FIS

RESULTADOS :

A sequência tomada pelos autores é 1-2-3-4.

A sequência proposta no trabalho é 1-2-3-4. Os tempos de transferência e estabelecimento foram tomados iguais a zero.

A FIG. 3.11 mostra a Carta de Gantt para a solução do problema sob o modo NIS/FIS dada pelos autores e por este trabalho (ambos mostra a mesma solução neste caso). Com relação aos mesmos dados e considerações deste mesmo exemplo.

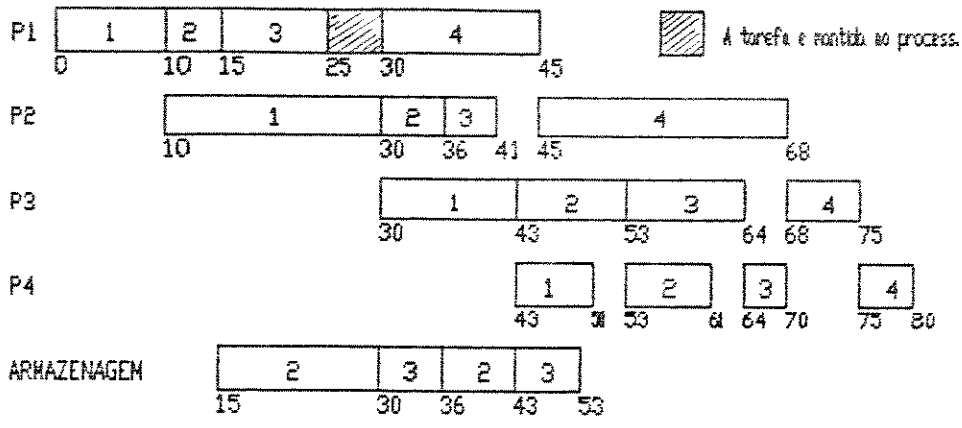


FIG 3.11 – Solução dada pela técnica BAB desenvolvidada neste trabalho e pelo algoritmo de Hong Ming Ku e Karimi para o modo NIS/FIS (mesma solução)

B.2.2) Política ZW/FIS

RESULTADOS :

A sequência tomada pelos autores é 1-2-3-4, usando uma estratégia de prioridade do produto (seção 2.6).

A sequência proposta no trabalho é 1-2-3-4. Os tempos de transferência e estabelecimento foram tomados iguais a zero.

RESULTADOS :

A FIG. 3.12 mostra que novamente a solução dada por ambos (autor e este trabalho) é a mesma para o modo ZW/FIS.

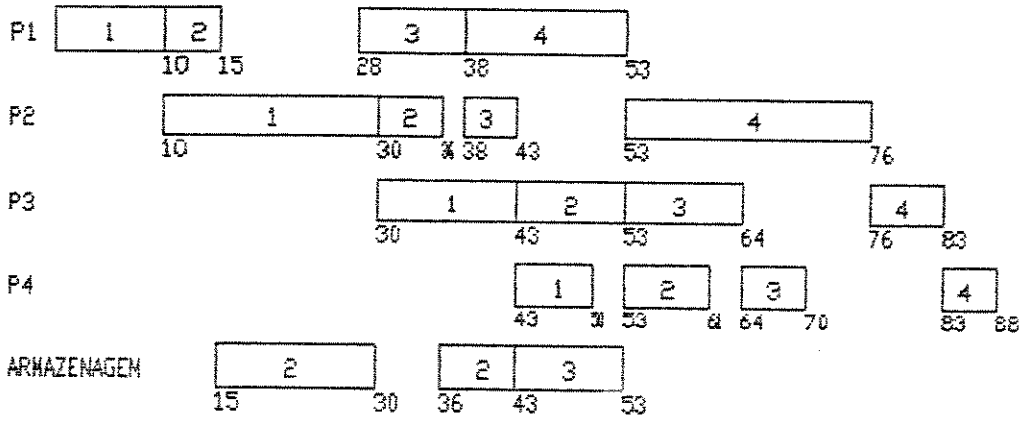


FIG. 3.12 - solução dada pela técnica BAB e pelo algoritmo de Hong Ming Ku e If. Karimi para o modo ZW/FIS (mesma solução)

Conforme as FIGS. 3.11 e 3.12 pode se notar que os algoritmos desenvolvidos nesta tese para abranger os casos NIS/FIS e ZW/FIS apresentaram os resultados idênticos ao do artigo de [Ku e Karimi , 1990]. Em outras palavras, para este exemplo específico os algoritmos desenvolvidos nesta tese funcionaram da mesma forma que o daqueles autores.

CAPÍTULO 4

Resultados

4.1 - Natureza oportunista dos resultados

Os recursos de mão-de-obra e de utilidades são normalmente disponíveis em quantidades limitadas no horizonte de tempo da programação de produção. São caracterizados por perfis de ofertas e figuram no problema de otimização como "restrições". A importância relativa do perfil temporal da utilização destes recursos no problema de otimização determinará em grande parte a complexidade e forma de abordagem do problema da programação. Estes recursos estão sujeitos, quando as condições permitirem, a flexibilidade ou relaxamento das restrições de forma a acomodar soluções aproximadas.

A principal diferença entre os recursos compartilhados tais como energia elétrica, vapor, água de refrigeração, os quais estão disponíveis em quantidades limitadas e a armazenagem intermediária está no fato de que os primeiros possuem perfis de demanda especificados pela receita tecnológica de execução da tarefa. No último, a demanda é gerada por necessidades surgidas no decorrer da programação de produção. Pode se dizer que o processador e os recursos oferecem um perfil de demanda de 100% de certeza, porisso é possível prever a sua necessidade. Isto significa que para cada sequência escolhida pode se conhecer a demanda de processadores e recursos enquanto que no caso da armazenagem intermediária a demanda é incerta porque depende da sequência tomada. Isto evidencia o caráter oportunista da demanda de armazenagem. No caso de se utilizar uma técnica BAB, o custo de cada nó é o resultado da

soma de duas parcelas: uma dependente do custo passado e uma estimativa do custo futuro. Esta função custo pode, ao longo dos ramos aumentar em função da não utilização completa do volume de recursos ofertados no intervalo de tempo comprometido pela sequência parcial em cada nó da árvore de busca. Além disso a parcela de tempo necessária para executar as tarefas remanescentes depende do volume global de recurso ofertados e também da oferta instantânea destes mesmos recursos. Se a oferta instantânea for insuficiente, o perfil de oferta é incapaz de acomodar esta nova demanda, deslocando-se o instante de início da operação da tarefa candidata até que a demanda seja compatível com a oferta. Nas condições em que a oferta é capaz de suprir a demanda, o tempo mínimo necessário para executar as tarefas não programadas dependerá apenas do compartilhamento eficiente do recurso.

No caso da armazenagem intermediária, a sua limitação de oferta pode provocar o impedimento da simultaneidade de duas ou mais operações de diferentes tarefas, mas a sua demanda futura e consequente impacto sobre o valor do makespan é função também do encadeamento das operações de uma mesma tarefa.

A maioria dos trabalhos publicados na literatura determinam a sequência ótima utilizando um algoritmo de sequenciamento que considera o modo UIS, portanto, relaxando as restrições de limitações na oferta de armazenagem intermediária. A estratégia mais comum é em seguida factibilizar a solução "UIS" tendo em conta os deslocamentos temporais adequados, afim de acomodar a limitação na oferta de armazenagem intermediária (modo FIS). Neste trabalho, como já descrito

anteriormente, trata o assunto de uma outra forma, buscando reduzir os efeitos do caráter oportunista, ou seja, para cada nó sondado pela técnica BAB, é considerado o perfil de armazenagem limitado disponível levado-se em consideração no cálculo do makespan os efeitos passados da limitação da armazenagem intermediária sobre o valor do makespan.

A seguir são mostrados alguns resultados que evidenciam o caráter oportunista de um problema de sequenciamento. Os tempos de processamentos foram gerados randomicamente, variando de 0.00 a 80.00, para 49 problemas iniciais, sendo que destes, 31 puderam ser resolvidos nos equipamentos disponíveis, conforme TAB.4.1. A maior limitação está no crescimento exponencial do tempo computacional com o aumento da dimensão N (número de tarefas). No entanto existe um efeito de M (número de equipamentos) importante sobre os tempos computacionais que impediram a resolução de problemas maiores.

Para resolução de cada problema, foi inicialmente utilizado o algoritmo que fornecia o número de tanques de armazenagem intermediários necessários para operar a planta no modo UIS e em seguida o mesmo problema foi resolvido utilizando o algoritmo que acomodava a política FIS, utilizando um número de tanques inferior a este encontrado pelo modo UIS. Para cada caso foram utilizadas as duas políticas FIS: NIS/FIS e ZW/FIS [Ku e Karimi, 1990] que utilizaram as estratégias de utilização

do processador como tanque de armazenagem e atraso do início do processamento, respectivamente, para desfazer a necessidade de um tanque para armazenagem no caso em que este não era disponível.

TAB. 4.1 - Quantidade de prblemas resolvidos em tempos razoáveis

M \ N	2	3	4	5	6	7	8
2	X	X	X	X	X		X
3	X	X	X	X	X	X	X
4	X	X	X	X	X		
5	X	X	X	X			
6	X	X	X				
7	X	X	X				
8	X	X	X				

Serão mostrados apenas os exemplos em que se utiliza o atraso do início do processamento da referida tarefa como forma de acomodar a limitação de armazenagem através de uma política ZW/FIS. Esta é uma situação mais geral englobando os processos cujas máquinas não podem ser utilizadas como tanque de armazenagem e/ou quando se tem produtos instáveis.

As TABs. 4.2 e 4.3 mostram os resultados para estes exemplos. A primeira coluna indica o número de tanques de armazenagem disponíveis para que o sistema se comporte de acordo com o modo UIS. Na segunda coluna é utilizado o algoritmo que dá

o perfil FIS, reduzindo o número de tanques disponíveis desde o número igual ao dado pelo UIS até zero.

Onde: M^* = Makespan;

Z = Número de tanques de armazenagem;

N = Número de tarefas;

M = Número de processadores.

TAB.4.2 - Valores de Makespan e de número de tanques de armaz.

PROBL. NxM	MODO UIS		MODO FIS ZW/FIS							
	M^*	Z	M^*	Z	M^*	Z	M^*	Z		
2x2	78.	0.					78.	0.		
3x2	175.	1.					175.	1.	175.	0.
4x2	191.	1.					191.	1.	193.	0.
5x2	216.	1.					216.	1.	219.	0.
6x2	291.	1.					291.	1.	297.	0.
7x2	283.	1.					283.	1.	305.	0.
8x2	341.	2.			341.	2.	341.	1.	341.	0.
2x3	109.	1.					109.	1.	109.	0.
3x3	147.	1.					147.	1.	147.	0.
4x3	211.	1.					211.	1.	211.	0.
5x3	265.	2.			265.	2.	265.	1.	265.	0.
6x3	208.	3.	208.	3.	208.	2.	212.	1.	239.	0.
7x3	349.	2.			349.	2.	349.	1.	376.	0.

TAB.4.3 - Valores de Makespan e de número de tanques de armaz.

PROBL.	MODO UIS		MODO ZW/FIS							
NxM	M*	Z	M*	Z	M*	Z	M*	Z	M*	Z
8x3	495.	1.					495.	1.	495.	0.
2x4	225.	1.					225.	1.	225.	0.
3x4	259.	1.					259.	1.	259.	0.
4x4	264.	1.					264.	1.	279.	0.
5x4	239.	1.					293.	1.	332.	0.
6x4	283.	2.			283.	2.	*	1.	326.	0.
2x5	239.	1.					239.	1.	239.	0.
3x5	255.	1.					255.	1.	289.	0.
4x5	366.	1.					366.	1.	366.	0.
5x5	336.	2.			336.	2.	366.	1.	338.	0.
2x6	193.	0.							196.	0.
3x6	355.	1.					355.	1.	355.	0.
4x6	300.	1.					300.	1.	304.	0.
2x7	412.	0.							412.	0.
3x7	412.	1.					412.	1.	412.	0.
4x7	372.	1.					372.	1.	*	0.
2x8	250.	1.					250.	1.	250.	0.
3x8	393.	1.					393.	1.	393.	0.
4x8	437.	1.					437.	1.	461.	0.

Estes resultados mostram que para alguns casos, com valores de N e M relativamente grandes por exemplo: (N=8; M=2); (N=5; M=3); (N=6; M=3); (N=3; M=8) e (N=4; M=8), ao se diminuir o número de tanques disponíveis encontrou-se para estas novas situações, sequências cujos valores de makespans foram os mesmos para o caso onde se tinha uma política UIS. Isto evidencia dois aspectos importantes a serem ressaltados:

- Dificuldade de prever na função de custo qual a influência da limitação da armazenagem sobre o custo final da sequência completa (makespan).

- Qual o esforço que deve ser direcionado no desenvolvimento de uma função de custo que calcule o impacto da limitação da armazenagem intermediária sobre o custo final da sequência.

A demanda por armazenagem ocorre quando uma dada operação é terminada num processador j, e simultaneamente o processador (j+1) não está livre. Ao mesmo tempo, outras operações podem demandar este mesmo recurso. Calcular o impacto que este conflito possa causar no caminho ainda a ser percorrido na árvore para completar a sequência de tarefas, equivale a "simular" soluções completas do sistema. Neste caso é quebrada a solução de compromisso assumida para o cálculo do custo em cada nó, a saber, a parcela de custo remanescente deve ser calculada

de forma a evitar a simulação dos ramos ainda abertos na árvore. Caso contrário é preferível simular os perfis diretamente das soluções completas. Os tempos de transferências e estabelecimentos foram considerados iguais a zero por simplificação. Os dados de tempos de processamentos que foram gerados aleatoriamente, conforme mencionado anteriormente, estão mostrados nas tabelas que antecedem cada exemplo. Os exemplos restantes demonstraram que, novamente devido o caráter oportunista, para determinados casos, a sequência FIS encontrada apresentou valores de makespan maiores que no caso UIS (caso mais comum).

A . N=8; M=2

DADOS :

TAB.4.4 - Tempos de processamentos

N \ M	1	2
1	32.	38.
2	63.	3.
3	40.	26.
4	63.	58.
5	31.	10.
6	12.	15.
7	67.	1.
8	32.	65.

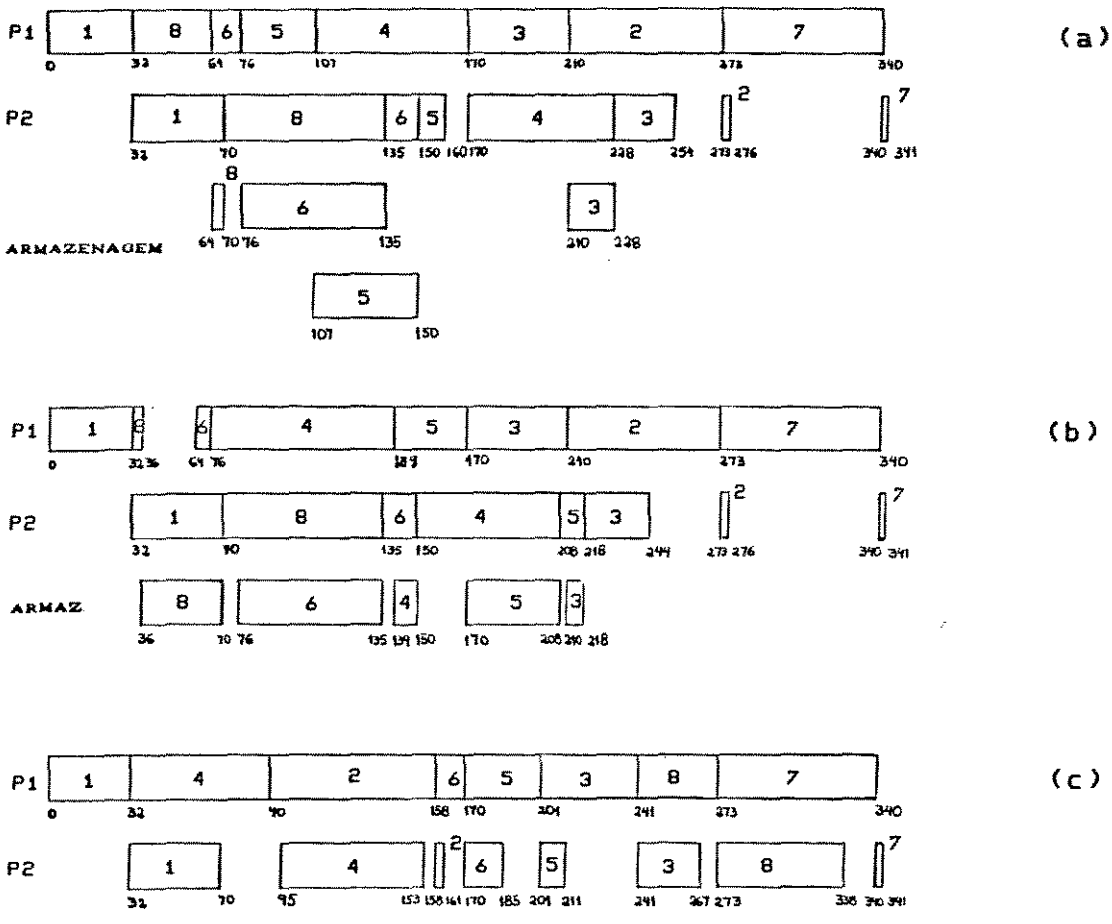


FIG. 4.1 - Carta de Gantt mostrando a alocação das tarefas, cujos dados são mostrados na TAB. 4.4, onde:

(a) - Modo UIS, mostrando a necessidade de 2 tanques de armazenagem;

(b) - Modo FIS, com utilização de 1 tanque de armazenagem e,

(c) - Modo FIS, sem utilização de armazenagem.

Observando a FIG. 4.1 nota-se que no modo UIS a seqüência ótima encontrada foi 1-8-6-5-4-3-2-7 e o makespan (M^*) igual a 341 (a). O algoritmo de sequenciamento e alocação desenvolvido neste trabalho indicou a necessidade de dois tanques de armazenagem para que o sistema se comportasse de acordo com o modo UIS. Reduzindo o número de tanques de armazenagem para um tanque apenas, foi encontrada uma seqüência diferente da anterior que permitiu utilizar o único tanque disponível e mesmo assim conseguiu-se um valor de makespan igual ao do caso UIS (b). No caso (c) desta mesma figura pode ser observado que mesmo sem a utilização de nenhum tanque, o algoritmo de sequenciamento e alocação para o modo FIS encontrou uma nova seqüência cujo makespan foi idêntico ao modo UIS (M^*= 341).

B. N=3; M=8

DADOS:

TAB.4.5 - Tempos de processamentos

N \ M	1	2	3	4	5	6	7	8
1	38.	28.	60.	34.	2.	74.	3.	74.
2	31.	30.	52.	6.	68.	11.	38.	34.
3	58.	73.	14.	65.	17.	29.	8.	60.

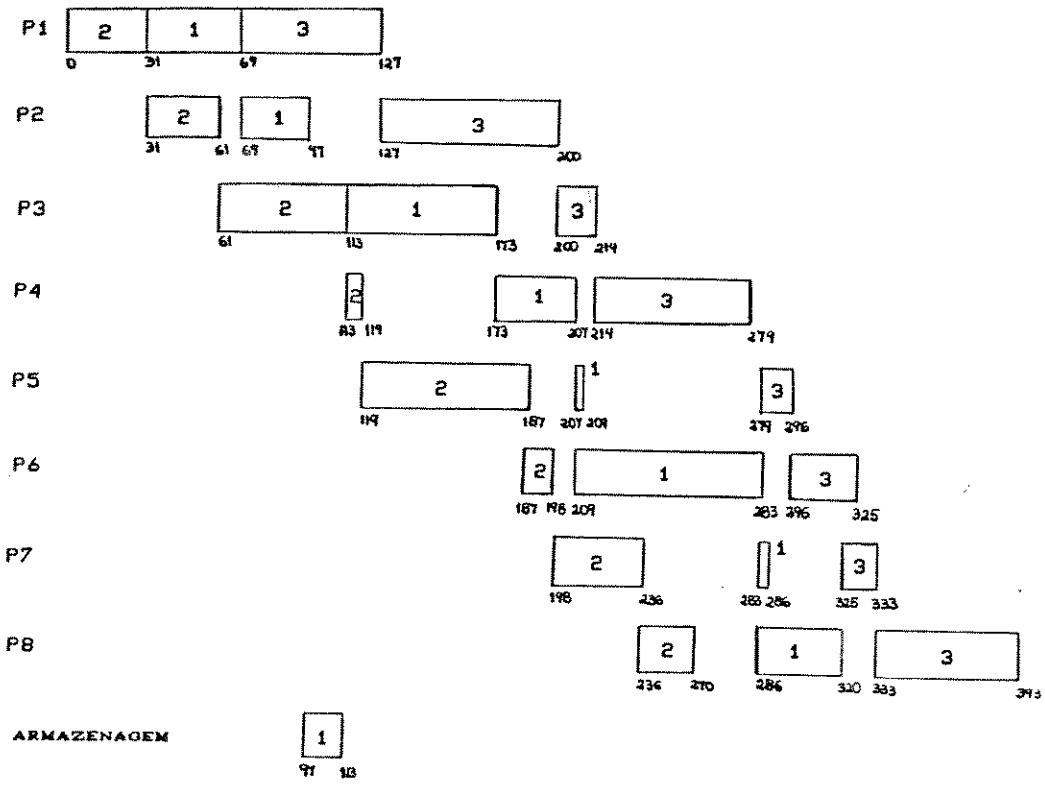


FIG. 4.2a - Carta de Gantt mostrando a alocação das tarefas, cujos dados são mostrados na TAB. 4.5 sob o modo UIS, mostrando a necessidade de 1 tanque

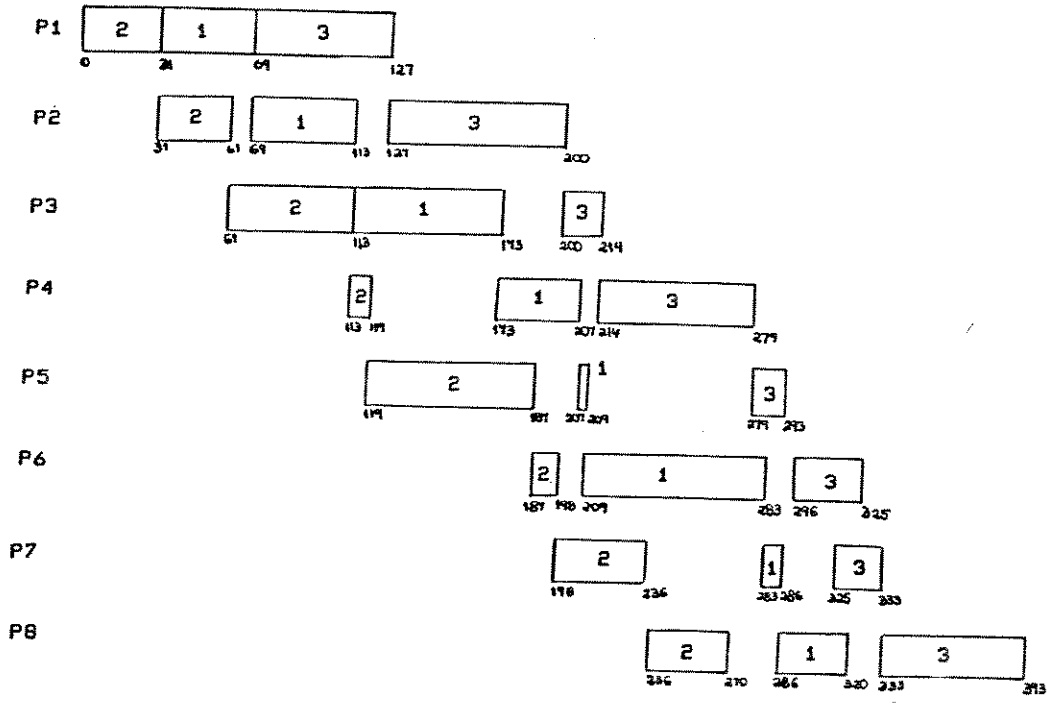


FIG. 4.2b - Carta de Gantt mostrando a alocação das tarefas, cujos dados são mostrados na TAB. 4.5 sob o modo FIS sem utilização de armazenagem.

Conforme notado nas FIGs. 4.2a e 4.2b uma mesma seqüência encontrada no caso do modo UIS pode ser estendida para o modo FIS sem a alteração do makespan. Nas duas figuras é mostrado uma mesma seqüência, primeiramente sob o modo UIS com a utilização de um tanque de armazenagem e em seguida sob o modo FIS sem utilização de armazenagem com o mesmo valor de makespan (M^*= 393).

Estes exemplos mostrados acima mostram que dependendo da seqüência ou mesmo para uma mesma seqüência, modificando a forma de alocação, os valores dos makespans podem ser o mesmo para um número de tanques diferentes. Isto não é mostrado na maioria da literatura. Nestes casos, a função custo foi capaz de incorporar o impacto do conflito da armazenagem no caminho a ser percorrido no futuro de forma que sem se realizar uma simulação completa conseguiu-se uma seqüência cujo makespan foi igual ao do caso UIS.

Outros exemplos semelhantes que demonstram o caráter oportunista destes tipos de problemas são mostrados nos ítens a seguir (FIGs. 4.3a e 4.3b e 4.4):

C. N=5; M=3

DADOS:

TAB.4.6 - Tempos de processamentos

M / N	1	2	3
1	11.	19.	33.
2	19.	46.	27.
3	31.	67.	15.
4	36.	63.	47.
5	40.	48.	11.

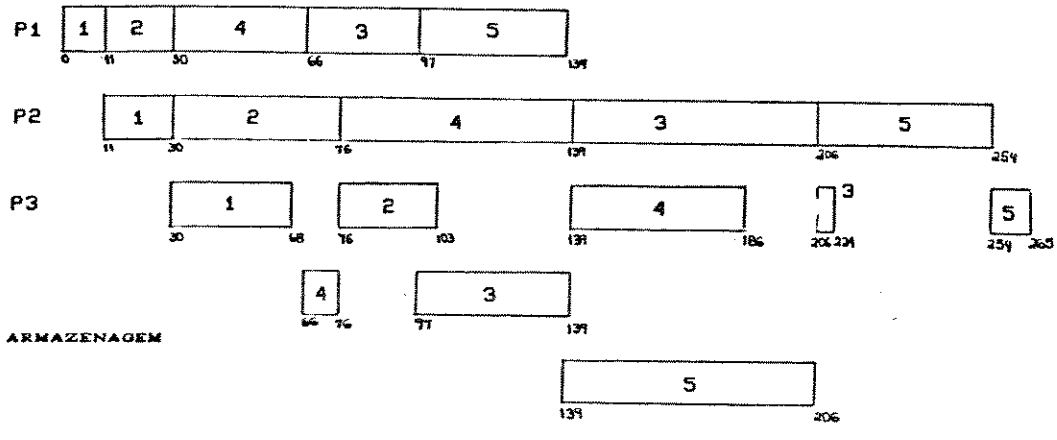


FIG. 4.3a - Carta de Gantt mostrando a alocação das tarefas nos processadores, cujos dados são mostrados na TAB.4.6, sob o modo UIS mostrando a necessidadede 2 tanques de armazenagem;

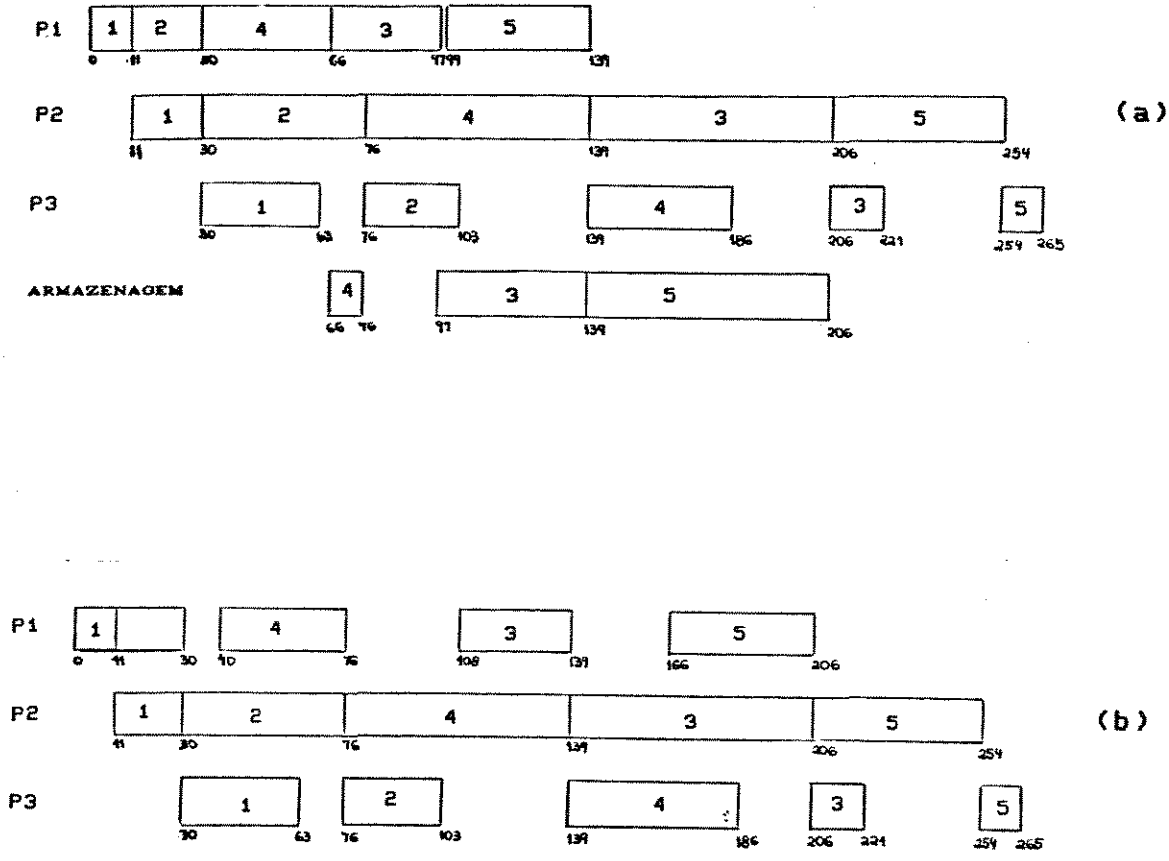


FIG. 4.3b - Carta de Gantt mostrando a alocação das tarefas nos processadores, cujos dados são mostrados na TAB.4.6, sob :

(a) - Modo FIS, com utilização de 1 tanque de armazeanagem e,

(b) - Modo FIS, sem utilização de armazenagem.

D. N=6; M=3

DADOS:

TAB 4.7 - Tempos de processamentos

N \ M	1	2	3
1	53.	61.	1.
2	26.	22.	57.
3	6.	63.	21.
4	14.	9.	18.
5	18.	13.	22.
6	5.	34.	35.

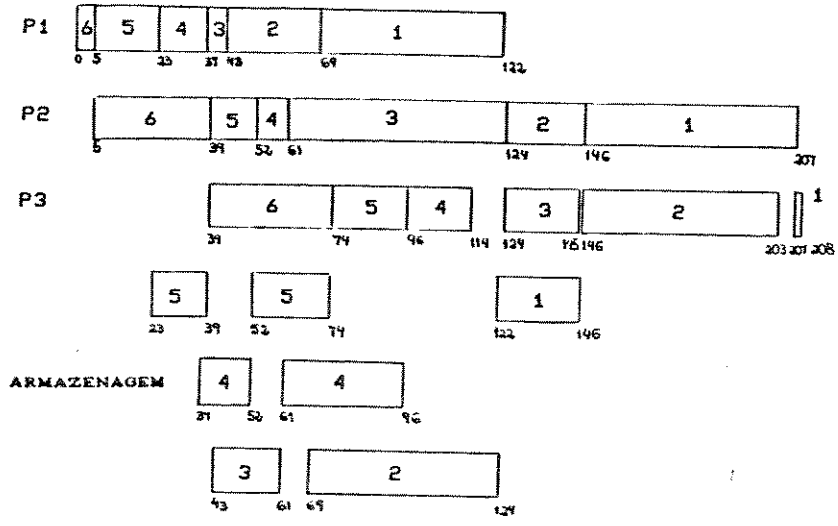


FIG 4.4a - Carta de Gantt mostrando a alocação das tarefas nos processadores, cujos dados são mostrados na TAB.4.7, sob o Modo UIS, mostrando a necessidade de 3 tanques de armazenagem.

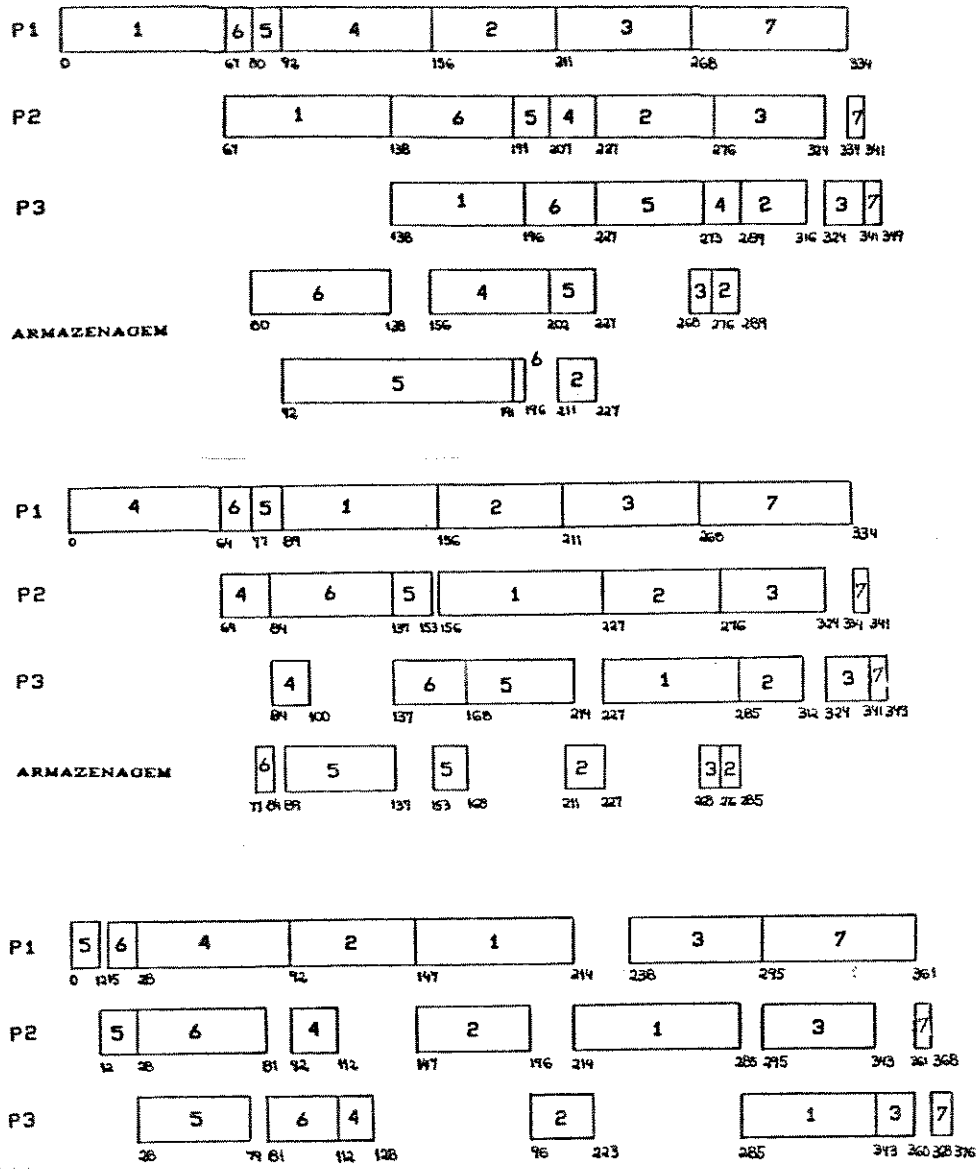


FIG. 4.4B - Carta de Gantt mostrando a alocação das tarefas nos proceesadores, cujos dados são mostrados na TAB. 4.7 sob:

(a) - Modo FIS, com utilização de 2 tanques de armazenagem;

(b) - Modo FIS, com utilização de 1 tanque de armazenagem e,

(c) - Modo FIS sem utilização de armazenagem.

As figuras pertinentes aos ítens E , F e G mostram outros casos em que ao se reduzir o número de tanques disponíveis não foi conseguido uma sequência com makespan igual ao do modo UIS:

E. N=4; M=2

DADOS:

TAB.4.8- Tempos de processamentos

N \ M	1	2
1	53.	65.
2	43.	30.
3	30.	41.
4	13.	32.

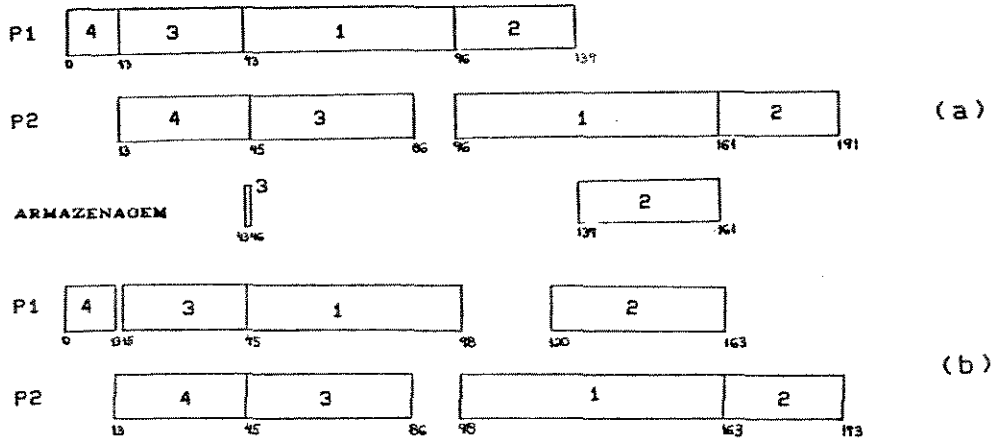


FIG. 4.5 - Carta de Gantt mostrando a alocação das tarefas nos nos processadores, cujos dados são mostrados na TAB.4.8, onde:

(a) - Modo UIS, com utilização de 1 tanque de armazenagem e,

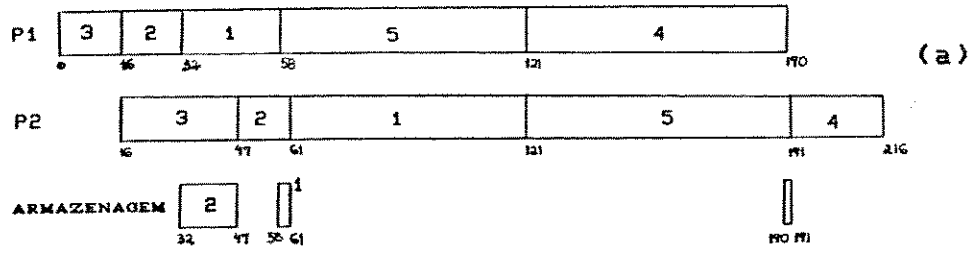
(b) - Modo FIS sem utilização de armazenagem.

F. N=5; M=2

DADOS:

TAB.4.9 - Tempos de processamentos

N \ M	1	2
1	26.	60.
2	16.	14.
3	16.	31.
4	69.	25.
5	63.	70.



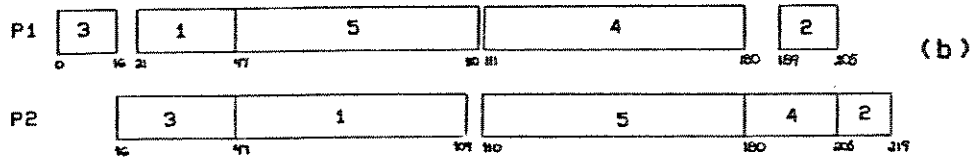


FIG. 4.6 - Carta de Gantt mostrando a alocação das tarefas nos nos processadores, cujos dados são mostrados na TAB.4.9, onde:

(a) - Modo UIS, mostrando a necessidade de 1 tanque de armazenagem e,

(b) - Modo FIS sem utilização de armazenagem.

6. N=6; M=3

DADOS:

TAB.4.10 - Tempos de processamentos

N \ M	1	2	3
1	53.	61.	1.
2	26.	22.	57.
3	6.	63.	21.
4	14.	9.	18.
5	18.	13.	22.
6	5.	34.	35.

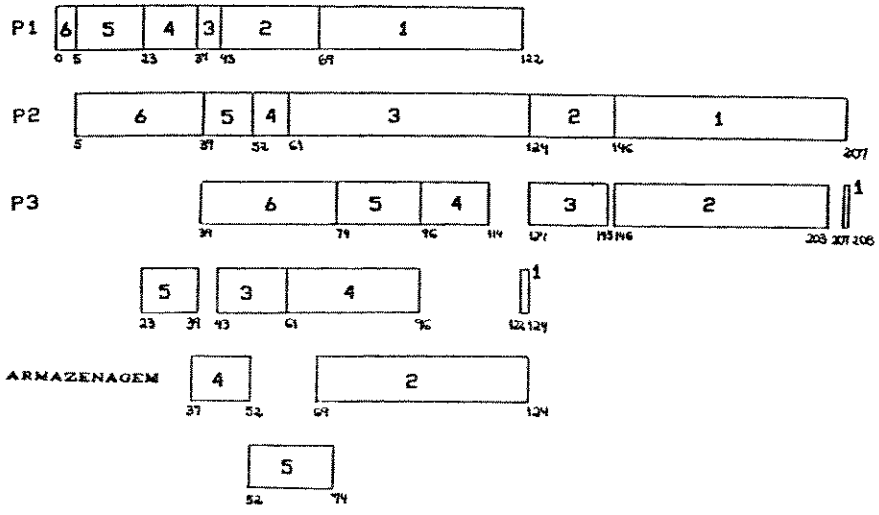
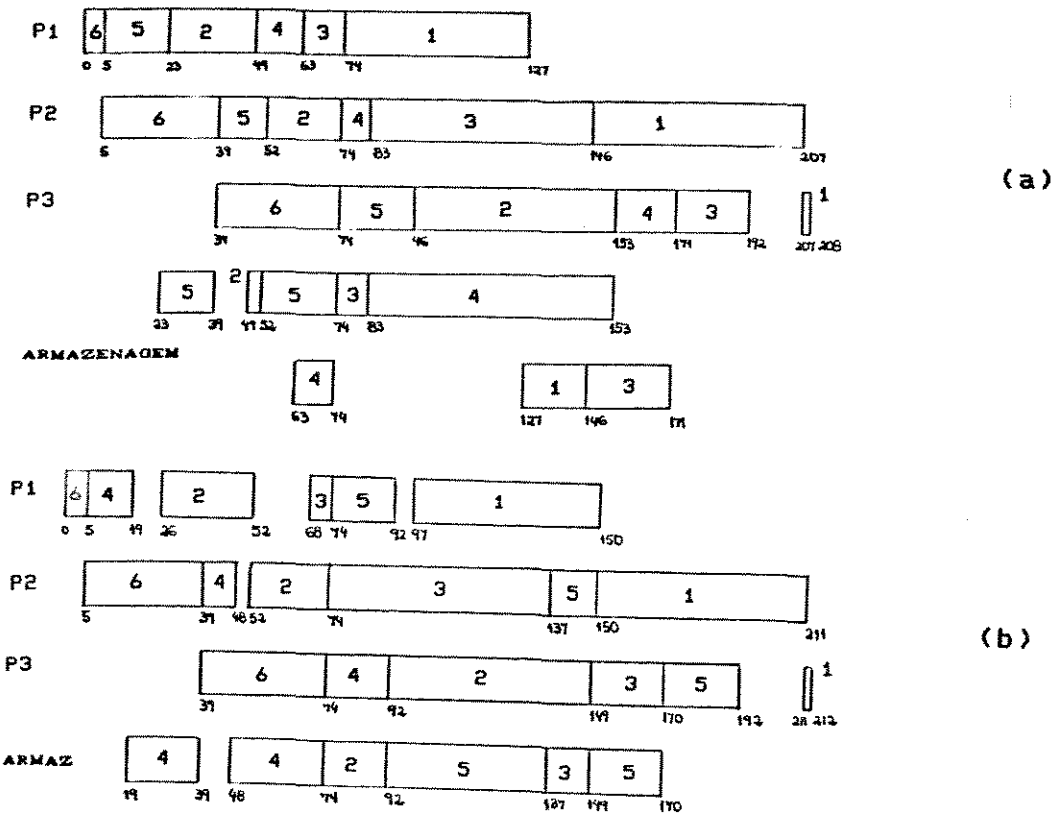


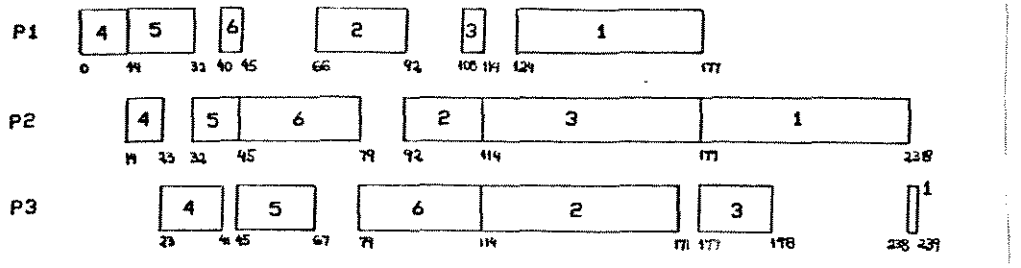
FIG.4.7a - Carta de Gantt mostrando a alocação das tarefas nos processadores, cujos dados são mostrados na TAB.4.10, sob o Modo UIS, mostrando a necessidade de 3 tanques de armazenagem.



TAB. 4.7b - Carta de Gantt mostrando a alocação das tarefas nos proceesadores, cujos dados são mostrados na TAB.4.10 , sob:

(a) - Modo FIS, com utilização de 2 tanques de armazenagem e

(b) - Modo FIS, com utilização de 1 tanque de armazeanagem e.



TAB. 4.7c - Carta de Gantt mostrando a alocação das tarefas nos proceesadores, cujos dados são mostrados na TAB.4.10 , sob o modo FIS sem utilização de armazenagem.

4.2 - Comparações entre as duas políticas NIS/FIS e ZW/FIS

Quando a armazenagem é limitada, é importante se definir a priori qual a estratégia a ser utilizada quando o recurso for limitado. Nesta seção serão estudadas as diferenças entre as políticas NIS/FIS (modo FIS com utilização do processador como tanque de armazenagem) e ZW/FIS (modo FIS com atraso do início do processamento). Inicialmente foram realizadas algumas comparações entre os resultados obtidos neste

trabalho utilizando estas duas políticas e exemplos retirados da literatura (seção 3.5.1). Nesta seção buscou-se comparar os dois modos FIS entre si para os mesmos exemplos, onde os tempos de processamentos foram gerados aleatoriamente evitando assim alguma indução nos resultados. Foi notado em alguns casos a superioridade da política NIS/FIS (processador utilizado como tanque) sobre a ZW/FIS (atraso de início do processamento), ou seja, o primeiro modo apresentou sequências com makespan menores. Os sistemas que apresentaram diferenças nos valores dos makespans estão exemplificados na TAB.4.6. Foram mostrados os valores de makespan e tempos de residência médios para os casos com número de tanques de armazenagem disponíveis iguais a zero pois somente nestes casos os valores destes foram diferentes. Estes exemplos foram selecionados do mesmo conjunto de problemas resolvidos na seção 4.1, ou seja dos 31 problema resolvidos (mencionados anteriormente).

TAB. 4.11 - Valores de makespan, flowtime médio e número de tanques para os problemas exemplos

PROBL. NxM	MODO UIS			ZW/FIS			NIS/FIS		
	M^*	$\bar{F}$	Z	M^*	$\bar{F}$	Z	M^*	$\bar{F}$	Z
7x3	349.	284.	2.	376.	231.	0.	360.	232.	0.
4x4	264.	223.	1.	279.	226.	0.	268.	220.	0.
5x4	239.	338.	1.	332.	257.	0.	293.	293.	0.
6x4	283.	219.	2.	326.	216.	0.	300.	300.	0.
3x5	255.	233.	1.	289.	245.	0.	255.	255.	0.
3x8	393.	327.	1.	398.	341.	0.	393.	393.	0.

As FIGs.: 4.8a; 4.8b; 4.8c e 4.9a e 4.9b mostram as Cartas de Gantt correspondentes a dois destes exemplos cujos tempos de processamentos são dados nas TABs. 4.12a e 4.12b respectivamente.

TAB. 4.12a - Tempos de processamentos

N \ M	1	2	3	4	5	6	7	8
1	38.	28.	60.	34.	2.	74.	3.	34.
2	31.	30.	52.	6.	68.	11.	38.	34.
3	58.	73.	14.	65.	17.	29.	8.	60.

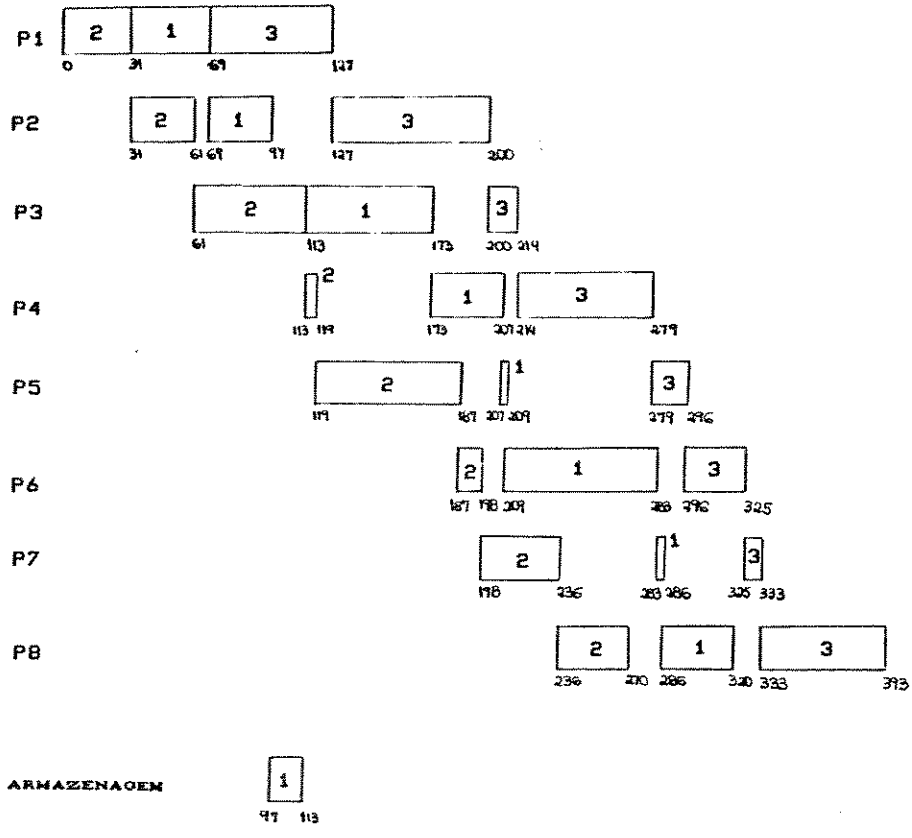


FIG. 4.8a - Modo UIS para exemplo referente a TAB. 4.12a indicando a necessidade de 1 tanque de armazenagem.

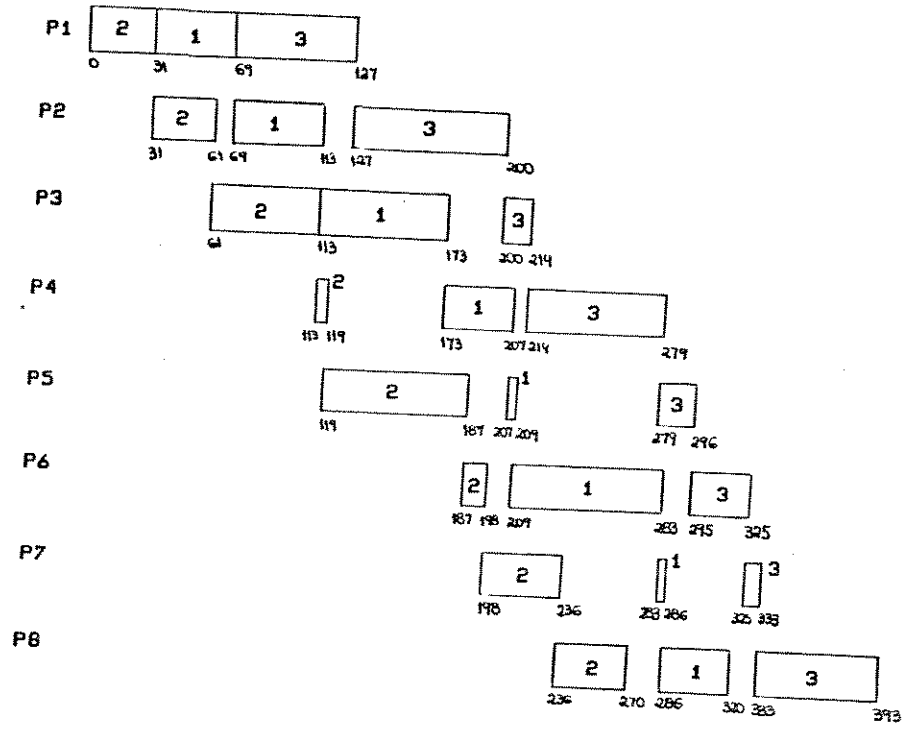


FIG. 4.8b - Modo NIS/FIS para o exemplo referente a TAB. 4.12a sem a utilização de armazenagem

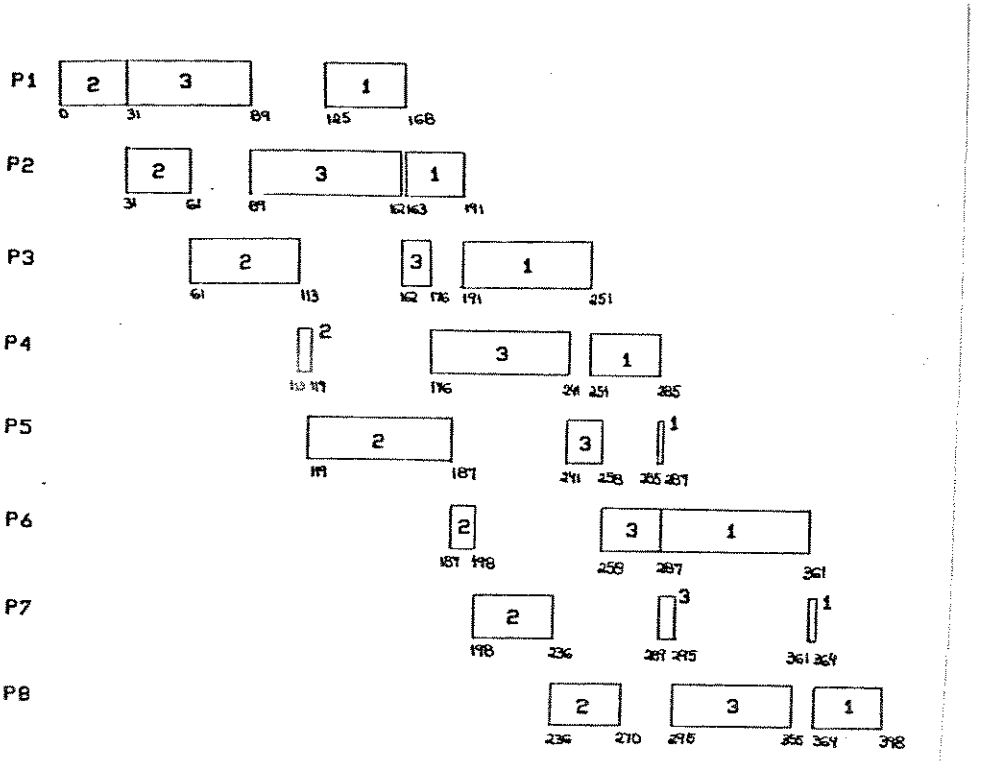


FIG. 4.8c - Modo ZW/FIS para o exemplo referente a TAB. 4.12a sem a utilização de armazenagem

TAB. 4.12b - Tempos de processamentos

N \ M	1	2	3	4
1	59.	24.	30.	39.
2	71.	20.	4.	46.
3	42.	34.	11.	0.
4	69.	42.	14.	29.
5	7.	65.	34.	65.

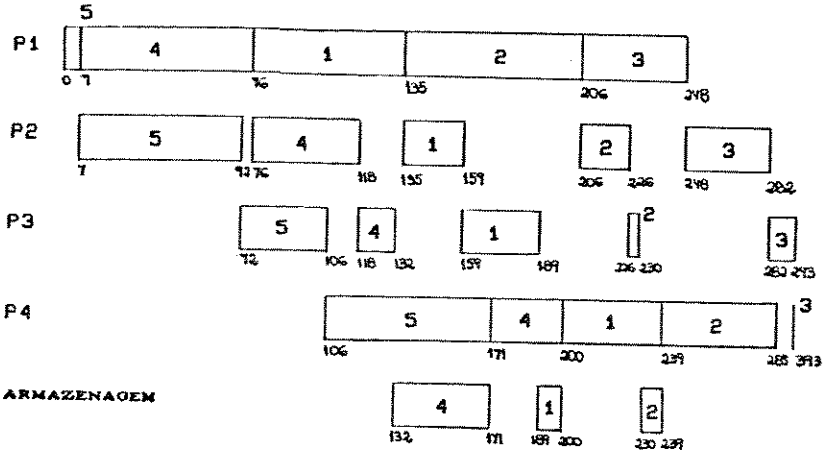
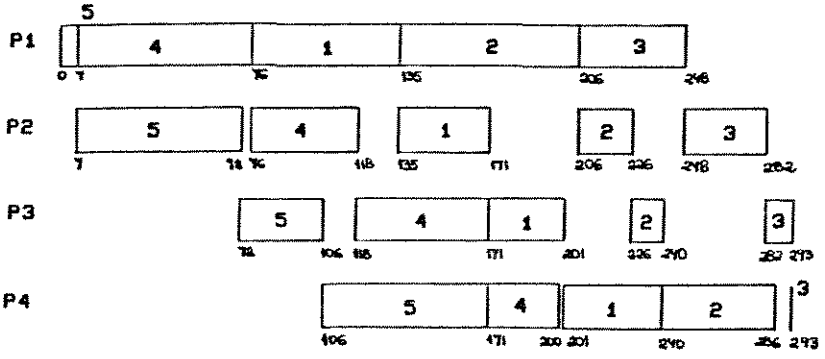
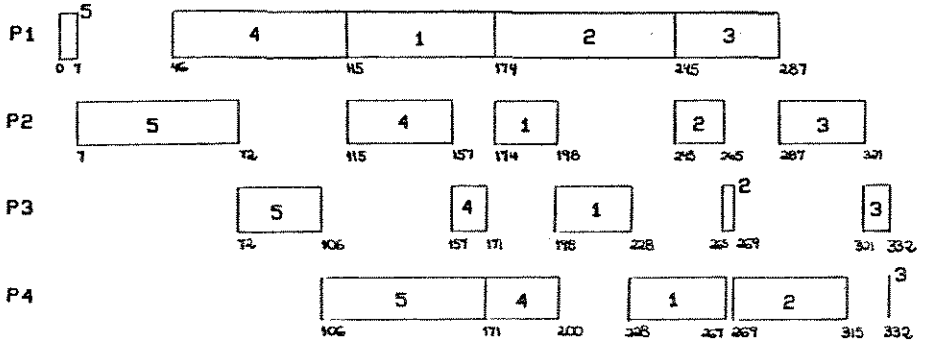


FIG. 4.8a - Modo UIS indicando a necessidade de 1 tanque de armazenagem cujos dados se referem a TAB.4.12b.



(a)



(b)

FIG. 4.8b - Carta de Gantt para o exemplo referente a TAB.4.12b

onde:

(a) - Modo NIS/FIS sem utilização de armazenagem

(b) - Modo ZW/FIS sem utilização de armazenagem.

4.3 - Comparações com heurísticas

Foram utilizadas duas heurísticas muito conhecidas: Heurística de Dannembring e a Heurística de Campbell-Dudek-Smith (CDS) para obtenção das sequências. Estas heurísticas são utilizadas dentro de uma estratégia sub-ótima de geração de sequências factíveis executada em duas etapas:

. geração de sequência e

. compatibilização da demanda e oferta de recursos (fase de factibilização a posteriori).

Para a realização destas comparações foi suprido a parte da técnica BAB de sequenciamento e com a utilização da melhor sequência possível obtida para um determinado problema fez-se a aplicação do algoritmo de cálculo do modo FIS com atraso de início de processamento (ZW/FIS). Estes resultados foram comparados com aqueles obtidos pelo algoritmo de sequenciamento e alocação para o modo FIS desenvolvido neste trabalho. Os dados de tempos de processamentos foram gerados aleatoriamente e alguns deles são mostrados nas TABs.4.13 e 4.14. As tabelas 4.15 e 4.16 mostram as comparações entre as duas heurísticas em questão utilizando os dados dos tempos de processamentos destas tabelas (4.13 e 4.14).

TAB.4.13 - Tempos de processamentos

M \ N	1	2	3	4	5	6	7	8
1	10.	5.	13.	8.	13.			
2	8.	10.	9.	5.	8.			
3	15.	5.	10.	10.	16.			
4	30.	15.	12.	14.	12.			
5	9.	13.	15.	8.	6.			
6	18.	10.	9.	14.	17.			
7	15.	8.	12.	16.	12.			
8	13.	12.	8.	5.	10.			

TAB.4.14 - tempos de processamentos

M \ N	1	2	3	4	5	6	7	8
1	3.	4.	8.	6.	9.	3.	6.	2.
2	5.	3.	2.	8.	1.	5.	2.	4.
3	8.	4.	3.	2.	6.	2.	8.	10.
4	4.	2.	8.	3.	2.	8.	1.	4.
5	6.	2.	5.	7.	2.	3.	4.	9.
6	3.	1.	8.	5.	3.	7.	2.	6.
7	5.	3.	2.	5.	2.	6.	2.	7.
8	3.	8.	1.	6.	3.	2.	8.	1.

TAB.4.15 - Comparações entre as Heurísticase o algoritmo FIS com os dados referentes a TAB. 13

PROBL. NxM	MODO FIS		HEURÍSTICA 1		HEURÍSTICA 2	
	M*	Z	M*	Z	M*	Z
6x5	139.	1.	140.	1.	140.	1.
seq.	3-1-4-2-6-5		1-3-6-4-5-2		1-3-6-4-5-2	
7x5	151.	1.	155.	1.	151.	1.
seq.	3-1-4-2-6-7-5		1-3-7-6-4-5-2		2-1-3-4-6-7-5	
4x4	94.	1.	96.	1.	94.	1.
seq.	4-3-1-2		1-4-3-2		4-3-1-2	
5x4	103.	1.	108.	1.	109.	1.
seq.	5-4-3-1-2		1-4-5-3-2		4-3-5-1-2	
6x4	117.	1.	121.	1.	120.	1.
seq.	5-4-6-3-1-2		1-4-6-5-3-2-		4-5-6-1-3-2	
7x4	132.	1.	136.	1.	135.	1.
seq.	7-5-4-6-3-1-2		1-7-4-6-5-3-2		7-4-5-6-1-3-2	
8x4	142.	1.	146.	1.	143.	1.
seq.	1-7-5-4-6-8-3-2		1-7-4-6-5-3-8-2		7-4-5-6-1-8-3-2	

HEURÍSTICA 1 - HEURÍSTICA DE DANNEMBRING
HEURÍSTICA 2 - HEURÍSTICA CDS

TAB 4.16a - Comparações entre as Heurísticas e o algoritmo FIS c/ os dados referentes a TAB. 14

PROBL. NxM	MODO FIS		HEURÍSTICA 1		HEURÍSTICA 2	
	M*	Z	M*	Z	M*	Z
8x4	52.	2.	56.	2.	53.	2.
seq.	2-6-8-7-5-1-4-3		6-2-4-1-5-8-7-3		2-7-6-5-1-8-4-3	
6x5	47.	2.	48	2.	47.	2.
seq.	6-1-5-4-3-2		6-1-5-3-4-2		6-1-5-4-3-2	
7x5	50.	2.	54.	2.	53.	2.
seq.	6-7-1-5-4-3-2		6-1-5-3-4-2-7		6-1-5-4-3-2-7	
8x5	56.	2.	59.	2.	59.	2.
seq.	6-5-7-1-8-4-3-2		6-1-8-5-3-4-2-7		6-1-8-5-4-3-2-7	
5x6	50.	2.	52.	2.	51.	2.
seq.	4-2-1-5-3		2-4-1-5-3		2-1-4-5-3	
7x6	58.	2.	61.	2.	62.	2.
seq.	6-2-4-7-1-5-3-8		7-6-2-4-1-8-5-3		2-7-6-1-4-5-3	
8x6	61.	2.	65.	2.	67.	2.
seq	6-2-4-7-1-5-3-8		7-6-2-4-1-8-5-3		6-4-1-7-3-2-8-5	
4x7	50.	2.	53.	2.	52.	2.
seq	1-2-3-4		1-3-4-2		1-3-2-4	
5x7	54.	2.	56.	2.	55.	2.
seq	1-5-3-2-4		1-3-5-4-2		1-3-5-2-4	

HEURÍSTICA 1 - HEURÍSTICA DE DANNEMBRING
HEURÍSTICA 2 - HEURÍSTICA CDS

TAB 4.16b - Comparações entre as Heurísticas e o algoritmo FIS p/ os dados referentes a TAB. 14

PROBL. NxM	MODO FIS		HEURÍSTICA 1		HEURÍSTICA 2	
	M^*	Z	M^*	Z	M^*	Z
6x7	59.	2.	61.	2.	61.	2.
seq	2-1-5-6-3-4		6-1-3-5-4-2		6-1-3-4-5-2	
7x7	62.	2.	67.	2.	67.	2.
seq	6-7-1-5-3-2-4		6-1-3-5-4-2-7		7-6-1-3-4-5-2	
8x7	68.	2.	73.	2.	71.	2.
seq	1-2-3-6-8-7-5-4		6-8-1-3-5-4-2-7		1-8-3-5-7-6-2-4	
5x8	63.	1.	67.	2.	64.	1.
seq	4-3-5-1-2		5-3-1-4-2		4-5-3-1-2	
6x8	67.	1.	70.	1.	68.	2.
seq	6-5-4-3-2-1		6-5-3-1-4-2		4-5-6-3-1-2	
7x8	71.	3.	75.	2.	73.	2.
seq	6-5-4-3-7-1-2		7-6-5-3-1-4-2		7-4-5-6-3-1-2	
8x8	75.	2.	79.	2.	79.	3.
seq	6-5-4-8-7-3-1-2		7-6-5-3-1-8-4-2		6-4-7-5-3-2-1-8	

HEURÍSTICA 1 - HEURÍSTICA DE DANNEMBRING
HEURÍSTICA 2 - HEURÍSTICA CDS

As tabelas 4.15 e 4.16a e 4.16b mostram as comparações entre as duas heurísticas em questão utilizando os dados dos tempos de processamentos das tabelas 4.13 e 4.14.

Os resultados contidos nestas duas últimas tabelas indicam a superioridade da heurística CDS sobre a de Dannembrig em quase todos os casos. Assim, de acordo com estes resultados, notou-se a superioridade do modo FIS utilizando a mesma sequência dada pela técnica BAB sobre as sequências dadas pelas heurísticas de Dannembring e CDS. A utilização de uma heurística nestes casos trouxe resultados com tempos de execução inferiores do que com a utilização da técnica BAB embora com valores de makespans geralmente maiores.

CAPÍTULO 5

Conclusão

Neste trabalho teve-se como objetivo o estudo dos conflitos decorrentes do uso compartilhado da armazenagem intermediária em estruturas de processamento multi-estágios com fluxo unidirecional. Como conseqüencia deste estudo, propôs-se que o sequenciamento de tarefas nestas estruturas seja feita usando uma estrutura de busca em árvore, tipo "branch-and-bound", sem que a parcela de estimativa contemple o custo adicional que possa existir em função da limitação da armazenagem intermediária.

A proposta de sequenciamento contida neste trabalho, deriva das seguintes observasões:

- De uma maneira geral, o tempo total de execução das tarefas (makespan) depende da política de armazenagem, mas existem casos em que a ausência de armazenagem ou a disponibilidade de um número "infinito" de tanques que conduzem a resultados, se não semelhantes, bastante próximos. Esta situação parece indicar que os esforços computacionais adicionais para incorporar a estimativa de aumento destes tempos na função de custos do "branch-and-bound" podem ter pouco impacto na redução do número de nós sondados.

- A idéia de usar uma estratégia de enumeração implícita para o sequenciamento de tarefas permite mostrar, a cada passo, o perfil atualizado de uso da armazenagem intermediária, abrindo caminho para uma armazenagem interativa com o usuário. A estratégia alternativa de se usar uma abordagem MILP, fornece a cada passo uma solução relaxada do

problema que pode exigir uma factibilização, podendo gerar uma solução local sub-ótima.

- A abordagem Branch-and-Bound usa implicitamente uma estratégia de alocação do tipo "propriedade do produto" derivada da própria fórmula de recorrência para a determinação de C_{ij}. Esta estratégia completa, com os recursos disponíveis, as tarefas na ordem em que são lançadas na linha de processamento, o que normalmente reduz os períodos que uma tarefa deve ficar suspensa, o que acaba induzindo ociosidades. No caso de seqüencias de permutação a suspensão de uma tarefa de posição i na seqüencia no processador j para permetir o processamento de uma tarefa de posição k, com $i<k$, irá sempre induzir períodos ociosos que provocam um aumento do "makespan", (Baker, 1975).

- A armazenagem intermediária não constitui um recurso compartilhado com um perfil de demanda conhecido a priori. A demanda é gerada durante a execução das tarefas. Esta característica faz com que, dependendo das tarefas a serem processadas, o número de tanques necessários para a sua execução seja maior ou menor que o disponível. Isto significa que não há uma dependência explícita entre otimalidade do "makespan" e número de tanques de armazenagem. Em outras palavras, o "número de tanques de armazenagem necessários", não configura uma medida regular de desempenho do sistema. Por definição, medida regular de desempenho é uma medida que depende do tempo de término de cada tarefa no sistema (C_{iM}, tempo de término da tarefa i no último processador). Se C_{iM} cresce, então o desenpenho do

sistema diminui. Esta característica garante a monotonicidade da função de custo em uma abordagem BAB. A monotonicidade é importante pois garante que o custo das soluções parciais nunca diminua, e que portanto a solução final encontrada seja ótima.

- O custo de utilização da armazenagem intermediária parece então não poder ser explicado apenas em termos de aumento de tempo de execução das tarefas, mas pode ter um custo monetário. de fato existem situações em que o período em que uma tarefa deve ser retirada em um tanque pode ser inferior à soma dos tempos necessários para transferir o produto do processador $(j-1)$ para o armazém e do armazém para o processador j. Isto implica em custos que podem ser considerados "mistos", isto é, existe um custo monetário (funcionamento de bombas por exemplo) e temporal (aumento de C_{IM}).

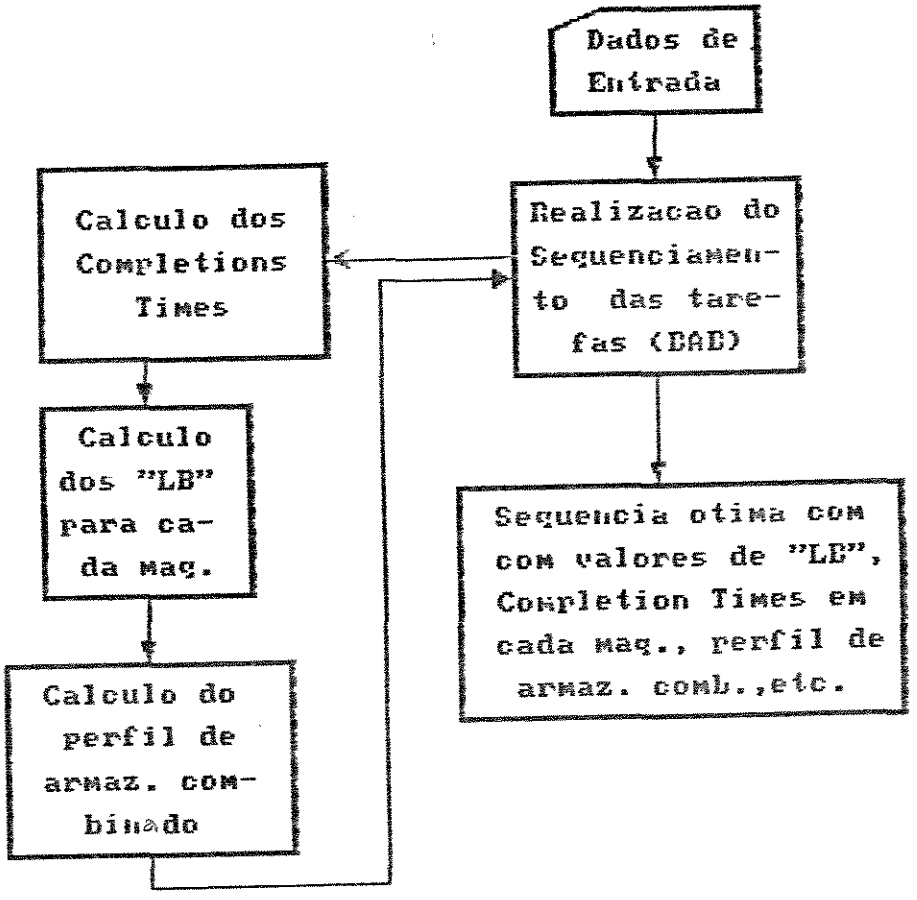
-O problema da armazenagem intermediária deve ser então estudados em estruturas de processamento mais complexas, bem como na presença de outras restrições. De fato, a presença de outras restrições tais como limitação de utilidades, mão-de-obra, etc; parecem exigir o recurso mais intensivo à armazenagem intermediária. Nestas condições, a armazenagem intermediária possa talvez ter evidenciado seu benefício ao permitir a maximização do uso dos recursos compartilhados.

Anexo

A.1 – Estrutura Geral do Programa

A.1.1 – Perfil de Armazenagem Combinado (modo UIS)

Um dos programas desenvolvidos neste trabalho realiza o sequenciamento das tarefas a serem executadas em todas as máquinas utilizando o método " Branch And Bound " (BAB) descrito com maiores detalhes na seção 3.1. Para cada nó representando uma sequência parcial que está sendo expandida é realizado uma avalização dos valores dos limitantes inferiores correspondentes, os quais são calculados com a utilização da heurística FMBB (seção 2.5.1.2). Através das comparações entre os valores destes limitantes é montado uma lista em ordem crescente dos nós sondados. Quando o primeiro da lista tiver todas as tarefas sequenciadas significa que aquela sequência é a melhor. Assim este programa oferece um perfil de armazenagem combinado, ou seja, para um determinado exemplo, ele sugere a quantidade de tanques de armazenagem necessários no caso de disponibilidade não limitada, ou seja no modo UIS. Sempre que ocorrer necessidade de armazenagem, haverá alocação da tarefa para a mesma. Um diagrama de blocos sumarizando a estrutura geral deste Programa pode ser visto na FIG. A-1.



FI. A1 - Diagrama de blocos sumarizando o algoritmo que dá o perfil de armazenagem combinado.

A.1.1.1 – Estrutura detalhada do algoritmo que fornece o perfil de armazenagem combinado para o modo UIS

O sequenciamento neste algoritmo envolve as seguintes etapas:

1. Realização do sequenciamento das tarefas no primeiro Nível :

1.1 - É criado uma nova sequência de duas tarefas no primeiro nível, a partir do primeiro nó inicial (nó-pai), que representa a primeira tarefa na sequência, colocando uma nova tarefa após esta.

1.2 - Através de dados tais como : o vetor que representa esta nova sequência (ISP), o vetor de tarefas não sequenciadas (INS), número de tarefas sequenciadas (NTS) ou não sequenciadas (NTNS), calcula-se os valores dos limitantes inferiores (LB) e dos instantes de término das tarefas (Completion Times).

1.3 - Nesta etapa é colocado este vetor sequenciado em uma lista e em outra, o valor do LB gerado.

1.4 - É repetido o item 1.1 para o sequenciamento do segundo nó-pai, repete-se também o item 1.2 para esta nova sequência e no item 1.3 é colocado esta nova sequência em posição adequada de acordo com o valor de seu LB gerado. Os valores dos LBs são colocados em ordem crescente.

1.5 - É repetido o processo até que seja tomado o último nó-pai. Neste ponto, existe uma lista com todas as

sequências parciais do primeiro nível e os seus correspondentes LBs em ordem crescente.

2. Escolha do nó candidato :

2.1 - Conforme a lista criada no item 1.3 , a primeira posição da mesma contém a sequência parcial com menor valor de LB entre todas as sequências do primeiro nível. Portanto, esta sequência parcial que será o nó escolhido como candidato para uma futura expansão, será ramificado e seus nós-filhos serão sondados tal como foi feito no primeiro nível. Esta primeira posição, de onde saiu o nó candidato, será ocupada, agora, pela sequência parcial, que anteriormente era pertencente a segunda posição e daí por diante.

3. É realizado o sequenciamento dos outros níveis:

3.1 - É calculado entre outras coisas os valores dos LBs para esta nova sequência escolhida.

3.2 - Este valor de LB, pertencente a esta sequência é comparado com os outros valores contidos na lista discutida anteriormente, e assim, este valor e a sequência parcial correspondente são colocados na posição adequada.

3.3 - Este processo é repetido até que a sequência pertencente a primeira posição na lista contenha todas as tarefas sequenciadss, ou seja, até que o número de tarefas sequenciadas (NTS) seja igual ao número de tarefas a serem sequenciadas (NTNS) neste Flowshop.

Para cada nova sequência formada é calculado o instante de término das tarefas. É lógico que é necessário que sejam fornecidos alguns dados tais como : número de tarefas e

máquinas, os tempos de processamentos e de transferência de cada tarefa em cada máquina, além dos tempos de estabelecimentos, que é o tempo requerido para começar a processar uma tarefa após a anterior ser completada.

Neste algoritmo, todas as vezes que ocorre a necessidade de armazenagem para alguma sequência em questão, em um determinado número de máquinas é calculado a quantidade de tanques necessários. Além disso, são fornecidos, para a sequência em questão os tempos que ocorrem as demandas. Assim, tem-se o número de tanques de armazenagem para o caso de armazenagem intermediário não limitada (UIS). Então é fornecido um perfil combinado de armazenagem intermediária, ou seja, esta armazenagem não está localizada entre as unidades de processamentos. Existem tanques de armazenagem suficientes para receber quantas tarefas forem necessárias.

A.1.2 — Modo FIS

Neste caso ocorre a limitação de um recurso: armazenagem intermediária. Foi desenvolvido, então em uma fase a seguir um programa que é capaz de limitar o número de tanques em um determinado flowshop. Neste algoritmo, as etapas envolvidas no sequenciamento e cálculo do instante de término das tarefas foram desenvolvidas como no caso UIS. Uma vez conhecendo à priori, o número de tanques necessários para que o flowshop em questão se comporte como um UIS, é desenvolvido este novo

algoritmo que avalia para cada sequência parcial sondada esta nova restrição, ou seja, para cada problema existe um número de tanques existentes, que podem ser utilizados. São feitas, então modificações na programação para acomodar as restrições de limitações de tanques disponíveis. Isto significa que dependendo do instante de término de uma determinada tarefa e da não disponibilidade de tanque(s) de armazenagem, deve haver uma modificação da programação corrente através da introdução de um atraso do instante de início do processamento ou a utilização do processador como tanque de armazenagem. O limitante inferior (LB), é então recalculado neste novo cenário e terá a sua posição adequada na lista de sequências. Da messma forma será escolhida no final aquela sequência completa que tiver o menor valor de "makespan".

Este algoritmo que dá o perfil FIS para um determinado flowshop foi desenvolvido de duas formas diferentes de acordo com as estratégias de alocações discutidas anteriormente: quando há necessidade de tanque(s) de armazenagem e este(s)não é(são) disponível(eis), pode ocorrer:

1) Atraso do ínicio do processamento da devida tarefa (ZW/FIS) ou;

2) Utilização da própria unidade de processamento como tanque(s) de armazenagem (NIS/FIS) sob a estratégia de prioridade do evento.

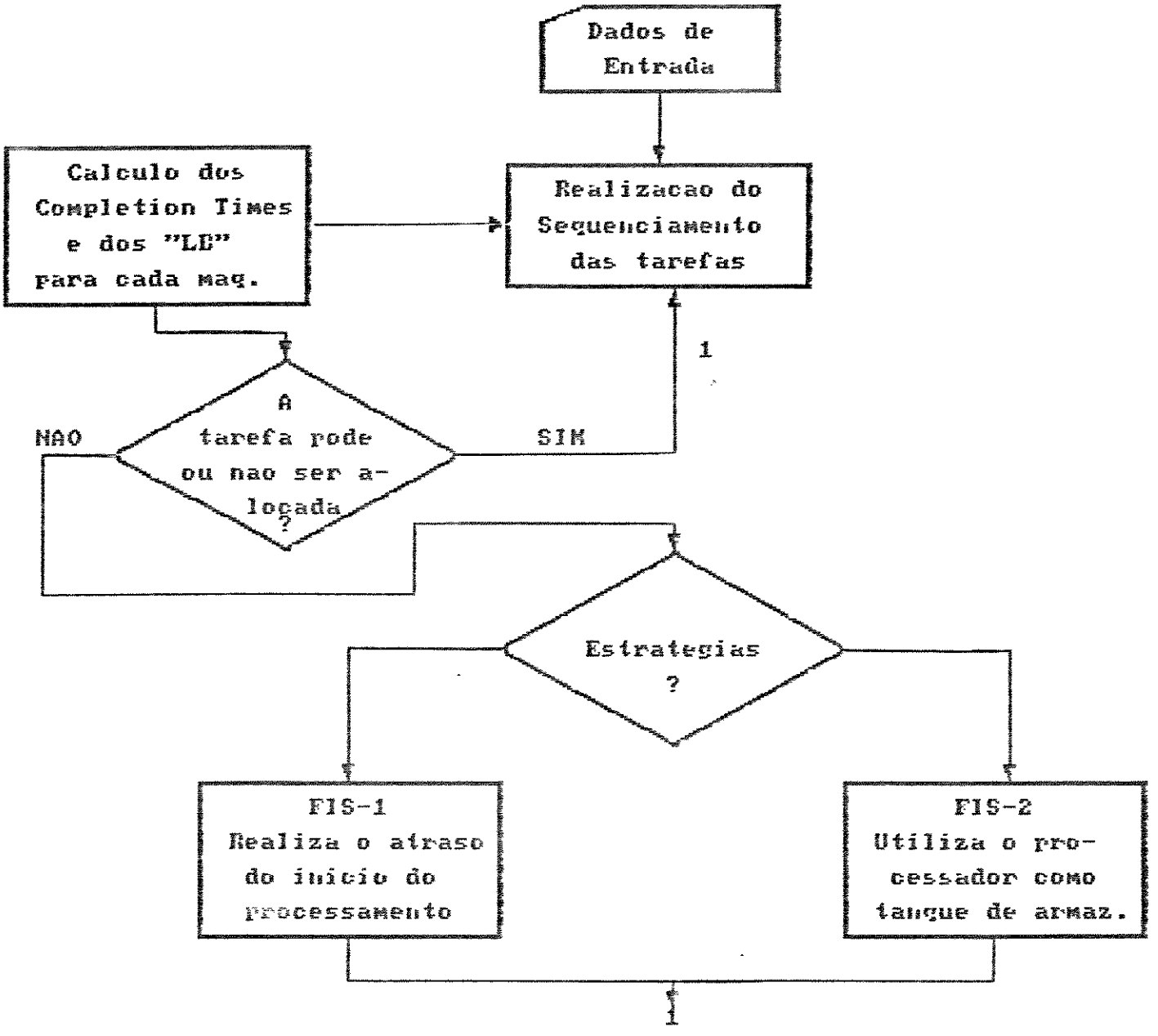


FIG. A.2 - Diagrama de blocos do algoritmo que trata o problema de armazenagem limitada (FIS)

Referencias Bibliográficas

Baker,K.R., "Introduction to Sequencing and Scheduling", John Willey, 1974.

Campbell,D.G.,Dudek,R.A e Smith, " A heuristic algoritm for the n job machine sequencing problem", Mgmt Sci, 16, pp B-630-637, 1970.

Dannenbring,D.G.,"An evaluation of flowshop sequencing heuristics, Mgmt. Sci. 23, pp 1174, 1977.

Das,H.,Cummings,P.T. e Le Van,M.D., "Scheduling of serial multiproduct batch processes via simulated annealing", 14(12), pp 1351-1362,1990.

Dolan,W.B.,Cummings,P.T., e LeVan,M.D., "Process Optimization via Simulated Annealing : Application to Network Design, AIche Journal, 35(5), pp 725-736, 1989.

Dutta,S.K e Cunningham,A.A," Sequencing two machine flowshops with finite intermediate storage", Mgmt. Sci, 21, pp 989-996, 1975.

Gilmore,R.C. e Golmory,R.E., " Sequencing a one-state variable machine: a solvable case of the traveling salesman problem", Operations Research, 12, pp 655-679, 1964.

Gonzales,M.J.," Deterministic processor scheduling", Assoc. Comput. Mach Surveys. 9, pp 173-204, 1976.

Graham,R.L., Lawer,E.L., Lenstra,J.K. e Rinnooykan,A.H.G., " Optimization and Approximation in Deterministic Sequencing and Scheduling : A Survey", Ann. Discrete Math.5, pp 287-326, 1979.

Gupta,J.N.B,"Optimal flowshop schedules with no intermediare storage space", Nav. Res. Logist. Q., 23, pp 235-243, 1976.

Knopf,F.C.," Sequencing a generalized two-stage flowshop with finite intermediare storage", Computers. Chem. Engng, pp 208-221, 1984.

Ku, H., Karimi,I., " Scheduling in Serial multiproduct Batch Processes with Finite Interstage Storage: A mixed Integer Linear Program Formulation", Ind. Eng. Chem., 27, pp 1840-1848,1988.

Ku, H., Karimi,I., " Completion Time Algorithms for Serial Multiproduct Batch Processes with Shared Storage",1990,*.

Leisten,R.," Flowshop sequencing problems with limited buffer storage", Int. J. Prod. Res., 28,(11),pp 2085-2100, 1990.

Levner,E.V.,,"Optimal plannining of parts'machining on a number of machines",Automn Remote Control, 12, pp 1972-1978, 1969.

Mcmahon,G.B, Burton,P.G.,"Flowshop scheduling with the branch and bound method", Ops. Res. 15, pp 473-481, 1967.

Metropolis,N.,Rosenblu,A.,Rosenbluth,A.T. e Teller,E., "Equation of State Calculations by Fast computing Machines", J. Chem. Phys., 21, pp 1087,1953.

Panwaker,S.S e Woollan,C.R.,"Flowshop scheduling problem with no inprocess waiting : A special case", J. Opl. Res. Soc., 30, pp 661-664, 1979.

Panwaker,S.S. e Iskander,W.," A survey of scheduling rules", Ops. Res., 25, pp 45-61, 1977.

Papadimitriou,C.H. e Kanellakis,P.C., " Flowshop scheduling with limited temporary storage",Journal of the Association for Computing Machinery, 27, pp 533-554, 1980.

Rajagopalan,D e Karimi,J.A.," Completion Times in Serial Mixed-Storage Multiproduct Processes with Transfer and Set-Up Times, Comp. Chem. Engng , 1/2(13), pp 175-186, 1989.

Reklaitis,G.V.,"Review of Scheduling of Process Operations", AIche Symposium Series, 78, pp 119-133, 1982.

Rippin,D.W.T. e Hofmeister,M.,"The flowshop scheduling problem problem with no intermediare storage", AIche Symposium Series,1981.

Rodrigues,M.T.M.,"Sequenciamento e alocação de operações em flowshops na indústria química com restrições sobre recursos compartilhados: uma abordagem de busca orientada pro restrições",FEE, 1992.

Suhami,I e Mah,R.S.H.,"An implicit enumeration scheme for the flowshop problem with no intermediare storage, Comput. Chem. Engng, 5, pp 83-91, 1981.

Wiede,W.,Kuryan,K. e Reklaitis,G.V.,"Determination of Completion Times for Serial Multiproduct Processes - 1. A two unit finite intermediate storage system." Comput . Chem. Engng., 11, pp 337-344, 1987.

Wiede,W.,Kuryan,K. e Reklaitis,G.V.,"Determination of Completion Times for Serial Multiproducts Processes - 2. A multiunit finite

intermediate storage system", Comput. Chem. Engng., 11, pp 345-356, 1987.

Wiede,W.,Kuryan,K. e **Reklaitis,G.V.**,"Determination of Completion Times for Serial Multiproduct Processes - 3. Mixed intermediate storage systems". Comput. Chem. Engng., 1987, 11, pp 357-367, 1987.

Wismer,D.A.," Solution of flowshop scheduling problems with no intermediare queues",Ops. Res. 20, pp 689-697, 1972.