UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA QUÍMICA
ÁREA DE CONCENTRAÇÃO - SISTEMAS DE PROCESSOS QUÍMICOS E INFORMÁTICA

"ESTRATÉGIA DE SIMULAÇÃO PARA A ALOCAÇÃO DE PRODUTOS EM PLANTAS MULTIPROPÓSITO"

Autora: Ana Claudia Gondim de Medeiros
Orientadora: Profª. Drª. Maria Teresa Moreira Rodrigues

Tese apresentada à Faculdade de Engenharia Química como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Química

Abril de 1995
Campinas - SP

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

M467e

Medeiros, Ana Claudia Gondim de
Estratégia de simulação para a alocação de produtos em plantas multipropósito / Ana Claudia Gondim de Medeiros.--Campinas, SP: [s.n.], 1995.

Orientadora: Maria Teresa Moreira Rodrigues
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Química.

1. Pesquisa operacional. 2. Controle de produção. 3. Produção - Sequenciamento 4. Indústria química. 5. Indústria - Automação. I. Rodrigues, Maria Teresa Moreira. II. Universidade Estadual de Campinas. Faculdade de Engenharia Química. III. Título.

Tese defendida e aprovada em 18 de abril de 1995, pela banca examinadora constituida pelos professores:

Profª. Drª. Maria Teresa Moreira Rodrigues

Prof.Dr.Luis Gimeno Latre

Prof. Dr. Mario de Jesus Mendes

AGRADECIMENTOS

À minha mãe Jandira, ao meu pai Glauco, à minha irmã Carla e a Daniel pelo carinho, apoio e incentivo em todas as horas. A eles dedico este trabalho.

À prof. Dra. Maria Teresa Rodrigues, pela orientação e por tudo que me ensinou.

Ao prof. Dr. Luís Gimeno Latre, pelas valiosas sugestões

A Márcio Dutra, pela colaboração e sugestões no trabalho árduo de muitas manhãs.

À CAPES, pelo apoio financeiro.

A todos que contribuíram para a realização deste trabalho, em especial à Andreia, pelo apoio técnico nos micros e workstations.

Aos meus amigos, que estiveram comigo em todos os momentos.

"...quem sabe o fim não seja nada
e a estrada seja tudo..."
Marina

SUMÁRIO

RESUMO

Com o desenvolvimento recente das indústrias de especialidades químicas e com a crescente busca de melhoria na qualidade e aumento da produtividade das plantas industriais, o problema de elaboração de planos de produção eficientes em plantas químicas flexíveis tem crescido de importância ao longo dos últimos anos. Nestas plantas é possível se processar um grande número de produtos que compartilham os mesmos equipamentos e recursos disponíveis. Os planos de produção visam tirar o máximo proveito na utilização destes recursos, ao mesmo tempo em que se busca atender as exigências impostas pelo mercado, estabelecendo a ordem em que os produtos devem ser produzidos em cada equipamento e a alocação temporal destes produtos na unidade produtiva, respeitando-se as restrições na oferta de recursos comuns com um custo mínimo.

Por sua natureza combinatorial, este tipo de problema apresenta grande complexidade computacional que aumenta exponencialmente quanto maior for sua dimensão (número de equipamentos disponíveis X número de produtos que se deseja produzir). Além disto, a limitação na oferta de recursos comuns, se constitui em um outro fator que também interfere grandemente na complexidade do problema. Em vista disto, as estratégias desenvolvidas para sua resolução normalmente assumem algumas hipóteses simplificadoras, através do relaxamento de algumas restrições, com o objetivo de diminuir esta complexidade.

Neste trabalho, será apresentado um estudo efetuado sobre um problema bastante restrito proposto por U.M.Egli e D.W.Rippin, onde os autores sugerem um procedimento heurístico para obter um plano de produção para quatro produtos com restrições de oferta de recursos compartilhados e outras restrições, a fim de minimizar custo total de produção.

Por último, será apresentado um **algoritmo de simulação** desenvolvido a partir dos dados do problema do Rippin, que tem como objetivo calcular o instante mais cedo em que as operações não alocadas que compõem cada batelada de cada produto podem ser iniciadas, a partir de uma alocação efetuada. A abordagem utilizada divide o problema em dois níveis: no primeiro são geradas as janelas de tempo baseadas na disponibilidade de matérias-primas e venda de produtos, e no segundo é efetuada a alocação das etapas de produção sem que haja relaxamento das restrições envolvidas no problema.

1- INTRODUÇÃO

No início da indústria de processamento químico, quando o conhecimento tecnológico acerca dos processos era ainda incipiente, as plantas químicas operavam com modo de produção descontínuo ou batelada, que permitiam mais facilmente a interferência humana. Com o desenvolvimento de pesquisas sobre estes processos, impulsionado pelo aumento de demanda do mercado mundial dos produtos químicos, o modo de produção passou a ser contínuo que, em geral, apresenta maior produtividade e eficiência que as indústrias batelada.

Entretanto, com o surgimento de especialidades químicas nos anos recentes, em especial da indústria de química fina e farmacêutica, as plantas químicas bateladas voltaram a ter importância neste cenário. Estas plantas possuem como principal característica uma grande flexibilidade, onde muitos produtos de diversos tipos, podem ser produzidos em um mesmo conjunto de equipamentos. Assim, os produtos que se deseja produzir freqüentemente compartilham, além dos equipamentos, outros recursos tais como energia elétrica, vapor e matéria-prima.

Neste contexto, observa-se que esta flexibilidade estabelece um elevado número de alternativas e variantes de produção, introduzindo a necessidade da elaboração de um plano de produção adequado a fim de que se possa alocar os recursos disponíveis (tempo, processadores, mão-de-obra, matérias-primas, utilidades, etc.) de maneira que sejam atendidas todas as exigências do mercado ao mesmo tempo em que se tira o máximo proveito dos recursos disponíveis na planta.

O objetivo do plano de produção é estabelecer a ordem em que os produtos devem ser produzidos em cada equipamento e a alocação temporal destes produtos na unidade produtiva, definindo-se quando e em que equipamentos cada uma das operações envolvidas no seu processamento devem ser realizadas, de forma a cumprir a previsão de demanda respeitando-se as restrições de recursos comuns com um custo mínimo.

Neste trabalho serão apresentados, primeiramente, os principais fatores que influenciam o problema de programação de produção. Dentre estes fatores destacam-se a estrutura de processamento do shop e a oferta limitada de recursos compartilhados que interferem no valor da função custo e no grau de complexidade que o problema apresenta. Serão apresentadas também estratégias utilizadas para a resolução do problema e os principais critérios de desempenho empregados.

Em seguida será apresentado um estudo efetuado sobre um problema proposto por U.M.Egli e D.W.Rippin (1986) onde deseja-se obter um plano de produção para quatro produtos com restrições de oferta de recursos compartilhados e outras restrições.

O problema de Rippin é um problema clássico citado na literatura onde os autores sugerem um procedimento heurístico para a resolução do problema e cujo critério de desempenho escolhido é o custo total de produção representado, principalmente, pelo custo de estocagem final. Assim sendo, o procedimento proposto pelos autores decide pelo processamento das bateladas dos produtos que se deseja processar o mais tarde possível a fim de minimizar o custo com estoques, ao mesmo tempo em que se observa as restrições de recursos comuns e demais restrições presentes no problema.

Por último, será apresentado um algoritmo de simulação que, utilizando os dados do problema do Rippin com todas as restrições a ele impostas, define quais operações são candidatas a alocação, e calcula o instante mais cedo em que estas operações não alocadas podem ser iniciadas, a partir de uma dada condição da planta. Este procedimento visa a minimização do tempo total de produção **(makespan)** como o critério de desempenho a ser otimizado. A estratégia de alocação dos produtos é dividida em dois níveis:

Nível 1: geração de janelas de tempo baseadas na disponibilidade de matérias-primas e venda de produtos

Nível 2: alocação das etapas de produção satisfazendo as restrições.

Um algoritmo de simulação é um instrumento que permite levar em consideração todas as restrições envolvidas no problema, sem relaxamento. Neste tipo de procedimento, a tomada de decisão acerca de qual a próxima operação que será alocada é efetuada pelo usuário, e não executada automaticamente pelo algoritmo. Assim, cada vez que uma nova operação é alocada, o simulador recalcula os instantes em que as demais operações que ainda não foram alocadas podem ser iniciadas. Cabe então ao usuário escolher qual a próxima operação que integrará a seqüência. Desta forma, pode-se testar soluções que foram obtidas por algoritmos de decisão automática, que em geral são algoritmos relaxados, mas levando-se em conta todas as restrições presentes no problema, ou ainda se testar qualquer seqüência que se deseje verificar o custo.

2-ASPECTOS RELEVANTES DO PROBLEMA DE PROGRAMAÇÃO DE PRODUÇÃO

2.1- INTRODUÇÃO

Este capítulo tem como objetivo descrever o cenário onde se situa o problema de seqüenciamento/scheduling na indústria química descontínua e apresentar os principais fatores que o influenciam, indicando as estruturas de processamento possíveis em unidades deste tipo e classificando-as segundo o seu fluxo de processamento e o número de processadores envolvidos no processo produtivo. Serão apresentadas também as políticas de armazenagem intermediária empregadas e sua influência na solução do problema. Em seguida será mostrado de que forma a oferta limitada de recursos compartilhados pode modificar o enfoque de sua resolução. Por último serão apresentados algumas características que são importantes na classificação do problema, bem como as funções de desempenho mais utilizadas e as estratégias de solução envolvidas na resolução do problema de seqüenciamento/scheduling em unidades industriais descontínuas.

2.2- APRESENTAÇÃO DO PROBLEMA DE PROGRAMAÇÃO DE PRODUÇÃO

Ao longo dos anos, a indústria química passou por grandes transformações que vem ocorrendo desde os seus primórdios, quando a forma de produção neste ramo de atividade ainda era bastante incipiente. Neste contexto, o processamento na indústria química era realizado em unidades descontínuas que favoreciam a interferência humana para maior controle do processos. Com o incremento de estudos e pesquisas na área, estimulados pela maior procura de produtos químicos no mercado mundial, os processos descontínuos foram progressivamente sendo substituídos por processos contínuos que apresentam maior eficiência e produtividade, mas que também requerem maior domínio e conhecimento sobre o processo. Em vista disto, ao longo dos últimos anos as operações contínuas vem sendo o modo de produção mais executado em indústrias de processamento químico.

Recentemente, entretanto, tem-se observado uma retomada na procura por processos descontínuos (ou batelada). Um estudo sobre estes processos feito por Parakram revelou que apenas 6% dos processos descontínuos identificados tem similares contínuos (Ku,H.M. et al (1987)). O crescimento do mercado de produtos de química fina tais como polímeros, corantes, bioquímica, herbicidas, cosméticos, gêneros alimentício e indústria farmacêutica tem favorecido sobremaneira o ressurgimento de indústrias não contínuas. Tais produtos utilizam-se deste modo de operação para sua produção.

A principal característica destas especialidades químicas é que, em geral, são produzidas em pequenas quantidades possuindo alto valor agregado do produto final e apresentando, desta forma, alta lucratividade. Em virtude da baixa escala produtiva, não é economicamente conveniente se estabelecer uma planta separada com equipamentos caros para cada produto que se deseja processar. Opta-se, então, por indústrias com alto grau de flexibilidade onde pode-se acomodar produtos similares, estabelecendo um grande número de alternativas e variantes de produção, de forma que se possa utilizar os mesmos equipamentos e recursos disponíveis.

Unidades industriais com as características descritas acima devem ser capazes de absorver as incertezas nas demandas do mercado atual, que vem se tornando cada vez mais competitivo, ao mesmo tempo em que mantém a capacidade e a qualidade de produção para uma larga faixa de produtos comercializados, além de aceitar novos produtos sem investimentos significativos.

Por esta razão, observa-se que esta mesma flexibilidade, embora desejável, também traz problemas adicionais no que se refere ao gerenciamento de produção, na medida em que se torna imprescindível a elaboração de um plano de produção global que seja ótimo no sentido de que sejam atendidas todas as exigências do mercado ao mesmo tempo em que se tira o máximo proveito dos recursos disponíveis na planta.

Por plano de produção entende-se a ordem em que cada produto deve ser produzido e a alocação temporal destes produtos na unidade produtiva, definindo-se quando e em que equipamentos cada uma das operações envolvidas no seu processamento devem ser realizadas, de forma a cumprir a previsão de demanda respeitando-se as restrições de recursos comuns com um custo mínimo.

Como tempo e equipamentos são compartilhados entre os diversos produtos, obter um plano de produção ótimo significa compatibilizar fatores tais como oferta limitada de matérias-primas e utilidades, inventário de produtos intermediários, disponibilidade ou não de

estocagem intermediária, disponibilidade de mão-de-obra, além das variações constantes na demanda de mercado. Isto gera a necessidade de se elaborar planos de produção flexíveis que sejam capazes de acomodar estes perfis de produção variáveis no tempo.

Quando estão disponíveis previsões de demanda bem estabelecidas para um longo período de tempo, torna-se relativamente fácil decidir a priori que produtos serão produzidos e quando isto ocorrerá de maneira que se obtenha aproveitamento total desta flexibilidade. Porém se esta previsão de demanda não está disponível e a planta é submetida a um regime de operação que visa o atendimento de pedidos a curto prazo o problema torna-se bastante complexo. Além disto, o gerenciamento de produção deve-se confrontar em algumas ocasiões com problemas imprevistos tais como indisponibilidade temporária e manutenção de equipamentos que obriga a modificar o que anteriormente estava previsto pelo planejamento da produção (Kondili et al (1993)). O problema de gerenciamento de produção não possui solução simples pois envolve inúmeros fatores que devem ser atendidos simultaneamente de forma a se obter o resultado pretendido.

São citados na literatura dois tipos de abordagens para resolução do problema: a abordagem nível único e a abordagem hierárquica.

Na abordagem de nível único todos os aspectos envolvidos no planejamento e programação de produção são vistos simultaneamente. O objetivo é analisar e determinar a quantidade de produtos que devem ser produzidos, os recursos requeridos, os equipamentos nos quais cada operação deve ser executada, a seqüência de produção em cada período de tempo e a política de estocagem intermediária utilizada a fim de otimizar um critério de desempenho estabelecido.

Na abordagem hierárquica, o problema global é subdividido em pelo menos dois níveis: o problema de planejamento (longo prazo) e o problema de programação (curto prazo).

O problema de longo prazo decide o número de bateladas de cada produto que devem ser produzidas para atender a demanda especificando os mesmos parâmetros definidos na abordagem de nível único. Neste nível, porém, não são tomadas decisões de seqüenciamento de produtos e não são necessárias informações temporais detalhadas do plano de produção.

No problema de curto prazo deseja-se determinar o instante em que cada produto/operação deve ser realizada de forma a otimizar o critério de desempenho preestabelecido objetivando tirar o máximo de proveito das instalações disponíveis. Conhecidos todos os parâmetros relativos ao processamento de cada produto como suas receitas, tempos de processamento, consumo de utilidades, além de um conjunto de fatores

que irão determinar a forma de produção tais como a natureza da armazenagem intermediária, a estrutura de processamento e os perfis temporais da oferta de recursos, parâmetros estes já especificados no nível de planejamento, o problema de programação da produção também denominado de **scheduling** preocupa-se em determinar a seqüência de produção, a alocação dos M equipamentos aos N produtos e o instante de início e fim de cada operação, definindo detalhes da produção diária da planta (Ku,H.M. et al.(1987), Rodrigues,M.T.(1992), Campos(1993)).

O problema é dito de **seqüenciamento** quando definida a ordem em que cada operação/terefa será executada em cada processador, através de um critério de desempenho previamente definido, o problema de otimização estará resolvido. Este critério de desempenho que definirá qual será a seqüência ótima de produção, em geral, não envolve tempos ou, se esta for uma variável presente na função, o tempo de início de cada tarefa/operação deverá assumir o valor máximo definido apenas entre os instantes em que o processador é liberado pela operação/tarefa anterior na seqüência de produção para a execução da próxima operação/tarefa, ou pelo instante em que a operação i de uma dada tarefa é concluída podendo ser iniciada a operação i + 1 desta mesma tarefa. Como o problema de seqüenciamento é de origem combinatorial, pois deseja-se alocar N tarefas nos M processadores disponíveis, dependendo da natureza do problema, podem existir N! seqüências possíveis. Pode-se observar, então, que o número de soluções cresce exponencialmente a medida que aumenta o número de produtos que se deseja processar.

No contexto de scheduling, em que existem restrições na oferta de recursos comuns ou limitações no prazo de entrega, determinar a ordem em que cada tarefa é executada nos processadores não é suficiente para se equacionar o problema adequadamente. Neste caso, o problema assume contornos de alocação em que a variável tempo tem que ser manipulada de forma que o instante de início obtido seja também factível com respeito as restrições presentes no problema para cada operação/tarefa a ser processada. Uma discussão a cerca das diferenças de abordagens entre os problemas que envolvem apenas seqüenciamento e os que envolvem scheduling será apresentada posteriormente.

Devido a esta elevada dimensão e complexidade, ferramentas computacionais sofisticadas para planejamento e scheduling de produção têm sido propostas na literatura, porém, mesmo para poderosos computadores atuais, encontrar uma solução ótima não é tarefa fácil visto que o tempo de processamento necessário para resolve-lo também aumenta exponencialmente com o aumento de sua dimensão.

Por estas dificuldades, os algoritmos hoje encontrados na literatura possuem aplicabilidade restrita a problemas de pequena dimensão ou problemas que não apresentem configurações complexas ou limitações em recursos compartilhados. Não existe hoje disponível uma ferramenta que considera automaticamente todas as possibilidades estruturais simultaneamente.

2-3- CARACTERÍSTICAS IMPORTANTES NO PROBLEMA DE PROGRAMAÇÃO DE CURTO PRAZO

2.3.1- ESTRUTURA DE PROCESSAMENTO

Uma indústria química descontínua pode ser classificada segundo sua estrutura de processamento em duas configurações principais: simples-estágio e multiestágios, com cada uma delas podendo ser subdivididas em duas outras subclasses, que serão apresentadas a seguir (Ku, H.M., Rajagopalan D. e Karimi, I. (1987)):

2.3.1.1- **Simples-estágio**:

Na estrutura simples-estágio, todas as tarefas processadas necessitam de apenas um estágio para a sua execução. Os dois tipos existentes desta estrutura de processamento são:

a) 1 estágio/1 processador

Neste caso, as tarefas necessitam de apenas um estágio para o seu processamento, e apenas um processador está disponível para a execução destas tarefas. A figura abaixo representa este tipo de processamento (Rodrigues,M.T. (1992)).

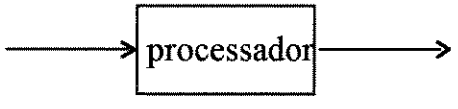


Figura 2.1: Estrutura 1 estágio/1 processador

b) 1 estágio/ M processadores

Para esta estrutura, as tarefas também necessitam de apenas um estágio, porém existem M processadores em paralelo disponíveis, que podem ou não apresentarem a mesma produtividade.

Este tipo de estrutura pode ser subdividida de acordo com sua forma de operação em preemptivo ou não preemptivo. No primeiro caso, a operação pode ser interrompida após seu início, para ser completada posteriormente no mesmo ou em outro processador, ou ter seu processamento dividido entre duas ou mais unidades. Na forma não preemptiva, esta interrupção não é permitida.

Quanto à taxa de processamento das unidades, ainda pode-se classificar esta estrutura em:

a) unidades idênticas

b) unidades uniformes

c) unidades não relacionadas

As unidades idênticas possuem a mesma taxa de processamento para todos os produtos. No caso de unidades uniformes, cada uma delas tem associada uma taxa de processamento diferente mas que é a mesma para todos os produtos. No tipo c, as unidades são dissimilares de forma que possuem diferentes taxa de processamento para diferentes produtos.

Entre todas as possibilidades relacionadas acima, as unidades idênticas ocorrem menos freqüentemente e o caso não preeemptivo é mais comum nos processos químicos industriais.

As figuras abaixo exemplificam as estruturas 1 estágio/ M processadores (Reklaitis (1982)).

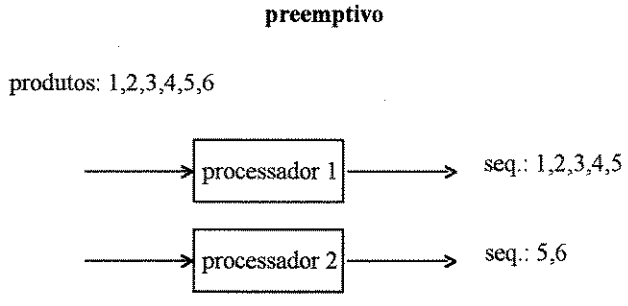


Figura 2.2: Estrutura 1 estágio/1 processador preemptivo

não preemptivo

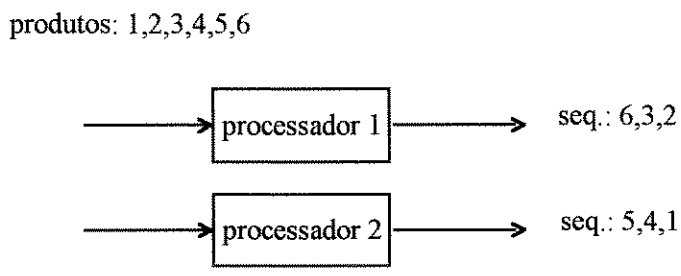


Figura 2.3: Estrutura 1 estágio/1 processador não preemptivo

2.3.1.2- Multiestágios

Neste tipo de processamento, N tarefas devem ser executadas em L estágios de processamento existindo M equipamentos disponíveis ($L \leq M$). As tarefas podem ou não utilizar todos os equipamentos. É o mais comumente utilizado nos processos químicos descontínuos sendo usado na indústria alimentícia, farmacêutica, de cosméticos, entre outras. A estrutura multiestágios pode ser subdividida em dois tipos de arranjo segundo seu fluxo de processamento: flowshop (multiestágios em série) e jobshop (multiestágios).

a) Flowshops - (Multiestágios em série)

Quando o fluxo de processamento de todas as tarefas que se deseja produzir é unidirecional, ou seja, o processador i sempre precede o processador i+1 para todas as tarefas do shop, este tipo de estrutura é denominada de estrutura multiproduto ou flowshop. Para cada produto, pode existir uma ou mais rotas de produção e cada estágio pode ser composto por mais de um processador que operam em paralelo.

O caso mais simples é aquele em que cada estágio é formado por apenas um processador caracterizando um flowshop puro ou estrito. Quando a seqüência de produção definida para a primeira máquina da planta é a mesma para as demais, esta seqüência é denominada seqüência de permutação. Neste caso existem N! seqüências possíveis. Esta estrutura é bastante utilizada em processos químicos industriais descontínuos especialmente quando os produtos que se deseja produzir pertencem à mesma família possuindo as mesmas rotas de produção, como no caso da industria de tintas. O esquema que representa o flowshop com seqüência de permutação é apresentado abaixo.

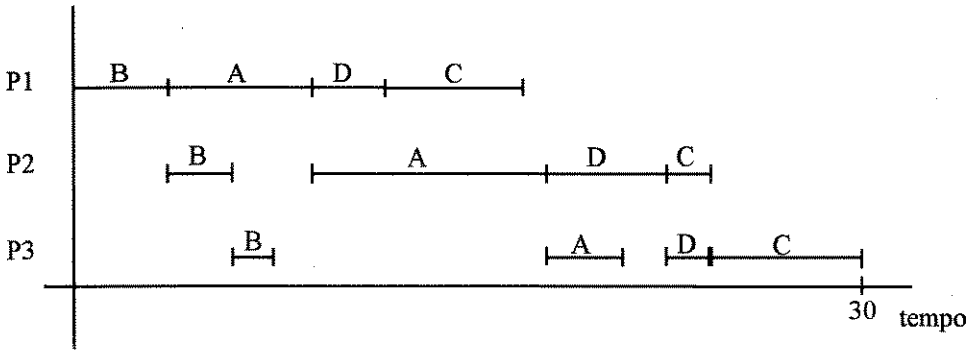


Figura 2.4: Flowshop com seqüência de permutação

Os instantes de fim (completion times) de cada tarefa que se deseja processar em cada um dos processadores quando ocorre seqüências de permutação sem limitações sobre outros recursos comuns são dados pela seguinte fórmula de recorrência:

$$C(i, j) = \text{máx} \{ C(i - 1, j); C(i, j - 1) \} + t(i, j) \qquad (2.1)$$

onde: $C(i, j)$ = instante em que a tarefa i é terminada no processador j

$C(i - 1, j)$ = instante em que a o processador j é liberado após o processamento da tarefa com posição i - 1 na seqüência

$C(i, j - 1)$ = instante em que a tarefa i está concluída no processador j - 1

$t(i, j)$ = tempo de processamento da tarefa i no processador j

No caso mais geral, não existe a imposição de que as seqüências de execução das operações sejam as mesmas em todos os processadores. Isto é possível quando existem rotas de produção de algumas tarefas com menos estágios que o número de processadores disponíveis. Neste caso, existe cerca de N! seqüências possíveis em cada processador, o que gera um conjunto de soluções da ordem de $N!^M$ (Rodrigues,M.T. (1992)). Este tipo de estrutura é denominada de flowshop generalizado.

No flowshop generalizado, existe a possibilidade de um estágio "n" interagir com outros estágios que não o prévio na rota de produção, "n-1", e o seguinte "n+1". Assim sendo, de acordo com a seqüência produzida em cada processador, é possível estabelecer duas formas em que esta interação pode ocorrer denominadas de by-pass e "crossing". Quando em um determinado shop estabelece-se o crossing, a seqüência para cada processador será diferente. Isto acontece principalmente quando os tempos de limpeza e preparação entre as operações

das tarefas não são os mesmos ou quando as tarefas possuem o mesmo fluxo na planta mas não possuem a mesma rota de produção, ou seja, não utilizam os mesmos processadores podendo existir mais processadores disponíveis que os necessários para a execução das tarefas. Cada tarefa utiliza, então, um subconjunto L de processadores onde $L \leq M$. Nestes casos, a formulação estabelecida para resolver problemas de seqüenciamento em flowshop com seqüência de permutação (Equação 2.1) não descreve os casos onde ocorre o crossing (Graells, M., Espuña, A., Puigjaner,L.(199)). Este tipo de estrutura será alvo deste trabalho. As figuras de 2.5 a 2.8 abaixo representam estes dois tipos de flowshop's.

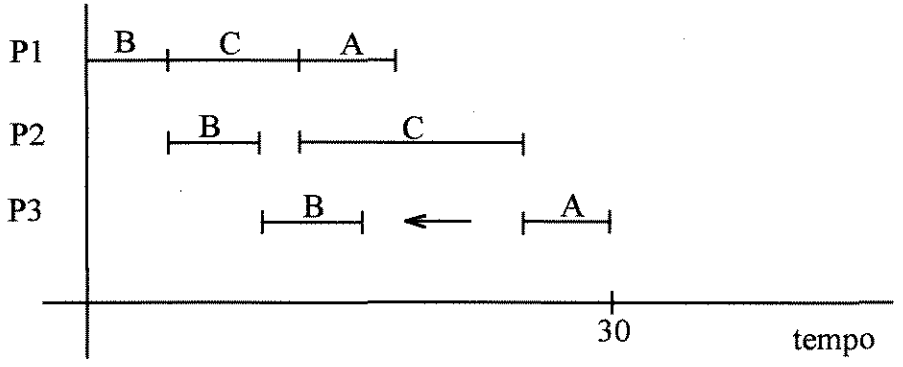


Figura 2.5: Flowshop sem by-pass

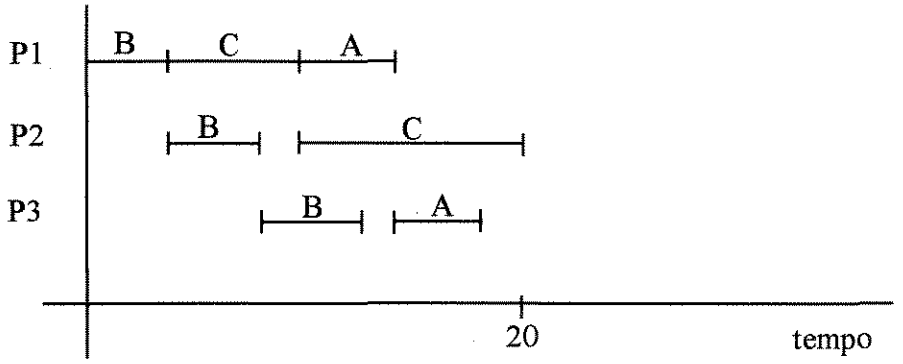


Figura 2.6: Flowshop com by-pass

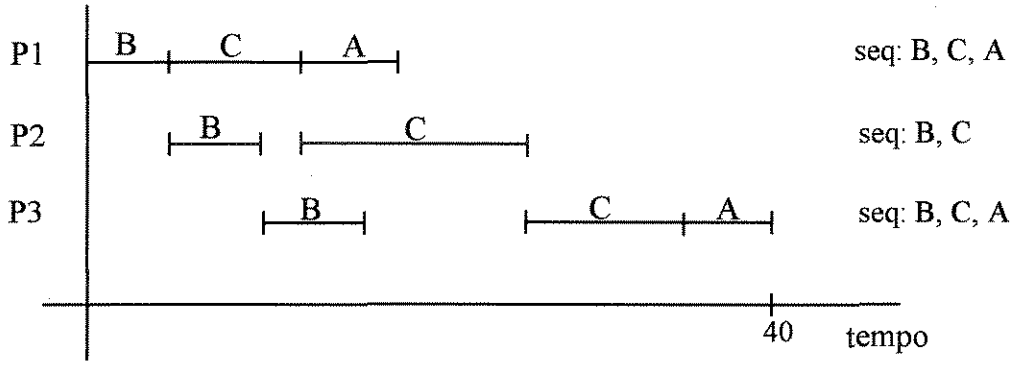


Figura 2.7: Flowshop sem crossing

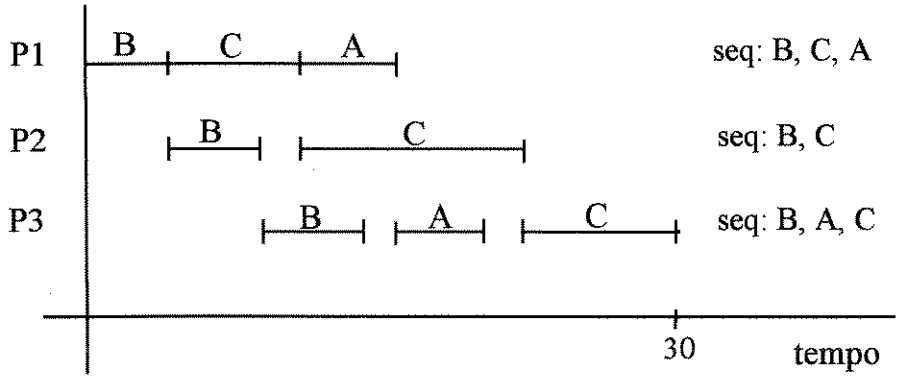


Figura 2.8: Flowshop com crossing

Observa-se que a não obrigatoriedade de existir a mesma seqüência de produção em todos os processadores pode acarretar em aumento de produtividade da planta, com melhor aproveitamento dos recursos disponíveis.

b) Jobshop

Em um jobshop, as N tarefas podem seguir diferentes rotas entre os M processadores não havendo obrigatoriedade do fluxo ser unidirecional. Este tipo de estrutura deve apresentar interligações fisicas entre os processadores bastante flexíveis o que possibilitará a operação das diferentes rotas de produção das tarefas que se deseja processar garantindo facilidade no transporte de material. Este problema é bastante difícil de ser resolvido quando se trata de uma indústria química (Rodrigues,M.T. (1992)). Em função disto, o uso de jobshop's é reduzido nestas indústrias sendo possível apenas quando N é pequeno, ou seja, quando se deseja processar poucas tarefas. A figura 9 representa a estrutura jobshop.

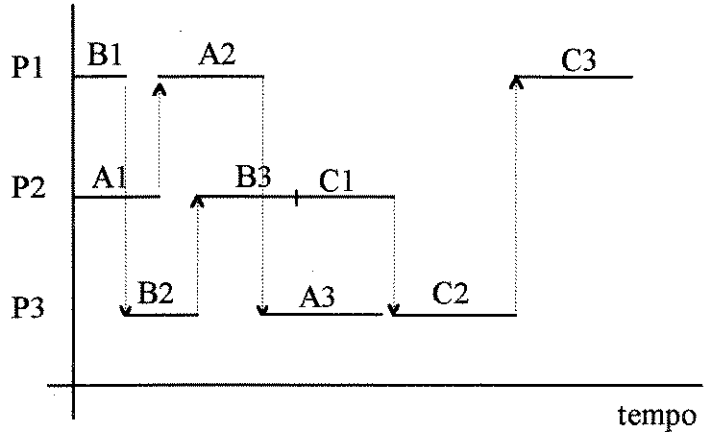


Figura 2.9: Estrutura Jobshop

2.3.2 - RECURSOS COMPARTILHADOS E OUTRAS RESTRIÇÕES

O problema de scheduling/seqüenciamento na industria química é, em última análise, um problema de compartilhamento de recursos, onde tarefas/operações devem ser processadas em um shop compartilhando tempo, equipamentos e demais recursos comuns, como matéria-prima, mão-de-obra, armazenagem intermediária e utilidades.

O objetivo, então, é resolver os conflitos decorrentes do compartilhamento destes recursos entre as tarefas de forma mais eficiente, a fim de que se possa obter uma solução otimizada respeitando todas as restrições presentes no problema.

Dentre os recursos citados, tempo e equipamentos são obviamente o de maior importância sendo que no problema de seqüenciamento/scheduling deseja-se conhecer em que instante de tempo cada tarefa/operação pode ser alocada em cada equipamento.

A armazenagem intermediária também é de grande importância na construção do scheduling, e a política de estocagem intermediária tem influência direta na alocação das tarefas/operações e no valor da função objetivo a ser otimizada. Devido a sua importância, a influência da armazenagem intermediária bem como os tipos de políticas comumente adotados serão descritos com mais detalhes no próximo item.

Utilidades e matéria-prima são recursos que em geral são disponíveis em quantidade limitada ao longo do horizonte de planejamento. Estes recursos influenciam grandemente na complexidade do problema, e, como será abordado adiante, existem poucos trabalhos na literatura que estabelecem uma relação de recorrência que seja capaz de calcular os instantes de início das tarefas/operações que levem em conta restrições deste tipo.

Os tempos de limpeza e preparação dos equipamentos (setup) após o término de uma operação e o início da seguinte também é uma restrição importante a ser observada na construção de um plano de produção. Em alguns tipos de processos estes tempos dependem da ordem em que as tarefas estão dispostas, ou seja, o tempo necessário para se preparar o processador que fez i para se iniciar k é diferente do tempo necessário para se preparar o processador que fez k para se iniciar i. Desta forma, os tempos de setup entre as operações têm uma grande influência sobre o critério de desempenho adotado para o sistema (Baker, K.R.(1974)).

2.3.3 - ARMAZENAGEM INTERMEDIÁRIA

Por armazenagem intermediária entende-se a estocagem de produtos intermediários entre os diversos estágios de processamento. Se depois de processado não for possível a transferência imediata do material para o próximo estágio de processamento, este intermediário pode ser transferido para um tanque, se houver, de forma que o equipamento que o processou seja liberado para a execução de uma outra operação, e onde deve aguardar até que o próximo estágio esteja disponível para o seu processamento.

Se o material processado em um determinado estágio for instável, a sua transferência para o tanque intermediário não é possível uma vez que, nestes casos, corre-se o risco de degradação deste material se ele tiver que aguardar a liberação do próximo equipamento para ser processado. Desta forma, o estágio que processa o material instável só pode ser iniciado no instante em que é assegurado que a transferência para o próximo estágio de processamento será efetuada imediatamente após o seu término

A importância da armazenagem intermediária reside no fato de que, quando ela está disponível, aumenta a produtividade do shop em função da liberação do processador para outras tarefas quando o próximo estiver ocupado, facilita os procedimentos de limpeza e preparação e permite taxas de produção diferentes em equipamentos adjacentes (semicontínuos)

A existência ou não de armazenagem intermediária e em que forma ela estará presente depende da política de processamento adotada. Os tipos de política de armazenagem intermediária descritos na literatura são (Rodrigues,M.T. (1992)):

1) Armazenagem dedicada: o tanque está disponível para apenas um dos estágios de processamento, não sendo compartilhado com os demais.

2) Unlimited Intermediate Storage (UIS): neste caso, não existe limite para o número de produtos ou quantidade de material transferido para os tanques intermediários. existindo armazenagem suficiente para atender qualquer demanda. É a política mais flexível.

3) Finite Intermediate Storange (FIS): neste tipo de política, os tanques estão disponíveis em quantidade limitada sendo compartilhada entre os estágios de processamento. Neste caso, o material processado só é transferido se houver disponibilidade de tanque, se não, permanece no equipamento até que possa ser transferido.

4) Zero Wait (ZW) : não existe armazenagem intermediária e o material intermediário processado deve ser imediatamente transferido para o próximo estágio de produção. Este tipo de política é conveniente quando se está processando material instável já que, neste caso, o processamento deve seguir sem interrupção sob pena de degradar o material instável.

5) No Intermediate Storange (NIS): não existem tanques intermediários e o material processado permanece no equipamento se houver necessidade de aguardar até que o próximo estágio esteja liberado.

É muito comum em processos químicos uma mistura deste tipos de política de armazenagem definindo-se o que se denomina política Mixed Intermediate Storange (MIS). No presente trabalho a política de armazenagem intermediária adotada foi um misto da política NIS e da política ZW.

A existência ou não de tanques intermediários tem reflexo imediato no tempo total que as tarefas permanecem no shop e, conseqüentemente na função custo adotada. As relações de recorrência para a política UIS pode ser encontrada em Ku , Rajagopalan e Karimi (1987). Em Campos (1993) pode-se obter algumas citações para as políticas FIS, NIS e ZW.

2..3.4 - PRINCIPAIS VARIÁVEIS E CRITÉRIOS DE DESEMPENHO

Segundo Rodrigues (1992) as principais variáveis e critérios de desempenho utilizados no seqüenciamento de tarefas são:

A) Principais variáveis

* Tempo de processamento - (t_{ij})

Quantidade de tempo que a tarefa i está sendo executada no processador j.

* Instante de Término (Completion Time) - C_{ij}

Instante em que a tarefa i está terminada no processador j. Se M é o número total de processadores disponíveis, C_{iM} representa o instante de tempo em que a tarefa i completou seu processamento no último equipamento e está pronta para deixar o shop.

* Instante de Pronto (Ready Time) - (r_i)

instante em que a tarefa i está pronta para ser executada.

* Instante de início (Beginning Time) - ($B_{i,j}$)

instante em que a tarefa i iniciou seu processamento no equipamento j

* Instante de Fim (Due date) - (d_i)

instante em que a tarefa i deve estar concluída

* Tempo de residência - (F_i)

Quantidade de tempo que uma tarefa i permanece no sistema = $C_{iM} - r_i$

* Lateness ou defasagem (L_i)

Quantidade de tempo que excede a data desejada para o término da tarefa i = $C_{iM} - d_i$

* Tardiness ou atraso (T_i)

Definido como a defasagem positiva (atraso), quando $L_i > 0$, ou zero se a defasagem é negativa ($L_i \leq 0$) = máx (L_i, 0)

B) Critérios de otimização

O objetivo do seqüenciamento é minimizar os critérios de desempenho relacionados a seguir.

* Tempo Médio de Residência - (F^*)

$$F^* = (1/N) \sum_{i=1}^{N} F_i$$

* Atraso Médio - (T^*)

$$T^* = (1/N) \sum_{i=1}^{N} T_i$$

* Tempo de Residência Máximo - ($F_{máx}$)

$$F_{máx} = máx\ (F_i) \quad \text{onde: } 1 \leq i \leq N$$

* Atraso Máximo ($T_{máx}$)

$$T_{máx} = máx\ (T_i) \quad \text{onde: } 1 \leq i \leq N$$

* Número de Tarefas Atrasadas (N_t)

$$N_t = N \sum_{i=1}^{N} (T_i)$$

onde: $N(T_i) = 1$, se $T_i \geq 0$

$N(T_i) = 0$, caso contrário

* Tempo Total de Processamento - Makespan ($C_{máx}$)

$$C_{max} = máx(C(i,M)) \quad \text{onde: } 1 \leq i \leq N$$

2.3.5 - SEQÜENCIAMENTO X SCHEDULING

O problema de seqüenciamento e alocação de tarefas em uma indústria química requer a manipulação de inúmeros fatores que tornam o problema de difícil resolução. O tempo computacional requerido para a sua solução cresce exponencialmente com o aumento do número de tarefas envolvidas no processo. A presença ou não de armazenagem intermediária em suas diversas formas, a utilização de recursos compartilhados como utilidades, mão-de-obra e matéria-prima entre outros são característica que determinam que estratégia deve ser utilizada para resolve-lo. Neste cenário pode-se definir duas categorias de problemas em função da natureza das restrições apresentadas:

1-seqüenciamento de tarefas

2- scheduling de tarefas

O problema é dito de seqüenciamento quando não existe limitação na oferta de recursos comuns, a política de armazenagem intermediária é do tipo UIS, NIS ou ZW, apenas um processador é responsável pela execução das tarefas em cada estágio de processamento e não existe restrições no prazo de entrega. Neste tipo de problema, deseja-se definir a seqüência

de produção nos processadores de forma a otimizar o critério de desempenho adotado, não sendo necessária qualquer outra decisão a cerca desta alocação.

Os problemas onde a oferta de recursos compartilhados e/ou armazenagem intermediária são limitados, ou existem prazos de entrega bem definidos constituem um problema de scheduling. Neste caso, o instante de início definidos para cada tarefa nos estágios de processamento devem também ser factíveis com respeito às restrições presentes no problema. Assim, apenas o conhecimento da seqüência de produção não é suficiente para se construir a carta temporal de eventos, sendo necessário conhecer os instantes de início de cada operação em cada estágio de produção.

Em função de sua complexidade, o problema de scheduling não apresenta solução simples e, por isso, a sua resolução satisfazendo todas as restrições simultaneamente, em muitas ocasiões apresenta-se inviável. Torna-se necessário, então, desenvolver estratégias de manipulação destas restrições a fim de que seja possível modelar todos os níveis de detalhes presentes em problemas desta natureza. Um modelo comumente empregado é aquele em que se estabelece uma hierarquização das restrições segundo sua importância, ou até mesmo o relaxamento de alguma delas para que seja possível a resolução do problema. Abordagens deste tipo visam reduzir o espaço de soluções possíveis. São citados na literatura modelos onde é estabelecida uma representação simplificada do problema, de forma que é gerado um conjunto de soluções sub-ótimas as quais posteriormente são incorporados os detalhes que anteriormente foram ignorados (Rodrigues,M.T. (1992)).

Apesar de todos os trabalhos publicados nos anos recentes com o objetivo de se resolver o problema de scheduling, estas propostas ainda não foram suficientemente desenvolvidas a ponto de solucionar inteiramente o problema em todos os seus aspectos. Uma discussão a respeito das técnicas para resolução será apresentada posteriormente, mas antes serão descritas as técnicas utilizadas para o seqüenciamento de tarefas, nos quais as técnicas para o scheduling se baseia.

2.4- ESTRATÉGIAS DE SOLUÇÃO PARA O PROBLEMA DE SEQÜENCIAMENTO

A literatura aponta três formas principais de se abordar o problema de seqüenciamento que dependerá basicamente das condições de se formular matematicamente o problema bem

como da possibilidade de se reduzir o espaço de soluções a serem testadas. Estas três abordagens serão apresentadas rapidamente a seguir (Rodrigues,M.T.(1992)).

2.4.1- MÉTODOS DE BUSCA

Como já mencionado, o número de soluções possíveis de se obter num problema de seqüenciamento e alocação é da ordem de $N!^M$ sendo portanto impraticável a enumeração de todas elas quanto maior forem as dimensões do problema. Mesmo para flow shop's restritos, onde a ordem de grandeza do problema é N!, a enumeração completa só é possível para problemas com N pequeno.

Para que se possa obter soluções satisfatórias com um menor esforço computacional, lança-se mão de métodos de busca controlada cuja eficiência depende de sua capacidade de reduzir o espaço pesquisado para a obtenção da solução desejada. Este controle é realizado através de técnicas que visam eliminar os subconjuntos de soluções que não contenham a solução ótima, ou seja, a solução otimizada no que se refere a um critério de otimização previamente definido.

Dentre as técnicas de busca, o método do tipo Branch and Bound é bastante utilizado. Este método consiste em decidir qual a próxima operação a ser seqüenciada e quando ela deve ser executada de acordo com o critério de otimização escolhido.

O procedimento é basicamente uma enumeração parcial inteligente das soluções possíveis. A busca em árvore do tipo BAB pode ser graficamente representada por um conjunto de nós e ramos. Os nós representam uma solução incompleta (solução parcial do problema) e os ramos interligam os nós "pais" com o conjunto de nós chamados "filhos". Depois de decomposto, cada nó pai gera um conjunto de nós filhos em um nível inferior da árvore que é um subconjunto factível das operações ainda não seqüenciadas e alocadas. A enumeração encerra-se quando todas as operações já foram seqüenciadas.

A estratégia BAB contém dois tipos de atividades: ramificação (branching) e sondagem (bounding). A figura abaixo representa a estratégia de solução adotada em cada etapa da busca.

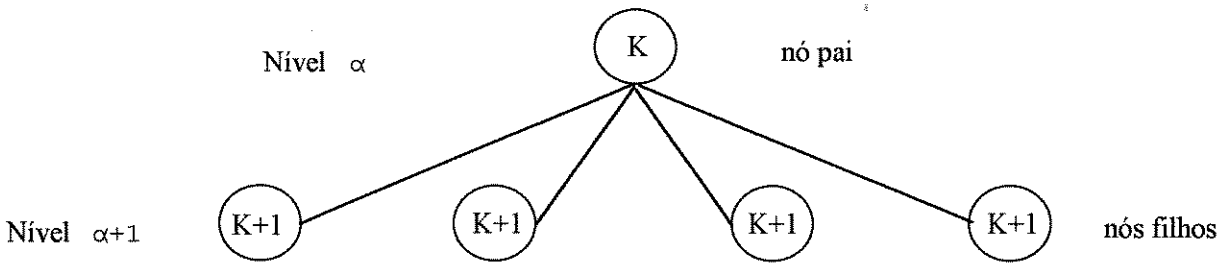


Figura 2.10: Estrutura da árvore de busca

No caso dos problemas de scheduling/seqüenciamento, o nó pai representa uma solução parcial do problema. A etapa de ramificação consiste em acrescentar à solução parcial uma tarefa ou operação ainda não executada. A eficiência da estratégia depende de sua capacidade de reduzir o número de nós candidatos. Quando se considerar restrições sobre recursos compartilhados (problemas de scheduling), a ramificação pode se dar por uma estratégia de busca orientada por restrições (constrained based search) onde o processo de enumeração é limitado pelas restrições presentes no problema, ou seja, os nós a serem expandidos no nível α da árvore de busca são aqueles que respeitam estas restrições ou, no mínimo, aquelas restrições que não podem ser relaxadas. Dentre as restrições que não podem ser relaxadas estão, por exemplo, as de precedência tecnológica, as de fornecimento de matéria-prima entre outras. Em alguns casos pode-se considerar restrições não passíveis de relaxação aquelas cuja violação possa impactar fortemente a função de custo adotada. Se o problema a ser solucionado for muito restrito, o espaço de busca pesquisado pode ser bastante reduzido. Assim, utilizando-se a busca orientada por restrições, o esforço computacional necessário para sua resolução pode ser diminuído. O trabalho aqui apresentado baseia-se neste tipo de estratégia para a redução do número de nós candidatos a expansão.

Na atividade de sondagem, a cada seqüência parcial é associado um custo já consolidado, pois constitui um conjunto de decisões já tomadas. O valor da função custo pode ser representado pelo custo total de produção ou outro critério conveniente. O critério de desempenho mais utilizado é o tempo total de produção ou makespan, onde se busca realizar as tarefas o mais cedo possível. Este valor é composto de duas parcelas: a primeira representa o custo da seqüência parcial construída até o nível onde se encontra o nó, e que normalmente é calculado através do completion time das tarefas (operações) já seqüenciadas. A segunda parcela diz respeito a uma estimativa do custo mínimo futuro das operações que ainda não foram seqüenciadas. Assim sendo, cada nó da árvore tem associado a ele uma previsão do custo total de produção como se já existisse uma seqüência completa que contém a seqüência

parcial representada por aquele nó. A enumeração prossegue seguindo o caminho do ramo que contém o nó com menor custo total.

Existe na literatura um conjunto de funções de estimação do custo mínimo associado com o restante do caminho a ser percorrido em direção a solução final. Estas funções foram desenvolvidas para estimar o makespan em flowshop's UIS quando não há restrições no que se refere a recursos compartilhados. Dentre as principais funções pode-se citar:

- Simple Machine Based Bound (SMBB)
- Full Machine Based Bound (FMBB)
- Non Interference Bound (NIB)
- Job Based Bound (JBB)

Entres as heurísticas citadas, a mais comumente empregada é a FMBB cuja expressão é dada por:

$$Bj = C(K,j) + \sum_{i \in U} t_{ij} + \min \{ \sum_{l=j+1}^{M} t_{il} \} \qquad \text{com: } i \in U \text{ e } 1 \leq j \leq M-1 \qquad (2.2\,a)$$

$$BM = C(K,M) + \sum_{i \in U} t_{iM} \qquad (2.2\,b)$$

$$LB = \text{máx} \{ Bj \} \qquad \text{com: } 1 \leq j \leq M \qquad (2.2\,c)$$

onde: C (k,j) = completion time da seqüência parcial k no processador j

ti, j = tempo de processamento da tarefa i no processador j

M = número total de processadores

U = conjunto de tarefas não seqüenciadas

Uma explanação mais detalhada a respeito das funções heurística para estimativa de custo futuro pode ser encontrada em Rodrigues (1992).

Poucos trabalhos foram desenvolvidos na tentativa de se desenvolver funções para cálculo do custo futuro considerando restrições em recursos compartilhados. Em geral utiliza-se as mesmas regras relacionadas para flowshop's sem restrições de recursos.

O custo mínimo calculado em cada nó da árvore é o limitante inferior ou Lower Bound (LB) da função objetivo que é o controlador da enumeração na busca da seqüência ótima de

produção. O nó com o menor LB será então decomposto em um conjuntos de soluções factíveis gerando novos nós que serão incorporados à árvore em um nível inferior. Estes nós também poderão ser explodidos posteriormente de acordo com o valor do seu LB (Campos (1993)). A figura 9 foi extraída de um exemplo dado por Rodrigues (1992) e demonstra este procedimento.

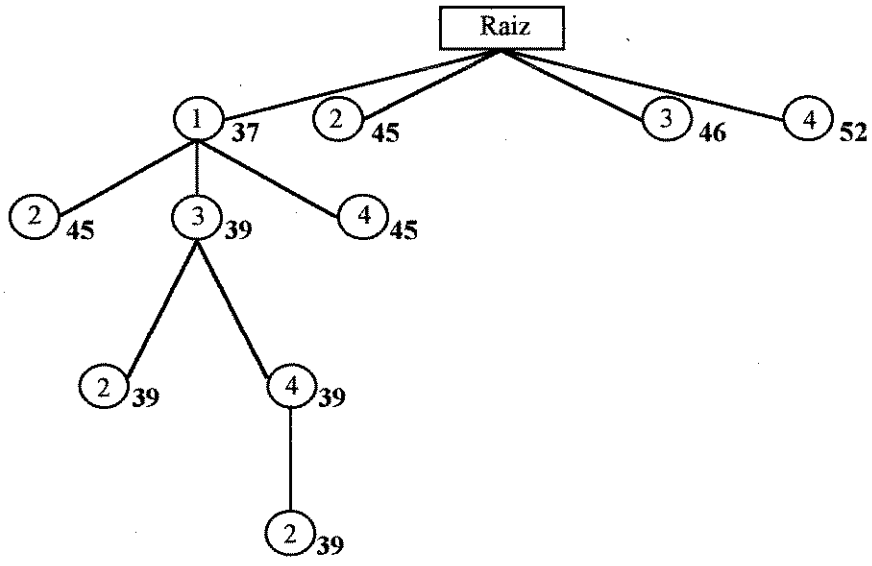


Figura 2.11: Explosão da árvore de busca

Os números dentro do círculo representam as tarefas a serem processadas e os números que aparecem ao lado são os valores do makespan representando o LB calculado para cada nó. A expansão se dá seguindo o caminho de menor LB. Neste exemplo, a heurística utilizada foi a FMBB (Full Machine Based Bound) para um flowshop restrito UIS.

É importante que, em nenhum momento, o custo total, representado pela soma das duas parcelas descritas acima, diminua a medida que se prossegue na enumeração de maneira a garantir que o caminho escolhido represente a seqüência ótima de produção (Rodrigues,M.T. (1992)).

Freqüentemente define-se um parâmetro denominado upper-bound que representa o valor máximo admitido para a função de desempenho adotada. Se em um determinado nível da árvore, a seqüência parcial supera este valor, este caminho será eliminado reduzindo-se assim a quantidade de nós visitado.

São citadas na literatura algumas formas de busca possíveis dentre as quais as mais utilizadas são a busca em profundidade (depth-first) e a busca pelo melhor (best-first ou breadth first). Na busca em profundidade, apenas o nó com menor LB em cada nível é explodido sendo descartados os nós restantes. Se existir mais de um nó com o mesmo LB em

um determinado nível, a escolha de qual será o nó decomposto é aleatória. Na busca pelo melhor, nenhum nó é, a princípio, descartado a menos que o seu LB seja maior que o upper-bound previamente definido, ou se em alguma seqüência completa pesquisada obtenha um LB menor que aquele calculado para ele.

Para se reduzir o número de caminhos percorridos, pode-se utilizar uma mistura destas duas formas de busca onde a princípio uma busca em profundidade define o valor do upper-bound que será levado posteriormente para um procedimento best-first. Assim procedendo, a enumeração poderá ter seu espaço de busca reduzido pelo limite imposto pelo upper-bound calculado pelo depth-first e reduzindo-se também a freqüência com que se retorna a níveis anteriores da árvore para se percorrer um novo caminho (backtracking).

Um grande número de nós visitados leva a um alto custo computacional o que pode até inviabilizar o procedimento. Por isso, a busca via BAB deve estabelecer uma relação ótima entre a precisão do algoritmo determinada pela precisão com que se calcula o lower bound e o upper-bound, e o tempo computacional gasto para se obter a solução.

Ku e Karimi (1990) desenvolveram um procedimento heurístico para uma planta multiestágios em serie com política de armazenagem intermediária UIS que consistia em seqüenciar bateladas ou campanhas de produtos de forma a minimizar penalidades devidas à atraso no prazo de entrega. Foi utilizado como estratégia de solução um algoritmo BAB utilizando dois procedimentos para cálculo do lower bound das seqüências parciais.

A principal limitação do método BAB está na estimativa do custo futuro associado a cada nó. Deve-se, por isto, tomar cuidado com a escolha da função heurística empregada sob o risco de tornar o procedimento de busca ineficiente. A função escolhida deve representar bem o sistema estudado de forma que se possa reduzir a freqüência de backtracking e se obtenha um resultado final satisfatório.

2.4.2- PROGRAMAÇÃO MATEMÁTICA

A programação matemática é outra técnica bastante empregada com a qual se pode resolver um problema de alocação e seqüenciamento de um flowshop. Em função das características normalmente encontradas em problemas deste tipo, o mesmo é formulado como sendo uma Programação Linear Inteira Mista, ou MILP. Uma formulação MILP consiste de um problema de otimização com função objetivo e restrições lineares. Ela difere de uma

programação linear (LP) porque algumas de suas variáveis são inteiras ou binárias (0-1) (Ku,H.M., Rajagopalan,D., Karimi,I. (1987)). As variáveis discreta e continuas devem satisfazer a um conjunto de restrições de igualdade e desigualdade que representam o problema proposto, e que devem se escolhidas de forma a otimizar uma dada função objetivo.

A formulação para um problema simples de flowshop pode ser inferida por relações básicas de recorrência. Em sua forma mais geral, uma programação inteira mista (MIP) corresponde a otimização do seguinte problema:

otimizar:

$$Z = f(x,y)$$

sujeito a:

$$h(x,y) = 0$$

$$g(x,y) \leq 0$$

$$x \in Rn \text{ e } y \in N+m$$

onde x é um vetor de variáveis contínuas e y é um vetor de variáveis inteiras.

Quando a função objetivo f e as restrições h e g são lineares em x e y a formulação é uma Programação Linear Inteira Mista (MILP). A maioria das aplicações de interesse estão restritas ao caso em que as variáveis inteiras y são binárias, isto é, $y \in \{ 0,1 \}$, como já mencionado. Se a função objetivo e/ou as restrições são não-lineares tem-se uma Programação não-Linear Inteira Mista. A forma mais comum encontrada para este caso é linear nas variáveis inteiras e não linear nas variáveis contínuas.(Grossmann,I.E., Quesada,I., Raman,R., Voudouris,V.T. (1992))

Para a resolução do sistema de equações e inequações da formulação MILP utiliza-se pacotes de otimização comercialmente disponíveis, entre os quais os mais citados na literatura são LINDO, GAMS, SCICONIC, MINOS, entre outros.

2.4.3- REGRAS HEURÍSTICAS

Quando a resolução do problema de alocação e seqüenciamento torna-se de difícil resolução, lança-se mão de regras e procedimentos heurísticos que visam sobretudo rapidez na

obtenção de uma solução factível. A eficiência destes algoritmos está relacionada com sua capacidade de se aproximar da solução ótima. Esta estratégia de resolução é, em geral, utilizada para problemas de dimensões elevadas que tem formulação complicada via programação matemática ou branch and bound.

As regras heurísticas são comumente baseadas em observações práticas de onde se extrai dados para se estabelecer padrões que possam reproduzir de forma simplificada o comportamento de uma situação do processo real (Rodrigues,M.T.(1992)).

Para problemas de seqüenciamento em um flowshop, os procedimentos heurísticos podem ser divididos em duas categorias distintas. A primeira gera um conjunto de soluções factíveis segundo alguma regra preestabelecida e escolhe-se a melhor. Nesta categoria pode-se enquadrar as heurísticas baseadas no algoritmo de Johnson (Baker (1974)). A segunda, parte de uma solução inicial e, através de alguns procedimentos específicos, tenta-se melhora-la até que se obtenha uma solução satisfatória (Ku,H.M., Rajagopalan,D.,Karimi,I.(1987)). Como exemplo deste tipo de procedimento pode-se citar a heurística de N.E.H. (Nawaz, Enscore e Ham) para flowshops.

Existem na literatura algumas heurísticas que tratam do problema de flowshop e que se baseiam no algoritmo de Johnson para sua resolução. A estratégia consiste em se reduzir um problema de multiestágios a um shop de 2 estágios/2 processadores e aplicar Johnson para se obter a solução final. Dentre elas pode-se citar a Heurística de Dannenbring (RA - Rápid Access), e a Heurística de Campbell-Dudek-Smith (CDS). A Heurística de Petrov utiliza o mesmo artifício de redução da dimensão do problema. Estas heurísticas tem como função objetivo a minimização do makespan e não consideram tempos de transferência nem setups dependentes da seqüência. Suas formulações estão apresentadas em Rodrigues (1992).

A heurística de Dannenbring foi estendida para outras situações simplificadas que não a de flowshop UIS denominada de heurística RAES (Rapid Access Extensive Search). Na primeira fase gera-se uma seqüência inicial pela heurística RA e em seguida tenta-se melhora-la por um procedimento recursivo de intercâmbio entre tarefas adjacentes avaliando-se o makespan. Em Ku, Rajagopalan e Karimi (1987) pode-se encontrar a formulação desta heurística

Egli e Rippin (1986) propuseram uma estratégia para se resolver um problema de um flowshop generalizado com limitação de recursos compartilhados, setups dependentes da seqüência, tempos de transferência não nulos, instabilidades de intermediários, jornada de trabalho, perfil de vendas de produtos, datas de entrega de matéria-prima e equipamentos

operando em paralelo visando redução de custo total de estocagem. Um estudo mais detalhado deste problema será objeto deste trabalho. No capítulo 3 é apresentada a abordagem proposta pelos autores e no capítulo 4 será apresentado um algoritmo de simulação que utiliza os dados do artigo em questão visando definir quais as operações que podem ser alocadas a partir de um dado estado da planta ao mesmo tempo em que calcula os tempos de início de cada operação obedecendo todas as restrições sugeridas acima.

2.5 - EXTENSÃO PARA O PROBLEMA DE SCHEDULING

Devido a sua complexidade, o problema de scheduling ainda não apresenta solução integral, apesar dos trabalhos desenvolvidos no sentido de se equacionar o problema de alocação temporal de tarefas em um shop com vistas nas limitações de recursos compartilhados e de armazenagem intermediária. Entretanto, algumas alternativas são empregadas para que se possa obter um resultado satisfatório. O tratamento do problema em geral é efetuado através de métodos que se baseiam na técnicas empregadas para o seqüenciamento de tarefas, onde não ocorre restrições de recursos comuns, ou através da utilização de alguns conceitos utilizados para scheduling em job shop's, como o de janelas de demandas.

Duas classes de restrições podem ser definidas para o problema de scheduling: restrições estáticas e restrições dinâmicas. Dentre as restrições estáticas pode-se citar as restrições de precedência e as restrições no prazo de entrega.

Por restrições de precedência entende-se as situações em que uma dada tarefa "a" deve ser executada antes de uma outra tarefa "b" na seqüência de produção. Este tipo de restrição pode ser tratada tanto em uma abordagem tipo BAB quanto por programação matemática. Nas restrições de prazo de entrega a maior parte dos trabalhos desenvolvidos utilizam a abordagem BAB com modificações do limitante inferior proposto por Shwrimer para o flow shop UIS com dois processadores, cujo objetivo é minimizar o custo total dos atrasos (Rodrigues, M. T.(1992)).

Para os casos em que há restrições de recursos comuns e armazenagem intermediária, alguns estudos utilizam-se das técnicas empregadas para o seqüenciamento (Métodos de Busca, Programação Matemática e Métodos Heurísticos), como uma alternativa para a resolução do problema.

Quando se utiliza a técnica de programação matemática, relações matemáticas convenientes devem ser desenvolvidas de forma que se possa obter uma representação adequada do problema com todas suas restrições. Com respeito a recursos compartilhados, deve-se garantir que, em nenhum momento, a demanda destes recursos, ocasionada pela alocação das tarefas nos processadores, exceda a oferta disponível.

No que se refere a armazenagem intermediária, Ku e Karimi (1988) propuseram uma formulação MILP para resolver problemas do tipo FIS (armazenagem intermediária finita) baseados nas formulações para problemas envolvendo as políticas UIS e NIS. A formulação é adequada para se resolver problemas envolvendo N x M com ordem de grandeza de aproximadamente 30. Uma regra heurística (tipo RAES) foi proposta para se reduzir o número de variáveis inteiras de forma a estender esta formulação para problemas com ordem de grandeza de aproximadamente 60. O pacote de simulação utilizado foi o LINDO.

A principal limitação deste método é a dificuldade de se obter uma formulação adequada para todas as restrições envolvidas no problema. Em geral esta formulação torna-se bastante complexa principalmente quando se trabalha com recursos compartilhados o que aumenta consideravelmente o número de variáveis, especialmente das variáveis binárias. Nestas ocasiões, pode haver a necessidade de se relaxar algumas restrições ou se alterar a formulação quando não se obtém uma solução factível.

Dois tipos de abordagem são utilizadas quando se pretende utilizar o BAB para o cálculo de scheduling de tarefas. No primeiro, o tratamento de recursos comuns na construção da seqüência é efetuado simultaneamente com a otimização do critério de desempenho adotado. A principal limitação desta abordagem diz respeito a dificuldade de se obter uma função de custo adequada que represente bem o custo futuro associado a cada nó, função esta que deve levar em conta todas as restrições presentes no problema. Até o presente momento, nenhuma função adequada para este tipo de problema foi desenvolvida. Assim, na falta de uma estimativa mais realista, utiliza-se as heurísticas empregadas para o cálculo do makespan quando não há limitação de oferta de recursos, o que pode levar ao comprometimento da solução final. Desta forma, o instante em que cada tarefa estará alocada em cada processador é calculado levando-se em conta a presença de recursos comuns, porém, a escolha de qual a próxima tarefa que comporá a seqüência é feita com base nas heurísticas desenvolvidas para seqüenciamento de tarefas, que desconsidera estes fatores.

Na outra abordagem tipo BAB, o problema é resolvido como se fosse um seqüenciamento sem a presença de restrições, e o conflito resultante da violação das restrições,

calculado através da construção do perfil de demanda dos recursos compartilhados e armazenagem intermediária, é resolvido posteriormente com o deslocamento no tempo (atraso) das operações que geraram esta violação até que não haja mais conflito. Porém, alguns problemas adicionais decorrente deste deslocamento pode acontecer, como o aumento do makespan da seqüência, se isto resultar em interferência em operações posteriores. Além disto, nada garante que a compatibilização de um recurso por meio deste "shift" temporal, não promova problemas em um outro recurso compartilhado. As seqüências geradas por este método podem então ser utilizadas para se fazer uma busca em seqüências vizinhas através do intercâmbio entre tarefas adjacentes, com a finalidade de se reduzir a função de custo, porém sem a garantia de que ocorra melhoria na solução encontrada.

Também não há na literatura abordagens heurísticas que resolvam adequadamente o problema de alocação de operações, sendo que, em geral, se utiliza aquelas desenvolvidas para flow shop's UIS com busca de melhoria do makespan em soluções vizinhas à encontrada. Uma abordagem deste tipo é o algoritmo IMS proposto por Karimi (1988) (Rodrigues, M.T.(1192)).

Pelo apresentado até aqui, pode-se observar que, nos problemas de scheduling, o instante de início mais cedo de cada operação que compõem as tarefas não pode ser determinado apenas pelo máximo dado na equação 2.1. A alocação destas operações no shop é função também da disponibilidade de recursos compartilhados a cada instante e ao longo do processamento de toda a operação. Assim, uma parcela que represente o instante de início mais cedo em que há disponibilidade de todos os recursos comuns deve ser adicionada àquela equação, ou seja:

$$BT(i,j) = \text{máx}\left\{ \text{máx}\left[C(i-1, j); C(i, j-1)\right]; \text{máx}\left[t_r^*(i,j)_{1 \le r \le R}\right]\right\} \qquad (2.3)$$

onde: BT(i,j) =instante de início mais cedo onde pode ser iniciada a operação i no processador j respeitando-se rota, processadores e recursos compartilhados

R = número total de recursos compartilhados

$t_r^*(i,j)$ = instante mais cedo onde há disponibilidade dos recursos para o processamento da tarefa i no processador j..

A determinação de t^*, em geral, não é uma tarefa simples pois não pode ser definida pelos tempos de processamento, que são dados conhecidos á priori, como ocorre nos problema de seqüenciamento. Estes instantes de início factíveis das operações não alocadas, em uma abordagem tipo BAB, são determinados a medida que a seqüência vai sendo construída, e podem ser alterados cada vez que é realizada novas alocações.

Um outro fator que dificulta a resolução do problema de scheduling diz respeito à definição de qual a próxima operação a ser alocada. Para estes casos, ainda não existe relato na literatura de trabalhos em que a obtenção da solução ótima leve em consideração todas as restrições relativas a presença de recursos comuns, e onde o espaço de soluções não está restrito às seqüências de permutação. Isto se dá ou pela dificuldade de se formular adequadamente algumas restrições (programação matemática), ou pela impossibilidade de se determinar o custo futuro associado a cada nó (BAB), o que compromete a decisão automática de qual a próxima operação a ser alocada. Em vista disto, as soluções obtidas por estratégias deste tipo são, em geral, soluções sub-ótimas construídas pela relaxação de algumas restrições ou utilizando-se funções que não levam em conta a presença de restrições sobre recursos.

Neste contexto, é desejável que esteja disponível um instrumento em que se possa verificar a "qualidade" das soluções obtidas, o que é feito pela elaboração de um **algoritmo de simulação,** que leve em consideração todas as restrições envolvidas no problema, sem relaxação. Neste tipo de procedimento, a construção da seqüência é efetuada pelo usuário e não executada automaticamente pelo algoritmo, não ocorrendo, portanto, o problema da tomada de decisão de qual a próxima operação a ser alocada, ficando esta decisão a cargo do usuário. Assim, no caso dos algoritmos de simulação, cada vez que o usuário aloca uma operação, o simulador recalcula os instantes em que as demais operações que ainda não foram alocadas podem ser iniciadas. Cabe então ao usuário escolher qual a próxima operação que integrará a seqüência. Desta forma, é possível repetir as soluções que foram obtidas pelo algoritmo de scheduling, que em geral são algoritmos relaxados, mas levando-se em conta todas as restrições presentes no problema, ou ainda se testar qualquer seqüência que se deseje verificar o custo.

A elaboração de um algoritmo de simulação onde é levado em consideração restrições relativas a recursos comuns e outras restrições será o escopo deste trabalho.

2.6 - JANELAS DE DEMANDA

Janelas de demanda é um conceito utilizado para a resolução de problemas de job shop's que pode ser estendido para problemas de scheduling em flow shop's, em especial para flow shop's generalizados, a fim de se reduzir o espaço de busca do problema..

Como janela de demanda entende-se o intervalo de tempo em que uma dada tarefa pode ser executada. Este intervalo é delimitado pelo instante de início mais cedo em que a tarefa pode ser iniciada (EBT), dado em função do seu ready time ou de outras restrições como fornecimento de matéria prima ou por indisponibilidade temporária de processadores, e pelo instante de término mais tarde que ela deve ser concluída (LFT), definido em função de algum critério previamente estabelecido, como prazo de entrega ao cliente ou nível de estoque mínimo que deve ser reposto.

A partir das janelas de demanda das tarefas é possível se estabelecer as janelas iniciais de cada operação que compõem estas tarefas, cujo instante de início mais cedo é dado em função da propagação da restrição de precedência tecnológica entre as operações desta tarefa (restrições de rota de produção), assim como o instante de término é calculado a partir do LFT definido para a tarefa pela subtração do tempo de processamento da operação imediatamente posterior na rota.

Campos (1993) desenvolveu um algoritmo BAB, baseado na estratégia proposta por Rodrigues (1992), para scheduling de operações em um flow shop puro com restrições sobre os recursos compartilhados e prazos de entrega utilizando o conceito de janelas de demanda.

O algoritmo permite a verificação da factibilidade da solução que o BAB está construindo, definindo janelas de tempo para cada operação da planta. A atualização destas janelas durante o processo de busca permite uma análise consistente das restrições temporais do problema, visando evitar caminhos que levam a soluções infactíveis. Com este fim é efetuado um pré processamento, baseado em um mecanismo proposto por Erschler, aplicado a cada nó de uma árvore de busca, antes de sua explosão, a fim de se determinar, antes que uma tarefa seja realmente alocada, se sua alocação provoca a violação do instante de término mais tarde (fim da janela de demanda) de uma outra tarefa ainda não alocada que utiliza o mesmo processador. Se isto ocorrer, a operação que provocou esta infactibilidade é eliminada da expansão do nó, não sendo candidata a compor a seqüência naquele nível da árvore.

Isto ocorre quando uma operação é alocada em um dado instante e a janela remanescente de uma outra operação não alocada que utiliza o mesmo processador, torna-se

menor que o tempo necessário para o processamento desta outra operação, após o recorte provocado pela alocação da operação em questão. Assim, a operação que provocou a infactibilidade será eliminada da explosão do nó para evitar a violação do LFT da operação com a qual foi comparada, só podendo ser alocada posteriormente Este tipo de estratégia permite eliminar do espaço de busca aquelas soluções que são infactíveis, reduzindo-se o conjunto de nós visitados.

O conceito de janelas de demanda é utilizado no algoritmo de simulação proposto para a propagação das restrições inter e intra ordem entre estas operações, de maneira que o reflexo da alocação de uma dada operação nas outras operações ainda não alocadas é determinado pelo recorte de suas janelas de demanda provocado pela alocação desta operação.

3- O PROBLEMA PROPOSTO POR EGLI E RIPPIN

3.1- INTRODUÇÃO

No capítulo 2 foram apresentadas as características mais comumente encontradas em problemas de programação de curto prazo. Normalmente os exemplos citados na literatura carecem de dados suficientemente completos, com todas as características citadas. O exemplo de Egli e Rippin apresentado em 1986 propõe um algoritmo heurístico que tem por objetivo determinar um plano de produção factível na qual bateladas de um dado conjunto de produtos devem ser produzidas em uma planta química batelada multiproduto, definindo o instante em que as várias etapas de produção (operações) de cada produto devem ser executadas, ao mesmo tempo que são satisfeitas as restrições da planta, alocando os recursos disponíveis (tempo, processadores, mão-de-obra, matérias-primas, utilidades, etc) de forma a satisfazer a jornada de trabalho pré determinada e a demanda de mercado em um período de tempo predefinido. A sequência de produção escolhida é aquela que apresenta o menor custo total cuja principal parcela é o custo de estocagem final.

Para apresentar a metodologia proposta, os autores tomaram como exemplo uma planta em que deseja-se a produção de quatro produtos denominados D, E, F e H, onde, para cada produto, devem ser atendidas demandas exigidas pelo mercado em um período de tempo de 20 dias (24/07 a 12/08). Para o produto E são apresentadas duas rotas de produção possíveis que possuem diferentes tamanhos de batelada (batch sizes).

Os produtos são processados em duas linhas de produção distintas onde um dos produtos não compartilha equipamentos com os demais, porém compartilha recursos como utilidades. A presença de setups dependentes da seqüência, de processadores operando em paralelo, do elevado consumo de utilidades no momento da transferência de material processado, além do processamento de material instável em alguns equipamentos e da existência de períodos de tempo indisponíveis devido à jornada de trabalho, são fatores que introduzem dificuldades a mais na resolução do problema. O consumo e vendas dos produtos processados se dá ao longo de todo horizonte de planejamento, definindo-se um perfil de vendas para cada produto. Não existe armazenagem intermediária e o material processado, se não for instável, pode aguardar no próprio equipamento até que o próximo processador de sua rota de produção esteja liberado.

Observa-se que, na metodologia apresentada, o seqüenciamento e alocação são efetuados por batelada de produto, ou seja, uma vez iniciada a execução de uma determinada batelada, todas as operações desta batelada são alocadas em seguida. Pode-se definir, desta forma, uma seqüência de produção que representa a ordem na qual as bateladas dos produtos devem ser executadas. A execução de uma batelada não é descontinuada para que uma batelada de um outro produto seja iniciada, mesmo que, para as operações estáveis, esta interrupção seja possível com o material processado aguardando no equipamento até que a operação sucessora na rota de produção seja iniciada. Na estratégia apresentada, só ocorre interrupção de uma batelada para atender restrições de jornada de trabalho.

3.2- DADOS DO PROBLEMA

3.2.1-ESPECIFICAÇÃO DA PLANTA E DO PROCESSO.

3.2.1.1) Equipamentos disponíveis

Estão disponíveis na unidade de produção, os seguintes equipamentos:

* 6 reatores com capacidades que variam de 1600 a 4000 litros (R1, R2, R3, R4, R6 e R7)
* 1 filtro (F1)
* 2 filtros-prensa (FP1 e FP2)
* 1 secador (TRO)
* 1 cabine de secagem.(TRS)

3.2.1.2) rotas de produção:

Para cada produto, foram estabelecidos os equipamentos responsáveis pela execução de cada operação, selecionados dentro do conjunto de equipamentos disponíveis, definindo, desta forma, linhas de produção para estes produtos. As rotas assim determinadas estão apresentadas nas figuras do Apêndice 2. As respectivas quantidade de produto produzida por batelada, denominada de batch sizes, e os tempos de processamento serão apresentados nas tabelas no Apêndice 1. Observa-se que existem duas linhas de produção distintas que não compartilham equipamentos: uma composta apenas pelo produto D e outra composta pelos

produtos H, E e F. O tamanho da batelada é função dos equipamentos que compõem a linha de produção, sendo a quantidade máxima de produto que pode ser produzida pelo equipamento limitante.

3.2.1.3) estabilidade dos intermediários

E especificado se o material intermediário produzido ao fim do processamento em um dado equipamento é estável ou instável. Se é estável, pode permanecer no equipamento até que o próximo equipamento da sua rota de produção esteja disponível para recebe-lo, se não, o início da operação instável deve ser atrasado até que o equipamento responsável pelo processamento da próxima operação em sua rota de produção esteja disponível, tão logo termine o processamento da operação instável.

3.2.1.4) demanda de utilidades e restrições

É fornecida as demandas de vapor e energia elétrica necessárias em todas as operações envolvidas na fabricação dos produtos desejados. Foi estabelecido como limite máximo de utilização destas utilidades 40.000 kcal/h para o vapor e de 50 kW para a energia elétrica. Os dados de demanda estão apresentados nas tabelas a seguir. As durações dos consumos são dadas em horas

Tabela 3.1: consumo de recursos - Produto D

op	Processamento				Transferência			
	energia elétrica		vapor		energia elétrica		vapor	
	consum.	duração	consum.	duração	consum.	duração	consum.	duração
R1	5	4	39500	4	15	1	18400	1
R2	5	14	18400	5	0	1	0	1
R6	0	0	0	0	23	1	0	1
F1	15	9	0	9	45	1	0	1
TRO	30	13	0	13	0	0	0	0

Tabela 3.2: Consumo de recursos - Produto H

op	Processamento				Transferência			
	energia elétrica		vapor		energia elétrica		vapor	
	consum.	duração	consum.	duração	consum.	duração	consum.	duração
R3	5	7	20000	6	25	1	25000	1
R4	0	0	0	0	15	1	25000	1
R7	10	6	25000	5	25	3	0	3
FP1	10	7	0	6	20	1	0	1
TRS	20	34	0	34	0	0	0	0

Tabela 3.3: Consumo de recursos - Produto F

op	Processamento				Transferência			
	energia elétrica		vapor		energia elétrica		vapor	
	consum.	duração	consum.	duração	consum.	duração	consum.	duração
R4	5	11	10000	8	12	1	15000	1
R7	10	5	15000	5	17	1	0	1
FP2	12	2	0	2	42	1	0	1
TRS	42	21	0	21	0	0	0	0

Tabela 3.4: Consumo de recursos - Produto E1

op	Processamento				Transferência			
	energia elétrica		vapor		energia elétrica		vapor	
	consum.	duração	consum.	duração	consum.	duração	consum.	duração
R3	10	8	10000	8	15	3	0	3
FP1	10	2	0	2	20	1	0	1
TRS	20	14	0	14	0	0	0	0

Tabela 3.5: Consumo de recursos - Produto E2

op	Processamento				Transferência			
	energia elétrica		vapor		energia elétrica		vapor	
	consum.	duração	consum.	duração	consum.	duração	consum.	duração
R4	5	8	10000	6	10	2	0	2
FP1	5	2	0	2	20	1	0	1
TRS	20	12	0	12	0	0	0	0

3.2.1.5) interdependências entre os produtos e matérias-primas necessárias

O consumo total de matéria-prima bem como o consumo de intermediários são fornecidos. Este dado é importante para se determinar de que maneira a execução de uma batelada dos produtos finais penaliza o estoque dos produtos que lhe servem de intermediários. D e H são apenas produtos finais, enquanto F é intermediário para a produção de H, e E é

intermediário para a produção de F, além de ambos serem também produtos finais. São necessários 100 Kg do produto E para a produção de F e 200 Kg do produto F para a produção de H.

3.2.1.6) tempo e custo de preparação e limpeza dos equipamentos

O tempo total e o custo envolvido na limpeza e preparação durante o período de produção depende da seqüência escolhida. Estas variáveis são fornecidas para cada combinação possível entre os produtos em todos os equipamentos e são apresentadas no Apêndice 1.

3.2.2- CONDIÇÕES DE ESTOCAGEM

O custo de estocagem é uma parcela importante na composição do custo total de produção sendo usado para determinar o capital empatado na estocagem de produtos. Por isso, é um fator fundamental para a aceitabilidade de uma dada seqüência proposta. Para cada produto, são dadas informações sobre o nível inicial de estoque, o estoque mínimo permitido ao fim do período de planejamento, o estoque máximo permitido, e um estoque "pulmão" estabelecido para suprir demandas urgentes. Retiradas neste estoque incorrem em penalidades e aumento no custo total de produção. Os dados de estocagem são fornecidos na tabela abaixo.

Tabela 3 6: Dados de estocagem

produto	estoque inicial(Kg)	estoque final mínimo(Kg)	estoque pulmão(Kg)	estoque máximo(Kg)
D	1900	2100	1800	40000
E	2370	2100	2000	30000
F	2170	1800	1500	20000
H	2500	2500	2000	25000

3.2.3- DADOS DE PLANEJAMENTO E MERCADO

3.2.3.1) Horizonte de tempo e jornada de trabalho

As datas de início e fim do período de planejamento devem ser especificadas, assim como a jornada de trabalho regular adotada durante este período. Para o exemplo dado, o horizonte de planejamento inicia-se no dia 24/07 e termina no dia 12/08 contabilizando 20 dias ou 516 horas disponíveis para a término de todas as bateladas. A jornada de trabalho começa às 6:00 h de segunda-feira e a planta opera continuamente até às 18:00 h do sábado. Durante este período, ocorre uma interrupção na produção no dia 31/07 às 20:00 h até 2/08 às 6:00 h.

3.2.3.2) datas de vendas

São fornecidos os perfis de demanda para cada produto, ou seja, as datas de entrega dos produtos para os clientes e respectivas quantidades solicitadas por dia. O perfil diário de venda dos produtos finais está apresentado na tabela a seguir.

Tabela 3.7: Perfil de vendas

DIA	Produto D	Produto E	Produto F	Produto H
24/07	50	20	100	100
25/07	100	30	50	150
26/07	50	0	20	50
27/07	60	0	100	50
28/07	80	280	50	70
29/07	0	0	0	0
30/07	0	0	0	0
31/07	40	50	50	130
01/08	0	0	0	0
02/08	200	100	30	50
03/08	0	130	0	30
04/08	120	40	20	50
05/08	0	0	0	0
06/08	0	0	0	0
07/08	110	10	10	30
08/08	450	20	0	50
09/08	10	0	50	50
10/08	10	10	60	30
11/08	60	0	10	10
12/08	0	0	0	0

3.2.3.3) entrega de matéria-prima

Existem limitações no fornecimento de matéria-prima do produto E sendo necessário, portanto, adequar o scheduling as datas de recebimento deste recurso. Desta forma, só é possível o início de uma batelada deste produto se houver matéria-prima disponível. Na tabela abaixo estão especificadas estas datas.

Tabela 3.8: Datas de fornecimento de matéria-prima

ENTREGA	DATA	QUANTIDADE(Kg)
1	24/07 às 00:00 h.	1000
2	5/08 às 12:00 h.	1500
3	13/08 às 12:00 h.	2000

3.2.4-ESPECIFICAÇÃO DAS CONDIÇÕES INICIAIS DA PLANTA

Antes do início do período de produção, a planta já vem sendo ocupada por uma batelada de D e uma de H que invadem o período considerado, terminando após o dia 24/07. Estas bateladas são computadas para satisfação da demanda exigida dos produtos D e H. Assim sendo, a primeira batelada de D é concluída às 24 horas do dia 24/07 e a primeira batelada de H é concluída às 6 horas do dia 26/07, sendo o material processado enviado para estocagem final.

3.3- BALANÇO DE MASSA

Neste item será efetuado o cálculo da produção requerida e do número de bateladas necessárias para cada produto de forma a suprir a demanda de mercado. A produção requerida é determinada pelo perfil diário de vendas de cada um dos produto.

3.3.1- PRODUÇÃO REQUERIDA.

É definida como a soma das demandas externas durante o período de planejamento (venda do produto para clientes externos) e das demandas internas, (quando o produto é

intermediário para a produção de um outro produto final), como é o caso dos produtos E e F. A este valor é adicionado a diferença de estocagem entre o estoque final mínimo previamente estabelecido para o produto e o seu estoque inicial. Assim:

PRODUÇÃO (P) = D + DI + (EF - EI)

a) demandas externas (D):

D = 1340 kg

E = 690 kg

F = 550 kg

H = 850 kg

b) demandas internas (DI):

E = 100 kg para produção de F

F = 200 kg para produção de H

c) diferença na estocagem (EF - EI):

(EF - EI) = estoque final mínimo - estoque inicial

PRODUTO D

P = 1340 + 0 + (2100 - 1900)

P = 1540 kg

PRODUTO E

P = 690 + 100 + (2100 - 2370)

P = 520 kg

PRODUTO F

P = 550 + 200 + (1800 - 2170)

P = 380 kg

PRODUTO H

P = 850 + 0 + (2500 - 2500)

P = 850

3.3.2-NÚMERO DE BATELADAS A SEREM PRODUZIDAS:

O número total de bateladas que devem ser produzidas por produto é estabelecido dividindo-se a produção requerida pela quantidade de produto possível de ser produzida, considerando-se os equipamentos definidos para a produção de cada produto, ou seja, pelo seu tamanho de batelada (batch size).

$$N° = P / BS$$

onde: P = produção requerida

BS = tamanho da batelada

Assim, o número de bateladas que devem ser produzidas por produto estão apresentados na tabela abaixo.

Tabela 3.9: Número total de bateladas a serem produzidas

PRODUTO	PRODUÇÃO	BS	N°	N° REAL	PROD.REAL
D	1540	400	3,85	4	1600
E1	520	300	1,73	2	600
E2	520	240	2,16	3	720
F	380	600	0,63	1	600
H	850	300	2,83	3	900

3.4- DETERMINAÇÃO DO PLANO DE PRODUÇÃO

A estratégia proposta obedece a uma seqüência de decisões do tipo:

estágio 1 - definição de grupos de produtos independentes

estágio 2 - determinação da lista de prioridades

estágio 3 - definição dos blocos estáveis

estágio 4 - enumeração das seqüências de produção factíveis

estágio 5 - deslocamento no tempo para início o mais tarde possível

estágio 6 - retorno no tempo para obedecer restrições

estágio 7 - deslocamento no tempo para atender restrição de recebimento de matéria- prima

estágio 8 - determinação do custo das seqüências de produção aceitas

Para cada decisão tomada é necessário simular o cenário obtido, construindo a carta de Gantt até aquele ponto de desenvolvimento do programa de produção.

O objetivo é gerar um programa de produção que satisfaça as restrições do problema. Os autores propõem uma estratégia de solução que não tem o objetivo de obter a solução ótima. Na determinação do programa de produção, todas as seqüências possíveis entre os produtos desejados são geradas por um procedimento de enumeração. Seqüências não factíveis são progressivamente eliminadas de forma que apenas as candidatas factíveis são objetos de futura avaliação. Para cada seqüência, custos de estocagem são minimizados estabelecendo a execução de cada batelada o mais tarde possível ao mesmo tempo em que obedecem as datas de entrega dos produtos finais.

A alocação das operações que compõem as rotas de produção dos produtos no tempo deve ser tal que satisfaça as restrições de jornada de trabalho e limitação de recursos. Só ocorre violação em alguma restrição, quando mais de uma batelada de produtos diferentes ou do mesmo produto estão sendo processadas na planta ao mesmo tempo. Se isto se verificar, deve-se mover para um tempo de início mais cedo uma ou mais destas bateladas que estão contribuindo para tal violação, uma vez que as mesmas inicialmente estão dispostas o mais tarde possível no horizonte de tempo, como será abordado posteriormente na descrição do estágio 5 da metodologia de cálculo. Este retorno no tempo é efetuado até que não haja mais violação da restrição considerada. Para cada seqüência de produção testada, quando ocorre excesso de demanda de recursos comuns, é executado o "shifting" das bateladas seguindo a mesma ordem com que aparecem na seqüência de produção. Se assim procedendo a restrição é satisfeita, o algoritmo continua sua execução avaliando se ocorre nova violação de restrições em um instante de tempo anterior, não sendo considerado o "shift" das outras bateladas conflitantes. Desta forma, não são investigadas todas as possíveis formas de correção da demanda.

Um resumo dos diversos estágios do procedimento de cálculo é apresentado abaixo. Nos estágios 1, 2, e 3 são efetuados cálculos preliminares que evitam repetições desnecessárias em estágios posteriores. No Apêndice 3 é apresentado um fluxograma simplificado do procedimento utilizado pelos autores.

3.4.1-DEFINIÇÃO DE GRUPOS DE PRODUTOS INDEPENDENTES

Quando os produtos podem ser reunidos em grupos ou linhas de produção de tal forma que estes diferentes grupos não possuam produtos intermediários em comum nem compartilham equipamentos e demais recursos então, para propósito do algoritmo de scheduling, os produtos dentro de cada um dos grupos podem ser seqüênciados como se estivessem em uma planta independente. Porém, estes grupos não podem ser considerados separadamente quando restrições na demanda total de utilidades ou outros recursos forem compartilhadas por todos os produtos envolvidos no scheduling. Para o exemplo exposto, quando não são levadas em conta restrições nas utilidades, pode-se definir duas linhas de produção distintas: uma composta apenas pelo produto D e outra pelos produtos E, F e H.

3.4.2- DETERMINAÇÃO DA LISTA DE PRIORIDADES

A lista de prioridades provê uma seqüência inicial que deve ser utilizada inicialmente no procedimento de enumeração. Os primeiros a serem alocados na lista de prioridades são os produtos finais puros seguidos por aqueles que estão um estágio abaixo dos produtos finais e estes pelos que estão dois estágios abaixo daqueles mesmos produtos. Desta forma, a lista de prioridades para a linha de produção 2 é:

H,F,E1,E2

O produto H é apenas produto final. O produto F é intermediário do produto H e o produto E é intermediário do produto F.

3.4.3 - DEFINIÇÃO DOS BLOCOS ESTÁVEIS

Deve-se neste estágio se observar ao fim de quais operações o material processado é instável, de modo que, se houver necessidade de se interromper uma batelada para satisfazer restrições de jornada de trabalho, não ocorra degradação deste material e conseqüente perda da batelada. A batelada deve então ser deslocada até um instante no qual não ocorra violação desta restrição. Esta informação é disponível como dado do problema.

3.4.4 - ENUMERAÇÃO DAS SEQÜÊNCIAS DE PRODUÇÃO FACTÍVEIS

Neste estágio, a alocação se dá como se a planta operasse continuamente sem se observar as restrições de jornada de trabalho, de demandas de utilidades ou de fornecimento de matéria-prima que serão observadas nos estágios posteriores.

As seqüências dos produtos são geradas, de forma que atendam as demandas de mercado ao fim do horizonte de planejamento. Uma batelada do primeiro elemento da lista de prioridades é iniciada tão logo os equipamentos estejam disponíveis. Após a alocação de cada produto da lista, o estoque é atualizado dia-a-dia. Se algum produto cair abaixo do estoque "pulmão" aquela alternativa será rejeitada. Se o produto cujo estoque caiu abaixo do permitido foi aquele que acabou de ser seqüênciado, deve-se voltar ao elemento anterior e considerar outra alternativa na lista de prioridades. Se o produto for outro, outro produto pode ser considerado a partir deste ponto. Quando as alternativas são exauridas em algum nível da seqüência, deve-se retornar sucessivamente aos estágios anteriores até que todas as possibilidades tenham sido esgotadas em todos os níveis. São obtidas, desta forma, várias seqüências que satisfazem as restrições de nível de estoque, antes e depois do período de produção. Estas seqüências são passadas para os estágios de cálculo posteriores.

As seqüências da linha de produção 2 que atenderam as exigências de nível de estoque neste estágio foram:

H, H, E1, F, E2, H
H, H, E1, E2, F, H
H, F, H, E1, E2, H
H, F, E1, H, E2, H
H, F, E1, E2, H, H
H, E1, H, F, E2, H
H, E1, H, E2, F, H
H, E1, E2, F, H, H
H, E1, F, H, E2, H
H, E1, F, E2, H, H

Todas as seqüências desta linha de produção iniciam-se com uma batelada de H visto que esta batelada já estava sendo produzida na planta no início do período de planejamento.

3.4.5 - DESLOCAMENTO NO TEMPO PARA INÍCIO O MAIS TARDE POSSÍVEL

Como no estágio anterior as bateladas são iniciadas o mais cedo possível, é bastante provável que algum produto esteja disponível em estoque bem antes do requerido pelo mercado o que acarretaria em aumento do custo total de produção. Por isso, neste estágio, é efetuado o deslocamento de todas as bateladas para o tempo de início mais tarde. Este "shift" dos produtos é feito começando desde o fim do horizonte de planejamento e procedendo o retorno no tempo desde a última até a primeira batelada da seqüência.

3.4.6 - REAVALIAÇÃO DO PROGRAMA OBTIDO PARA ATENDER RESTRIÇÕES

Até este estágio, as restrições de jornada de trabalho e de demanda de utilidades, não haviam sido observadas. Isto é feito promovendo-se o retorno no tempo de operações ou dos blocos estáveis, uma vez que as operações foram movidas no estágio anterior para o tempo de início mais tarde. Deve-se lembrar que as bateladas só podem ser interrompidas durante um fim -de -semana, por exemplo, se o material que foi processado imediatamente antes da interrupção é estável podendo, desta forma, permanecer no equipamento aguardando o reinicio das operações.

O mesmo procedimento é utilizado para atender as restrições nas demandas de utilidades. Um ou mais produtos que contribuem para o excesso de demanda são movidos no tempo até que este excesso seja eliminado, seguindo o procedimento já descrito anteriormente.

No exemplo considerado, durante a avaliação da seqüência posteriormente escolhida como ótima, as primeiras datas encontradas depois de ter sido efetuado o "shift" para satisfazer a jornada de trabalho foram:

PRODUTO	DATAS
D	24,02,08,11
H	26,28,12
F	10
E	29,07

Observa-se, então, um excesso na demanda de vapor no dia 10/08 e de energia elétrica no dia 11/08. O procedimento descrito acima é executado e o produto D move-se retornando no eixo do tempo para eliminar estes excessos. As novas datas encontradas foram:

PRODUTO	DATAS
D	24,31,05,09
H	26,28,12
F	10
E	29,07

No dia 29/07, o limite de demanda de vapor é ainda ultrapassado, porém, se o material estável processado nos reatores 2 e 6 do produto D permanecer nos equipamentos por um período de duas horas, pode-se evitar a operação simultânea do reator 3 operando na produção de E1 e das primeiras 4 horas de operação do reator 1 operando para a produção de D, uma vez que este último equipamento consome toda a demanda de vapor disponível.

3.4.7-DESLOCAMENTO NO TEMPO PARA ATENDER RESTRIÇÃO DE RECEBIMENTO DE MATÉRIA-PRIMA

Se ocorre falta da matéria-prima de algum produto, não é possível iniciar uma batelada deste produto até que o suprimento seja reposto. Neste exemplo, uma segunda batelada do produto E só poderá ser iniciada depois das 12:00 horas do dia 05/08. Desta forma, o início da batelada é atrasado e as novas datas encontradas foram:

PRODUTO	DATAS
D	24,31,05,09
H	26,28,12
F	10
E	29,08

Em função deste atraso, o estoque "pulmão" do produto F é penalizado aumentando o custo de produção.

3.4.8 - DETERMINAÇÃO DO CUSTO DAS SEQÜÊNCIAS DE PRODUÇÃO ACEITAS

O custo das seqüência de produção é composto pelo custo de estocagem, custo de "changeover" e custo de utilidades. Neste estágio, é levantado o custo das seqüências que

foram aceitas nos estágios anteriores e é então escolhida aquela com menor custo total. A melhor seqüência escolhida foi:

H,H,E1,E2,F,H

4- ESTRATÉGIA DE SIMULAÇÃO PARA SCHEDULING DE OPERAÇÕES EM PLANTAS MULTIPRODUTO

4.1-INTRODUÇÃO

Neste capítulo será apresentado o procedimento desenvolvido para simulação da carta de Gantt e determinação dos ramos candidatos em uma estratégia BAB utilizando os dados fornecidos por Egli e Rippin (1986). Dada uma seqüência parcial, a estratégia proposta indica quais as operações que são candidatas a compor a seqüência em um dado nível da árvore de busca além de fornecer o instante de início mais cedo de execução de cada operação tendo em vista a limitação de recursos compartilhados e processadores.

No capítulo 2 foi mostrado que a limitação da solução do problema às seqüências de permutação e, conseqüentemente, limitação da abertura da árvore considerando tarefas (ou produtos), pode conduzir a soluções sub-ótimas. A abertura da árvore com base em operações permite a obtenção de soluções do tipo crossing ou by-pass mas, por outro lado, a dimensão da árvore de busca cresce fortemente.

Este exemplo apresenta também uma situação não discutida até o momento na literatura que é o processamento de produtos em linhas de processamento independentes, mas que compartilham os mesmos recursos (energia elétrica e vapor). Esta situação é bastante comum na indústria química.

O algoritmo de simulação proposto é composto de 6 etapas onde os seguintes procedimentos foram efetuados:

1- definição das janelas de tempo para cada batelada

a) cálculo do instante de início mais cedo (EBT)

b) cálculo do instante de término mais tarde (LFT)

2- cálculo das janelas de tempo de cada operação a partir das janelas definidas para as bateladas

3- definição do conjunto de operações candidatas (conjunto CAN)

4- escolha da operação a ser alocada e do seu instante de início

5- teste de consistência do instante de início escolhido para a operação que acabou de ser alocada

6- alocação de uma operação e cálculo dos instantes de início mais cedo de cada operação não seqüenciada, promovendo recortes nas janelas de tempo.

Serão apresentados alguns exemplos ilustrativos da construção de algumas seqüências parciais de operações utilizando os passos do algoritmo proposto.

No Apêndice 4 pode-se encontrar o algoritmo simplificado do procedimento proposto e um exemplo que ilustra a alocação de uma seqüência completa. O Apêndice 5 apresenta uma listagem completa do programa do simulador desenvolvido.

4.2.- DETERMINAÇÃO DAS JANELAS DE TEMPO

4.2.1- DETERMINAÇÃO DO EBT

Foi definido no capítulo anterior, o início do horizonte de planejamento como o dia 24/07, sendo necessárias 4 bateladas do produto D e 3 bateladas do produto H para suprir as demandas requeridas destes produtos. Porém, é dado no problema que nesta data uma batelada do produto D e uma do produto H já haviam sido iniciadas respectivamente às 12 horas e às 24 horas do dia 21/07, sendo necessário então o processamento de apenas mais 3 bateladas de D e 2 bateladas de H. Desta forma, na data prevista para o início do período de planejamento, alguns processadores que compõem as suas rotas de produção, estavam sendo utilizados para a conclusão destas bateladas iniciais, e outros estavam sendo limpos. Portanto, a liberação para o início de uma nova batelada só será efetuada quando estes procedimentos estiverem concluídos.

Assim sendo, para que seja possível a simulação das condições iniciais da planta, considera-se que o tempo zero é 12 horas do dia 21/07, estabelecendo-se que a primeira operação da primeira batelada de D é iniciada neste instante e que a primeira operação da primeira batelada de H é iniciada com t = 12. Desta maneira, o dia 24/07 às 6 horas da manhã (início real do horizonte de planejamento) passa a ser igual a 66 horas. Os instantes de início e de fim das operações destas bateladas seguem como dado no gráfico de Gantt apresentado no artigo, sendo este o estado em que a planta se encontra no início do horizonte de planejamento. Estes instantes são apresentados na tabela abaixo.

TABELA 4.1: Condições iniciais da planta

OPERAÇÕES	BT (HORAS)	ct_op
R1/D1	0	5
R2/D1	4	20
R6/D1	4	20
F1/D1	19	30
TR0/D1	70	84
R3/H1	12	20
R4/H1	19	21
R7/H1	19	30
FP1/H1	69	80
TRS/H1	79	114

Na tabela 4.1, BT representa o instante de início e ct_op representa o instante de fim de cada operação de D1 e H1, conforme é fornecido no artigo.

Simulando-se as condições iniciais da planta através da alocação das primeiras operações nos instantes indicados na tabela acima, o algoritmo de simulação calcula os novos instantes de início e promove o recorte nas janelas das demais bateladas. Assim, os EBT's definidos para elas são calculados de acordo com as seguintes condições:

$$EBT = máx (C_i)$$

onde: C_i = 1-instante em que o processador que faz i está liberado

2- instante em que há disponibilidade de recursos compartilhados

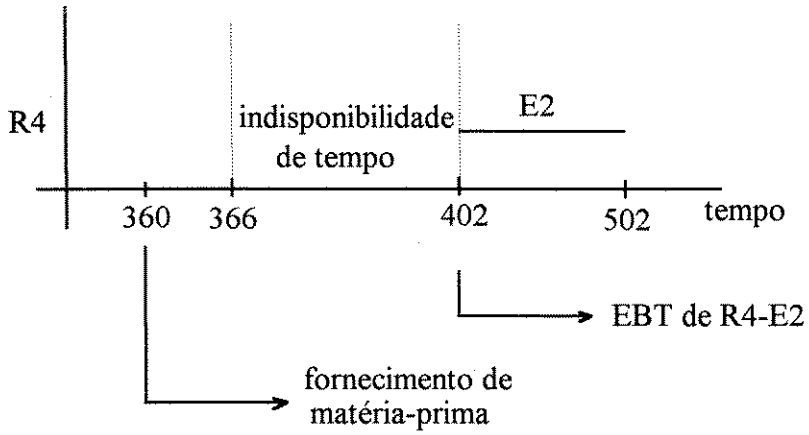
3- instante em que existe disponibilidade de matéria-prima

4- instante de fim das indisponibilidades de tempo devido a jornada de trabalho, se houver invasão de uma destas indisponibilidades pelo processamento da operação considerada.

5- instante em que o processador que faz a próxima operação a i na rota de produção está liberado, se i for instável.

As condições 3 e 4 pode ser exemplificada pelo que ocorre com o produto E. Para este produto observa-se que só existe disponibilidade de matéria-prima para o início de sua segunda batelada a partir das 12 horas do 12° dia (t = 360 horas). Desta forma, seu EBT seria, então, determinado pelo instante de entrega da matéria-prima e não pelo instante de liberação do primeiro equipamento de sua rota de produção que, a princípio, é zero. Observa-se, porém,

que, de acordo com a condição 4, seu instante de início deve ser deslocado pois o processamento da sua primeira operação invade uma das indisponibilidades de tempo se iniciada no instante determinado pelo fornecimento de matéria-prima. O novo EBT passa a ser então o instante de fim da indisponibilidade invadida (402 horas para este caso). A figura abaixo mostra esta situação:



Baseado nas observações acima, duas relações podem ser encontradas para o cálculo do EBT das bateladas posteriores daqueles produtos que necessitam de mais de uma para atender a demanda programada, e cujas bateladas possuem a mesma linha de produção, como por exemplo os produtos D e H. Como os instantes de início destas bateladas posteriores dependem do EBT das primeiras bateladas, a relação encontrada para o produto D, que possui uma linha de produção exclusiva, é a seguinte:

$$EBT\ (D_{(L+1)})\ |_{L=1,N} = EBT\ (D_L) + TP(D,R1) + TT(D,R1) + SU(D,D,R1) \tag{4.1}$$

onde:

L = índice da batelada de D (ex: L = 1 ==> primeira batelada de D)

N = número de bateladas do produto D (N=4)

R1 = primeiro processador da rota de D

TP(D,R1) = tempo de processamento de D em R1

TT(D,R1) = tempo de transferência de R1 para R2

SU(D,D,R1) = set-up de D para D em R1

Para as bateladas de H, que divide seus equipamentos com outros produtos, o EBT das bateladas seguintes é calculado considerando-se o EBT da batelada anterior somado ao tempo de processamento de H no primeiro processador (R3) e ao menor set-up entre H e os outros produtos que também utilizam R3, garantindo-se o menor instante em que as bateladas posteriores podem começar. Então, a equação fica:

$$BT(H_{(L+1)})_{(L=1,N)} = EBT(H_L) + TP(H,R3) + TT(H,R3) + SU(X,H,R3) \tag{4.2}$$

onde:

L = índice da batelada de H

N = número de bateladas do produto H (N= 3)

R3 = primeiro processador da rota de H

X = produto que possui menor set-up com relação a H em R3

Se o EBT de alguma batelada calculado pelas equações acima invadir uma indisponibilidade de tempo, deve ser deslocado para o fim desta indisponibilidade se constituindo no novo EBT.

Para os demais produtos, pode-se considerar que suas bateladas tem EBT igual a zero como já mencionado anteriormente. Assim sendo, os EBT's definidos para cada batelada estão relacionados na tabela 4.2 abaixo.

TABELA 4.2:Instante de início mais cedo das bateladas (EBT)

BATELADAS	EBT (HORAS)
D1	0
D2	66
D3	95
D4	124
H1	0
H2	18
H3	66
F	0
E1	0
E2	402

4.2.2- DETERMINAÇÃO DO LFT

Os estoques mínimo, máximo inicial e pulmão além da demanda interna e o perfil diário de vendas de produtos, que são fornecidos como dados do problema, são utilizados para o cálculo do LFT.

O LFT é calculado em função da variação diária dos estoques partindo-se dos estoques iniciais, debitando-se dia-a-dia as vendas e o consumo interno de cada produto. É definido como o instante em que o estoque de um produto cai abaixo de seu estoque pulmão estabelecido para suprir demandas urgentes. Este instante delimita o prazo máximo em que uma batelada deve estar concluída, pois retiradas neste estoque incorre em penalidades e aumento no custo total de produção. A partir desta data, pode-se definir o LFT por operação pela subtração dos tempos de processamento mais os tempos de transferência em cada processador. Nesta operação, pode ocorrer que as indisponibilidade de tempo relativas à jornada de trabalho sejam invadidas pela janela de alguma operação, porém isto não incorrerá em erro pois o algoritmo não permitirá que ela seja alocada nestas proibições, fazendo a alocação na região da janela onde não há indisponibilidade de tempo. Entretanto, o LFT da batelada completa foi definido em um instante de tempo factível.

Outro fator importante a ser observado é a necessidade de se calcular primeiro o LFT dos produtos que são apenas produtos finais uma vez que estes interferem no estoque dos demais produtos que lhe servem de intermediários. Pode-se construir assim uma seqüência de prioridades baseada na interdependência entre os produtos. A seqüência obtida é a seguinte:

1a linha de produção: E → F → H
2a linha de produção: D

O esquema acima indica que o produto E é matéria-prima para o produto F que por sua vez é matéria-prima para o produto H. Desta forma, o primeiro LFT a ser calculado será o do produto H pois este penaliza o estoque do produto F. Em seguida será calculado o LFT do produto F que penalizará o estoque do produto E. Procedendo desta forma, pode-se definir as datas máximas em que os produtos intermediários são consumidos, pois é dado no problema quanto tempo antes do término da batelada do produto final este consumo é efetuado. Este dado é apresentado na figura 4.1 abaixo.

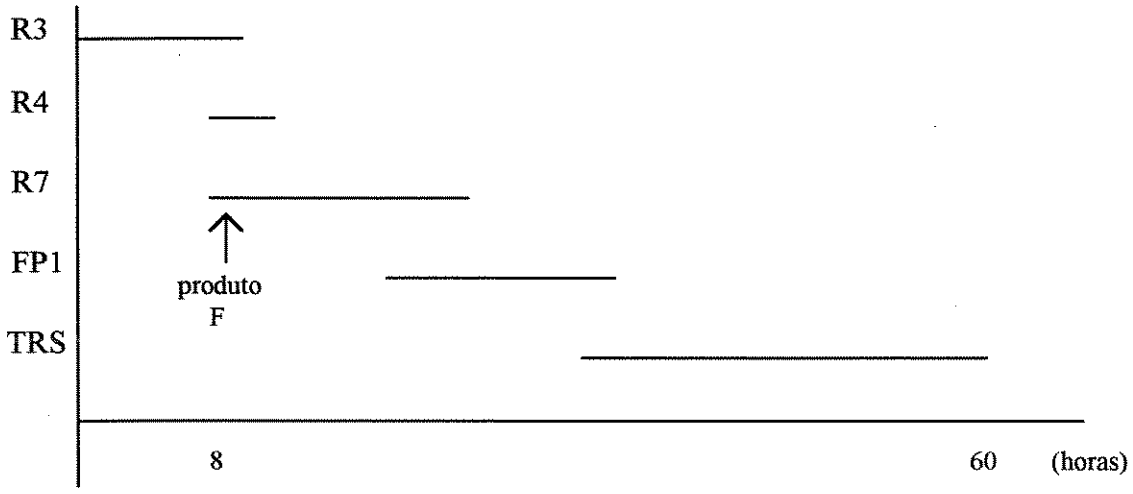


FIGURA 4.1: Carta de Gantt para o produto H

A figura 4.1 mostra que 200 kg do produto F são consumidos no reator 7 a aproximadamente 42 horas antes do término da batelada do produto H. Assim sendo, quando se calcula a variação diária de estoque do produto F, deve-se debitar esta demanda interna na data máxima em que ela pode ocorrer. Esta data só é definida após o cálculo dos LFT's das bateladas de H.

O cálculo do LFT do produto D é indiferente a esta seqüência, por não possuir nenhum produto intermediário que lhe sirva de matéria-prima. O LFT assim definido é, a priori, o instante mais tarde em que uma batelada deve estar concluída.

As Figuras 4.2 e 4.3 exemplificam o procedimento descrito acima. Nota-se que uma batelada do produto deve ser completada no máximo até o momento em que o estoque cai abaixo da linha do estoque pulmão definindo, desta forma, seu LFT.

Na figura 4.2 observa-se que a batelada do produto H que já vinha sendo processada na planta (H1) é completada ao fim do 3° dia (132 horas), sendo adicionado ao seu estoque os 300 kg relativos ao Bach size. Considerando que nenhuma batelada é concluída neste período, no 16° dia (444 horas) a posição de estoque desse produto seria de 2040 kg. No 17° dia (468 horas), deve ser vendido 50 kg e, desta forma, o estoque cai 10 kg abaixo do estoque pulmão que é de 2000 kg. É necessário, então, que uma batelada do produto H seja completada no máximo até este dia para que não ocorra déficit no estoque. Este instante define o LFT da segunda batelada de H. A última batelada deve ser concluída no fim do horizonte de tempo (534 horas) para que se feche o balanço de massa previamente calculado do produto, mesmo que o estoque não esteja abaixo do estoque pulmão. Este será o LFT da terceira batelada de H.

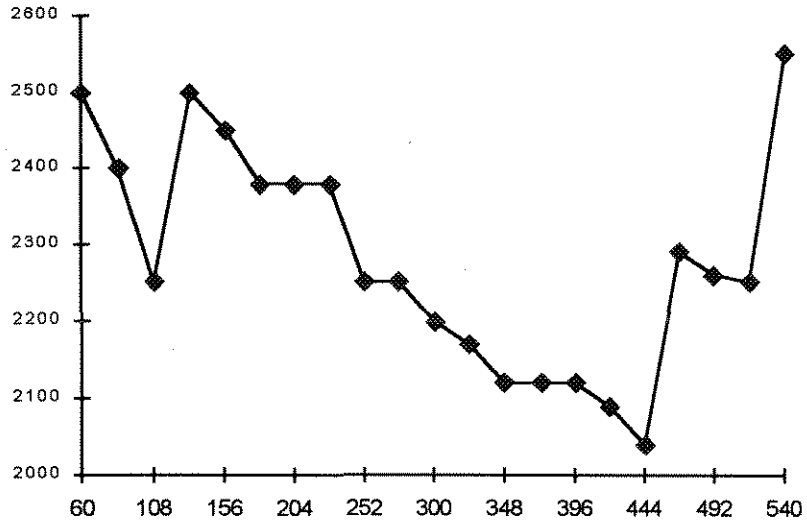


FIGURA 4.2: Variação diária de estoque do produto H

Na figura 4.3 está representada a variação diária de estoque do produto F. Nota-se que no 15° dia (420 horas) está indicada uma retirada de 200 kg de produto que é relativo ao consumo interno devido ao processamento da segunda batelada do produto H, ou seja, aproximadamente 42 horas antes do LFT de H. Após uma venda de 50 kg no 17° dia, o estoque de F cai abaixo de 1500 kg (estoque pulmão) sendo necessária a conclusão de uma batelada para evitar penalidades e aumento no custo de produção. Assim sendo, define-se o 17° dia (468 horas) como a data máxima para o término de uma batelada deste produto. O gráfico mostra ainda, que no 18° dia (492 horas) ocorre uma outra retirada de 200 kg do estoque que seria a demanda para a última batelada do produto H.

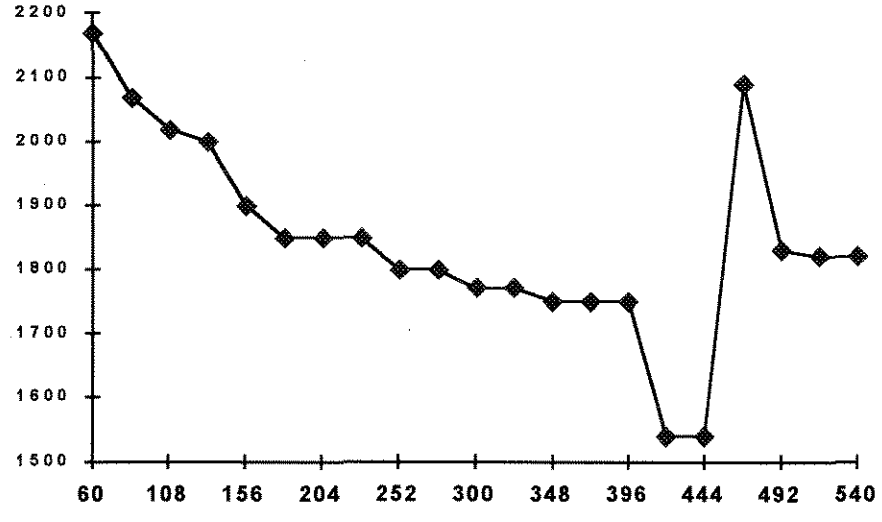


FIGURA 4.3: Variação diária de estoque do produto F

O LFT dos demais produtos é obtidos segundo o mesmo procedimento descrito acima. Para as bateladas iniciais de H e D (H1 e D1), o LFT considerado foi o completion time da última operação destas bateladas. Assim sendo, os LFT's calculados para todas as bateladas são os seguintes:

TABELA 4.3: Instante de fim mais tarde das bateladas (LFT)

BATELADAS	LFT (horas)
D1	84
D2	300
D3	444
D4	516
H1	114
H2	468
H3	534
F	468
E1	248
E2	444

Finalmente, após efetuados todos os cálculos, as janelas de tempo encontradas para cada batelada são mostradas na Tabela 4.4 abaixo. WE e DWE representam os instantes de início e fim das indisponibilidades de tempo estabelecidas em virtude da jornada de trabalho.

TABELA 4.4: Janelas de tempo de cada batelada

	EBT	WE(1)	DWE(1)	WE(2)	DWE(2)	WE(3)	DWE(3)	WE(4)	DWE(4)	LFT
D1	0	30	66							84
D2	66			198	234					248
D3	95			198	234	248	282	366	402	444
D4	124			198	234	248	282	366	402	516
H1	0	30	66							114
H2	18	30	66	198	234	248	282	366	402	468
H3	66			198	234	248	282	366	402	534
F	0	30	66	198	234	248	282	366	402	468
E1	0	30	66							198
E2	402									444

Após encontrado os LFT's e os EBT's, observa-se se é possível alocar as operações no intervalo de tempo por eles definidos, respeitando-se as indisponibilidades decorrentes da jornada de trabalho. Se isto não for possível, a restrição do estoque deve ser relaxada, incorrendo em penalidades e elevação do custo. Observa-se que, para todas as bateladas do exemplo apresentado, esta restrição não precisa ser relaxada pois as janelas de tempo são

suficientes para alocar todas as operações que compõem a rota de produção dos respectivos produtos. Uma figura ilustrativa das janelas de cada batelada está apresentada no Apêndice 1.

4.3-ESTRUTURA DE DADOS

A estrutura de dados desenvolvida deve ser capaz de representar todos os fatores que interferem na alocação das operações que se deseja processar. O problema proposto apresenta diversas características que requerem uma estrutura de dados bem definida para que se possa obter os resultados desejados.

Além dos dados que caracterizam cada operação como tempos de processamento, tempos de transferência, processadores responsáveis pela sua execução e consumo de recursos compartilhados durante os tempos de processamento e tempos de transferência, informações relativas à rota de produção, setup's entre tarefas e condições de instabilidade do produto intermediário e de equipamentos que operam em paralelo devem ser fornecidas para que o problema possa ser bem representado.

Devem ser processadas 10 tarefas (T = 10) para satisfazer as demandas interna e externa, sendo 4 bateladas de D, 3 bateladas de H, 1 batelada de F, 1 batelada de E na variante de produção 1 e 1 batelada de E na variante de produção 2, conforme previsto no cálculo do balanço de massa da unidade. Cada batelada é vista como sendo uma tarefa distinta cujos índices são representados pela variável TAR[i]. Estas tarefas produzem um total de 45 operações (O = 45) a serem seqüenciadas, sendo 5 operações por batelada de D, 5 por batelada de H, 4 por batelada de F e 3 por batelada de E nas duas variantes de produção. O número de operações por tarefa é denotado de NOP[i]. Os índices das tarefas e das operações que as compõem estão apresentadas no Apêndice 1, referente aos dados do problema.

Uma variável denominada de B_ANT[i] determina se existe mais de uma batelada para um dada operação, e qual é a operação correspondente a ela na batelada anterior do mesmo produto. Assim, se existir mais de uma batelada para um produto, como ocorre com os produtos D e H, B_ANT[i] é igual ao índice da operação correspondente a operação i na bateladas anterior a TAR[i]. Por exemplo, se 1 é o índice que representa a primeira operação da primeira batelada do produto D, e se 5 representa a primeira operação da segunda batelada

do produto D então B_ANT[5] = 1. Se só existir uma batelada da tarefa composta pela operação em questão, então B_ANT[i]=0.

A rota de produção é representada por duas variáveis distintas. A primeira é denominada de ANT[i] e determina a antecessora da operação na rota de produção. A segunda variável, definida como POS[i], representa sua sucessora. Se uma dada operação é a primeira da tarefa, então ANT[i] = 0, se for a última, POS[i] = 0.

O tempo de transferência é definido como o intervalo de tempo que uma determinada operação necessita para transferir o material processado para o próximo equipamento na rota de produção. O período em que um processador está ocupado recebendo este material é o tempo de transferência da operação anterior. Desta forma, a última operação de uma tarefa não possui tempo de transferência.

O setup é definido como o tempo de preparação e limpeza de um equipamento após a execução de uma dada operação neste equipamento e uma tarefa qualquer que tenha uma de suas operações que utiliza este mesmo processador.

A condição de instabilidade do intermediário processado na operação é definida por uma variável denominada SB[i] que assume o valor 1 se o material for instável e 0 se for estável. O processador que executa cada operação é denominado de PRO[i] e são 11 processadores ao todo (P = 11). Estes processadores podem executar mais de uma operação em mais de uma tarefa.

São fornecidas as janelas de tempo por batelada de produto, conforme valores obtidos no cálculo do EBT e LFT. Uma variável auxiliar, denominada de RELAX[TAR[i]], é definida para se estabelecer se a janela da batelada de i pode ou não ser relaxada. Se sim, a variável assume o valor 0 e se não assume o valor 1. Não é permitido o relaxamento de uma janela se isto incorrer em violação de restrição de matéria-prima, como é o caso da segunda batelada de do produto E.

É informado também o consumo de recursos compartilhados durante o processamento e durante a transferência. O período de tempo em que este consumo ocorre deve ser especificado visto que a demanda não se dá necessariamente durante todo o processamento ou durante toda a transferência de uma dada operação.

Um característica importante do problema proposto por Eglli e Rippin diz respeito à equipamentos que operam em paralelo, o que demanda um esforço adicional na definição dos dados de entrada para a resolução deste problema. Os produtos D e H possuem operações com esta característica em suas rotas de produção. Na rota de produção do produto D, o

material processado no reator 1 é transferido para os reatores 2 e 6 que trabalham em paralelo. Terminado o processamento, estes reatores transferem o intermediário para o filtro 1. As operações executadas nos reatores 2 e 6 possuem o mesmo tempo de processamento e devem ser iniciadas no mesmo instante. As operações paralelas do produto H são as que utilizam os reatores 4 e 7. Estes reatores recebem o material processado no reator 3 que as antecede na rota de produção Apesar de possuírem o mesmo start time, elas não apresentam o mesmo tempo de processamento, com o reator 4 descarregando para o reator 7, que por sua vez descarrega para o filtro prensa 1. Um esquema representativo das rotas de produção de todos os produtos está apresentado no Apêndice 2.

Para estas operações que trabalham em paralelo, as duas variáveis que definem a rota de produção, ANT[i] e POS[i], foram utilizadas de forma a representar este tipo de arranjo. A estratégia utilizada para se resolver este problema de paralelismo entre as operações foi a de se definir como a antecessora das operações executadas em paralelo, a mesma operação, ou seja, para o produto D tem-se que ANT[D.R2] = D.R1 e ANT[D.R6] = D.R1 e para o produto H tem-se que ANT[H.R4] = H.R3 e ANT[H.R7] = H.R3.

Uma distinção deve ser feita entre estes dois tipos de arranjo de operações em paralelo no que se refere a definição da operação que as sucede na rota de produção. Para o caso do produto D, a operação que sucede a aquela executada no reator 2 é a mesma que a do reator 6. Assim sendo, POS[D.R2] = POS[D.R6] = D.F1. Para as operações do produto H o mesmo não acontece pois as duas operações em paralelo não transferem o material processado para o mesmo equipamento, sendo que o material processado no reator 4 é transferido para o reator 7 que continua seu processamento.. Desta forma, POS[H.R4] = H.R7 e POS[H.R7] = H.FP1. Deve-se ressaltar que a estratégia desenvolvida só é válida para o caso de operações em paralelo com tempos de processamento diferentes se esta condição da primeira operação descarregar para sua paralela for verdadeira.

Foi considerado também que a operação H.R4 possui tempo de processamento igual a zero e que o tempo em que o processador permanece ocupado com sua execução é devido às transferências da operação anterior para ela (H.R3), e dela para a operação posterior (H.R7). Este artifício não acarreta erro na determinação dos instantes de início das operações visto que o tempo que o processador permanece ocupado (2 horas) e o consumo de recursos desta operação são preservados.

Uma variável auxiliar denominada PL[i] foi criada para que as operações em paralelo fiquem bem definidas, indicando se a operação possui ou não uma outra que opera em paralelo

com ela. Se PL[i] = 0, a operação não possui paralela. Para as que possuem este tipo de arranjo, PL[i] é igual a operação paralela a i, por exemplo, PL[2]=3 e PL[3] = 2. Isto significa que a operação 2 opera em paralelo com a operação 3, e vice-versa.

Uma outra variável interna auxiliar denominada SBaux[i] foi estabelecida para se determinar se uma dada operação está sendo alocada antes do início de uma outra previamente alocada no processador que a faz, ou para determinar se o início de sua janela foi calculado como estando posicionado antes do instante de início de uma operação previamente alocada. Se esta condição é verdadeira SBaux[i] é igual a 1, se não, SBaux[i] é igual a 0. Esta variável se faz necessária porque é utilizada para se testar a factibilidade do instante de início escolhido para a operação candidata.

Com relação a recursos compartilhados, ficou estabelecido para facilidade de cálculos que, para as operações paralelas do produto D, apenas a primeira consome todo o recurso necessário para a execução das duas, ficando a segunda operação com consumo de recursos compartilhados durante o processamento igual a zero. Durante a transferência de material processado, apenas a segunda operação paralela consome recursos para transferir para a posterior de ambas na rota de produção. Assim procedendo, evita-se que se compute duas vezes o mesmo consumo visto que estas operações são alocadas no mesmo intervalo de tempo.

Este procedimento não se faz necessário para as operações paralelas do produto H pois a primeira operação paralela tem tempo de processamento igual a zero consumindo recursos compartilhados apenas durante a transferência para a operação seguinte, que é também sua paralela.

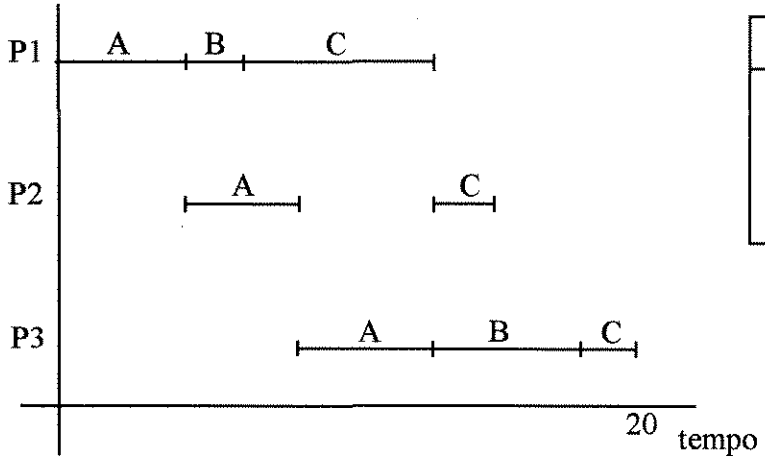
Todos os dados descritos acima podem ser encontrados no Apêndice 1

4.4 - ESTRUTURA DO ALGORITMO

4.4.1- ESTRATÉGIA DE EXPANSÃO DA ÁRVORE

A principal característica da estratégia proposta, diz respeito a expansão da árvore. Quando se trata de seqüências de permutação, a abertura da árvore é efetuada seqüenciando-se tarefas, ou seja, uma vez definida qual a primeira tarefa a ser processada no primeiro

processador, a prioridade nos demais processadores é desta mesma tarefa. O esquema abaixo exemplifica esta afirmação:



	P1	P2	P3
A	4	4	5
B	2		5
C	6	2	2

FIGURA 4.4: Abertura da árvore por tarefa

Na figura acima, a primeira tarefa seqüenciada no primeiro processador foi a tarefa A. Desta forma, a prioridade nos demais processadores é desta tarefa não havendo, portanto, crossing de tarefas em nenhum processador. O makespan é calculado segundo os dados de tempos de processamento fornecidos na tabela, e é igual a 20. A seqüência obtida é então A-P1, A-P2, A-P3, B-P1, B-P3, C-P1, C-P2, C-P3, ou, simplesmente A, B, C.

Para plantas com as características do problema proposto por Egli e Rippin, este procedimento não assegura uma utilização ótima de recursos e processadores uma vez que o equipamento que não é utilizado no processamento da tarefa alocada fica ocioso por um período de tempo maior do que seria realmente necessário o que não assegura makespan mínimo. Assim sendo, observou-se que a expansão da árvore por operação é a melhor política a ser adotada, onde não existe uma seqüência global para todo o shop, havendo entretanto uma seqüência para cada processador. No que se refere a recursos compartilhados, uma ordem de prioridade entre as operações é estabelecida na qual aquela alocada primeiro será a que tem a prioridade no consumo de recursos em um dado instante de tempo. O mesmo exemplo anterior é apresentado na figura 4.5 seguindo este procedimento.

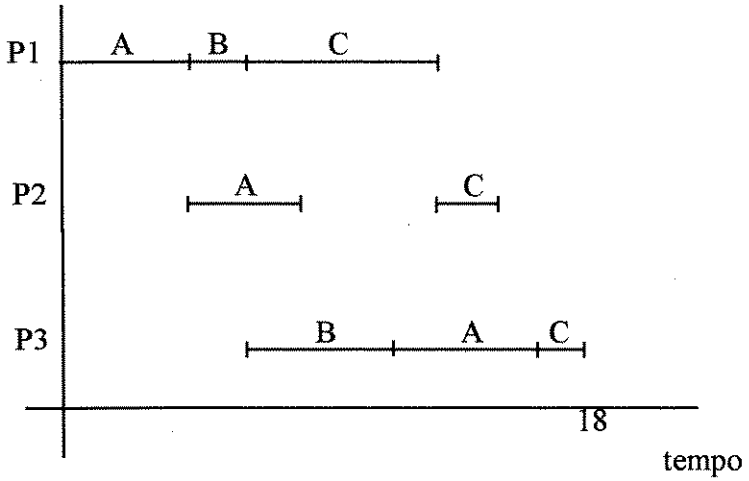


FIGURA 4.5: Abertura da árvore por operação

Na figura 4.5 observa-se que a seqüência obtida no processador 3 é diferente daquela para os demais processadores, onde o produto B inicia seu processamento antes que o produto A no processador 3. Este procedimento também determina um menor makespan que, neste caso, passa a ser igual a 18. Estabelece-se uma seqüência de decisões de alocação das operações que pode ser diferente da seqüência estabelecida para cada equipamento, como pode ser observado na figura 4.6 abaixo:

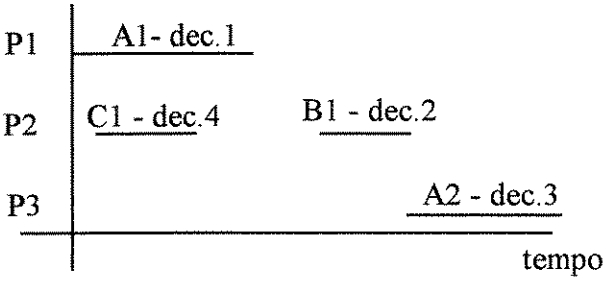


FIGURA 4.6: Seqüência de decisões de alocação

Na figura acima, nota-se que a decisão a cerca da alocação da operação B1 no processador 2 foi tomada antes que para a operação C1, apesar desta ser processada primeiro que B1 neste processador. Para este caso, a seqüência de decisões de alocação das operações nos processadores é A1-P1, B1-P2, A2-P3 e C1-P2.

4.4.2 - DETERMINAÇÃO DAS JANELAS DE TEMPO DE CADA OPERAÇÃO - (Programa Principal)

O primeiro passo seguido pelo algoritmo proposto é o cálculo das janelas de cada uma das operações a partir das janelas previamente definidas para as bateladas dos produtos. O calculo do EBT de uma dada operação é feito partindo-se do EBT inicial da batelada, que é também o EBT da primeira operação. Todos os cálculos efetuados para o cálculo do EBT inicial de cada operação são executados numa função interna denominada de **start_time** que será apresentada mais adiante.

Somando-se ao EBT[i] o tempo de processamento de i mais o tempo de transferência da operação anterior a i na rota de produção, que no caso da primeira operação é zero, obtém-se o EBT da operação seguinte na rota, POS[i].

Da mesma forma, o LFT é calculado a partir do LFT da batelada que é igual ao LFT de sua última operação Neste caso, o cálculo é feito subtraindo-se do LFT[i] o seu tempo de transferência, TT[i], e o seu tempo de processamento, TP[i], para se obter o LFT da operação anterior de i na rota de produção, ANT[i]. Os cálculos para o LFT de cada operação são executados em uma função denominada de **finishtime** e que será apresentada a seguir.

Quando a operação que se está calculando opera em paralelo com sua sucessora (PL $\neq$ 0), este cálculo deve ser modificado para que o EBT calculado seja o mesmo para ambas. Sempre que o algoritmo determina o EBT para uma operação por meio de sua rota de produção, é efetuado um teste de consistência com relação às indisponibilidades de tempo e da condição de instabilidade do produto intermediário processado nas operações, de forma que os valores calculados sejam realmente factíveis. Se ocorrer que no instante inicialmente calculado para a janela de uma dada operação i ou de sua posterior, no caso de i ser instável, o processamento de i ou de sua posterior invadir uma destas indisponibilidades, o novo EBT será definido como o fim desta indisponibilidade invadida. As equações utilizadas para o cálculo em todos os casos estão apresentadas abaixo.

4.4.2.1 - Cálculo do EBT por operação

Os EBT's iniciais de todas as operações são calculados pela função start_time. A seguir serão apresentadas algumas das equações desta função que são mais importantes no

cálculo das janelas iniciais, ficando o procedimento mais detalhado, que leva em conta outros fatores que não são relevantes aqui, para serem apresentados adiante.

Para i variando de 1 até o número total de operações = 45:

A) <u>Se i é a primeira operação de uma batelada (ANT[i] = 0)</u>

$$EBT[i] = W[TAR[i]][0] \tag{4.3}$$

onde:

ANT[i] = 0 →a primeira operação não possui anterior

W[TAR[i][0] = EBT da tarefa a que pertence i

B) <u>Se i é uma operação que não opera em paralelo (ANT[i] ≠ 0 e PL[i] = 0):</u>

$$EBT[i] = EBT[ANT[i]] + TT[ANT[ANT[i]]] + TP[ANT[i]] \tag{4.4}$$

onde:

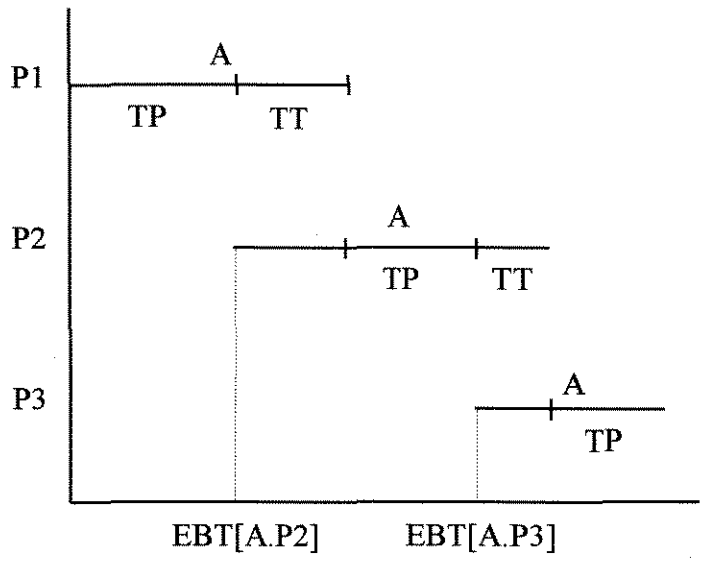
ANT[i] ≠ 0 → a operação i não é a primeira na rota de produção

EBT[ANT[i]] = instante de início mais cedo da operação anterior a i na rota

TT[ANT[ANT[i]]] = tempo de transferência da operação anterior a anterior de i na rota de produção

TP[ANT[i]] = tempo de processamento da operação anterior a i na rota de produção

A figura a seguir exemplifica este cálculo



	TP	TT
P1	7	5
P2	6	3
P3	5	0

FIGURA 4.7 : Cálculo do EBT de uma operação não paralela

C) Se i é uma operação que opera em paralelo

Para as operações que operam em paralelo são considerados dois casos distintos: as operações paralelas que possuem o mesmo tempo de processamento e as operações paralelas que possuem tempo de processamento diferentes, com a primeira descarregando material processado para a segunda. Os dois casos são calculados como descrito abaixo.

C.1) EBT das operações em paralelo e que possuem o mesmo tempo de processamento (ANT[i + 1] = ANT[i] e TP[i] = TP [i+1]) - neste caso, após calculado o EBT da primeira operação em paralelo, apenas força-se que o EBT da segunda operação seja o mesmo

$$EBT[i + 1] = EBT[i] \tag{4.5}$$

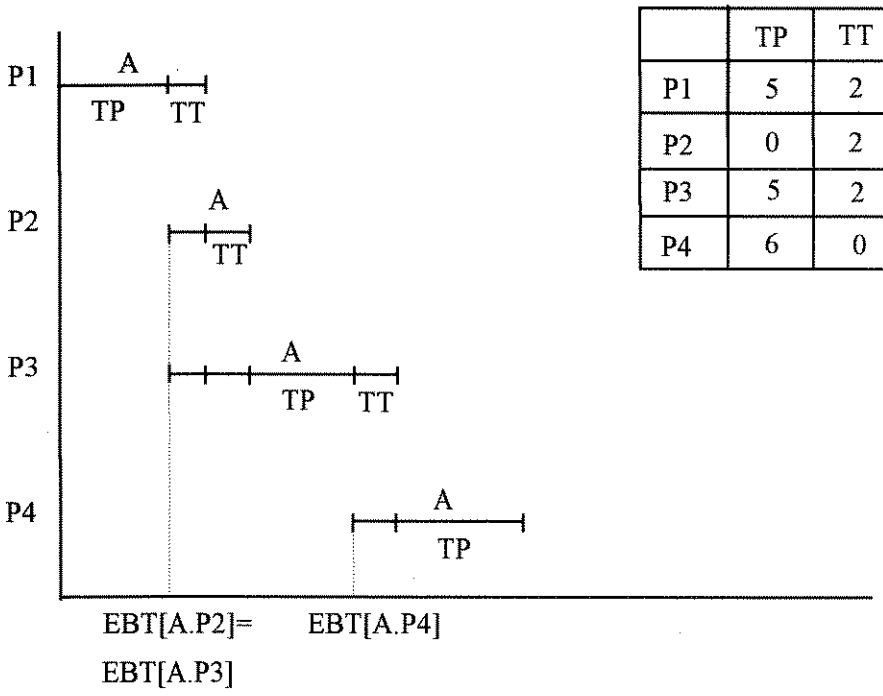
C.2) EBT das operações em paralelo que possuem tempos de processamento diferentes (ANT[i + 1] = ANT[i] e TP[i] ≠ TP[i+1]) - da mesma maneira que no caso anterior, o EBT das operações em paralelo é o mesmo, mas o cálculo do EBT da operação posterior a elas deve ser modificado. Quando ocorre este tipo de arranjo de operações paralelas, como é o caso da R4.H e R7.H que são operações do produto H, o tempo de processamento da operação paralela com menor período de ocupação no seu processador deve ser considerado zero para efeito de cálculos. Isto não acarreta erro se o tempo total de ocupação do

processador para a execução da operação e o consumo total de recursos forem preservados. Para o exemplo estudado. esta afirmação é verdadeira visto que o período de tempo que este processador é ocupado com esta operação (2 horas) é suficiente apenas para receber o material da operação anterior da rota e transferi-lo para a operação seguinte que é paralela a ela. As equações para este arranjo ficam:

$$EBT[i + 1] = EBT[i] \tag{4.6 a}$$

$$EBT[i + 2] = EBT[i + 1] + TT[i - 1] + TT[i] + TP[i + 1] \tag{4.6 b}$$

O calculo descrito acima é demonstrado na figura 4.8



	TP	TT
P1	5	2
P2	0	2
P3	5	2
P4	6	0

FIGURA 4.8: cálculo do EBT para operações em paralelo com tempos de processamento diferentes

4.4.2.2 - Cálculo do LFT por operação - **função finishtime**

Para i variando de 1 até o número total de operações = 45:

Os mesmos tipos de casos descritos para o cálculo do EBT são considerados para o cálculo dos LFT's, que são obtidos a partir da última operação de cada batelada. As equações que executam estes cálculos são descritas abaixo:

A) <u>Se é a última operação de uma batelada- (POS[i] = 0):</u>

$$LFT[i] = W[TAR[i]][1] \tag{4.7}$$

onde:

POS[i] = 0 → a última operação não possui posterior

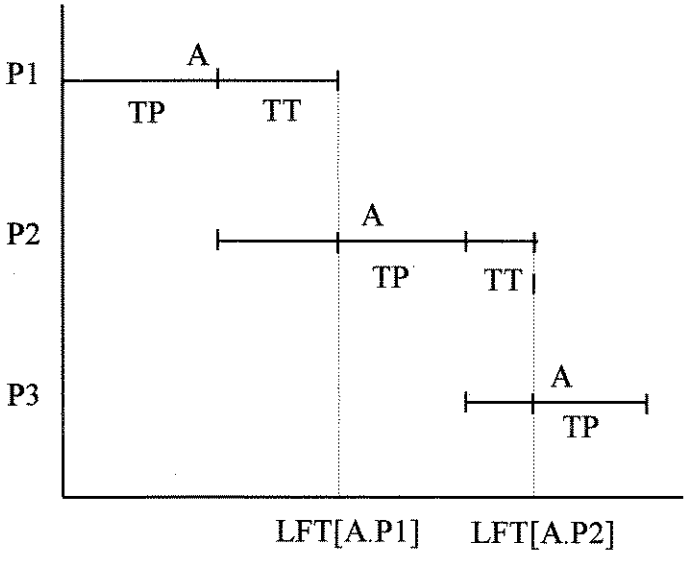
W[TAR[i]][1] = LFT da tarefa a que pertence i

B) <u>Se i é uma operação que não opera em paralelo (ANT[i] ≠ 0 e PL[i] = 0):</u>

$$LFT[i] = LFT[POS[i]] - TT[POS[i]] - TP[POS[i]] \tag{4.8}$$

onde:

LFT[POS[i]] = LFT da operação posterior a i na rota de produção



	TP	TT
P1	7	5
P2	6	3
P3	5	0

FIGURA 4.9: Cálculo do LFT de uma operação

C) Se i é uma operação que opera em paralelo

C.1) LFT das operações em paralelo e que possuem o mesmo tempo de processamento (POS[i] = POS[i+1] e TP[i] = TP [i+1])

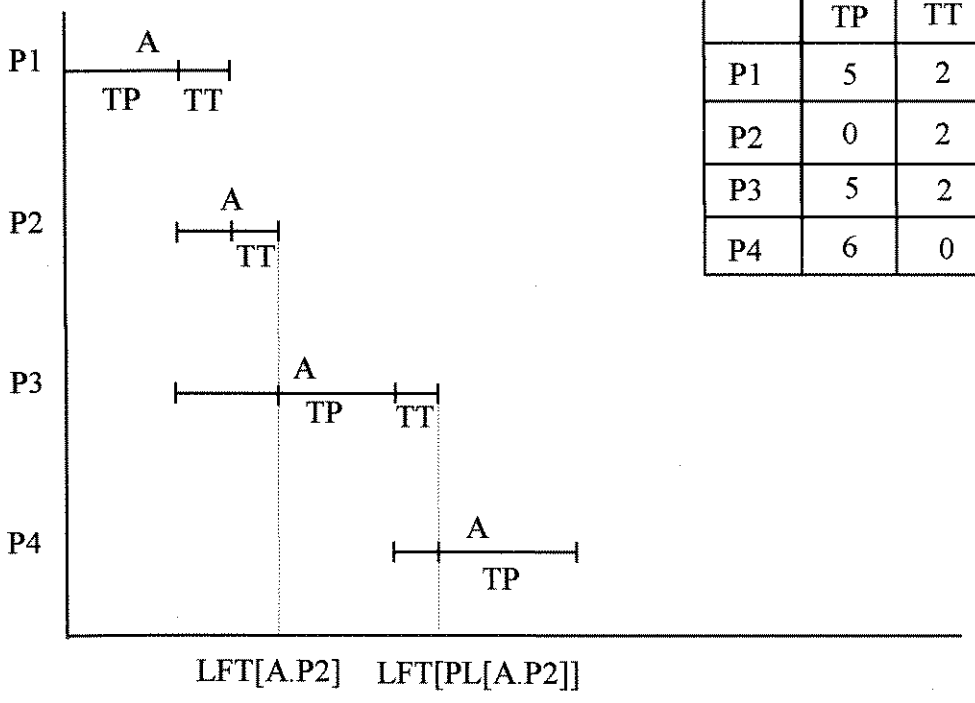
$$LFT[i-1] = LFT[i] \tag{4.9}$$

onde:

LFT[PL[i]] = LFT da operação paralela a i

C.2) LFT das operações em paralelo que possuem tempos de processamento diferentes (POS[i] ≠ POS[i+1] e TP[i] ≠ TP[i+1])

Neste caso, o cálculo do LFT é o mesmo que para as operações que não operam em paralelo visto que a primeira operação descarrega o intermediário para sua paralela que é também a sua posterior na rota de produção.



	TP	TT
P1	5	2
P2	0	2
P3	5	2
P4	6	0

FIGURA 4.10: Cálculo do LFT de operações que operam em paralelo com tempos de processamento diferentes

Depois de determinadas as janelas de todas as operações, o segundo passo é definir quais delas são candidatas a integrarem a seqüência a partir de um determinado nível da árvore de busca, bem como o instante de início mais cedo em que elas podem começar. Este procedimento é executado pela função simulação que será apresentada a seguir.

4.4.3- DETERMINAÇÃO DO CONJUNTO CAN E ATUALIZAÇÃO DO ESTADO DA PLANTA - FUNÇÃO SIMULAÇÃO

O conceito do conjunto CAN de operações candidatas utilizado na estratégia desenvolvida por Campos (1993) para se reduzir o esforço de busca nos casos em que ocorre seqüências de permutação, foi utilizado no desenvolvimento da estratégia proposta. Neste caso, o autor promovia a expansão da árvore considerando-se apenas um subconjunto de tarefas definido como conjunto CAN de tarefas ainda não seqüenciadas. A determinação do conjunto CAN é feita através da análise das restrições temporais originadas pelos limites de tempo das tarefas, determinados por suas janelas de tempo e pela escassez de recursos compartilhados. A idéia era indicar um pré-ordenamento entre as tarefas que utilizam o mesmo processador eliminando temporariamente da árvore aquelas que tornariam infactível o seqüenciamento de tarefas futuras que teriam suas janelas de tempo extrapoladas se fossem alocadas depois daquelas que foram retiradas da árvore. Uma regra proposta por Erschler foi utilizada para se determinar este pré-seqüenciamento entre as tarefas.

No caso da estratégia proposta neste trabalho, a redução do espaço de busca é obtida a partir da análise do conjunto de restrições que compõem o problema (busca orientada por restrições). Não é efetuado aqui um pré-ordenamento entre as operações segundo as restrições das janelas de tempo. Os fatores que determinam quais as operações que irão compor este conjunto num determinado nível da árvore de busca levam em conta a precedência tecnológica existente entre elas e a disponibilidade do equipamento em que é efetuada a operação. Um vetor denominado can[i] é utilizado para se determinar o conjunto CAN de operações não alocadas. Os elementos deste vetor assumem os valores de zero se a operação não é candidata num determinado nível da árvore, can[i] = 0, e um se ela é candidata, can[i]=1. Utilizando-se esta estratégia, observa-se uma redução significativa do domínio de soluções pesquisadas.

Para se definir se uma dada operação está alocada, é utilizado um vetor denominado PS[i], onde i varia de zero até 45, que é o número total de operações que se deseja alocar. Os

elementos deste vetor inicialmente são definidos como zero, e a medida que as operações são seqüenciadas PS[i] assume o valor da ordem em que a operação i foi alocada. Por exemplo, PS[24] = 7 significa que a operação 24 foi a sétima operação alocada.

Uma operação só é candidata se a sua antecessora na rota de produção já houver sido executada, ou seja, se PS[ANT[i]] # 0 tem-se que can[i] = 1, e se o seu processador não estiver ocupado por outra operação que utilize a mesma máquina, e cujo material processado esteja esperando para ser transferido para a operação subseqüente no instante em que se deseja processar esta operação candidata. Assim, se PRO[j] = PRO[i] e PS[j] # 0 e PS[POS[j]] # 0 implica que can[i] = 1, onde j é uma operação já alocada no processador que executa i.

Porém, uma operação que utiliza um processador ocupado não estará proibida se houver possibilidade de ela ser alocada em um intervalo de tempo anterior ao início da operação que proíbe o processador. Isto é possível uma vez que o algoritmo permite que seja escolhido para o início de uma operação, um outro instante de tempo diferente daquele calculado como o primeiro instante de tempo factível. Desta forma, pode ser delimitado um intervalo de tempo ocioso antes do início da operação já seqüenciada, suficiente para alocar a operação candidata mais o set up entre estas operações. Assim, é possível a alocação de uma operação cujo processador está ocupado desde que seu start time seja anterior ao início da operação que ocupa o processador, sendo que o algoritmo se encarrega de proibir o seu início em um instante posterior ao processamento da operação alocada, até que esta libere o processador. A escolha de um start time diferente daquele calculado pelo recorte das janelas está sujeita a uma serie de fatores que devem ser observados antes que este instante seja aceito como factível para a alocação da operação candidata. Estas condições serão descritas posteriormente no item referente a verificação da consistência do instante de início escolhido.

Além destes fatores, a condição de instabilidade da operação que acabou de ser seqüenciada e se existe ou não operações em paralelo com ela, também são parâmetros utilizados para definir quais as operações que serão incluídas no conjunto CAN. Assim sendo, se a última operação que foi alocada é uma operação instável, a única operação que aparece como candidata para sucede-la será a próxima operação na sua rota de produção. Assim, se SB[j] = 1, então can[POS [j]] = 1 e can [i] = 0 com i # POS[j], onde j é a última operação alocada. Da mesma maneira, se esta operação estiver em paralelo com outra, somente a paralela aparecerá como candidata, ou seja, se PL[j] # 0, então can[PL[j]] = 1 e can[i] = 0 com i # PL[j], onde j é a última operação alocada.

Um cuidado adicional deve ser tomado no que se refere as bateladas de um mesmo produto. Como cada batelada é definida como se fosse uma tarefa diferente, o algoritmo estabelece uma variável de controle que proíbe que uma operação da segunda batelada de um determinado produto seja candidata antes que a operação correspondente da primeira batelada tenha sido executada, e o material processado tenha sido transferido para a próxima operação para que o processador seja liberado de forma que ele possa processar i, ou seja, a próxima operação de B_ANT[i] também tenha sido executada. Assim, se B_ANT[i]# 0, então i só é candidata se PS[POS[B_ANT[i]]] # 0.

É importante ressaltar que todas as operações têm seus instantes de início calculados em todos os níveis da árvore, sendo proibidas aquelas que não pertencem ao conjunto CAN de tarefas não alocadas, ou seja, se can[i] = 0. Estas operações proibidas não aparecem como opção de escolha quando se vai fazer uma nova alocação.

Definido o conjunto CAN, a operação a ser alocada é escolhida entre aquelas pertencentes ao conjunto, e um instante de início é escolhido para ela. O próximo passo, então, é se certificar se o instante de início dado é realmente factível para esta operação. Com este fim, é executada a **função time_consistency**, que só dá continuidade ao processamento se o valor escolhido for factível. Depois de efetuados todos os testes com um instante de início factível escolhido para a operação candidata, o algoritmo retorna para a função simulação onde o estado da planta é atualizado e a operação candidata é aceita na seqüência com o seu PS[i] recebendo o valor da ordem em que ela foi seqüenciada. É acrescentado também, o consumo da operação i no perfil de recursos compartilhados **(Função share).** Este procedimento é descrito com detalhes em Rodrigues (1992).

4.4.4 - TESTE DE FACTIBILIDADE DO INSTANTE DE INÍCIO ESCOLHIDO - FUNÇÃO TIME_CONSISTENCY

O algoritmo permite que seja escolhido um outro instante para o início de uma dada operação, e antes de se calcular os novos EBT's das operações não alocadas, deve-se se verificar a factibilidade deste instante de início escolhido para a operação i que acabou de ser seqüenciada, se este instante for diferente daquele calculado como o início de sua janela. Para cada caso, testes diferentes são efetuados. Dentro desta função, a operação candidata recebe a denominação de "op"

A seguir serão apresentadas as diversas possibilidades para estes procedimentos, e todos os passos envolvidos no teste de factibilidade do instante de início escolhido para a operação que se quer seqüenciar estão descritos abaixo.

A) A operação candidata está antes de uma operação já alocada no processador (Sbaux[op] = 1) e a janela de uma operação posterior na rota torna infactível o instante escolhido

O primeiro caso analisado se dá quando o instante de início escolhido é posterior ao calculado para a janela da operação candidata, e se verifica não só a possibilidade da alocação da operação candidata neste instante como também o posicionamento possível para as operações posteriores na rota de produção a partir do instante escolhido. Neste caso, um teste de consistência é efetuado para se certificar se não haverá conflito de recursos ou de ocupação do processador, no caso de já existirem outras operações alocadas neste mesmo equipamento em um instante de tempo mais tarde. Além disto, é verificada também em que instante, a partir daquele escolhido para a operação em questão, as próximas operações em sua rota de produção têm possibilidade de serem alocadas. Se o posicionamento de uma destas operações só for possível em um instante tal que obrigue o material processado na operação candidata a esperar no processador até que o próximo equipamento esteja disponível, de forma que esta espera provoque um conflito de alocação entre a operação que se deseja alocar e aquela anteriormente alocada em um instante de tempo posterior no seu processador, o instante de início escolhido para a operação candidata é rejeitado e uma mensagem avisa que a operação não pode ser alocada no instante escolhido. Nestes casos, o algoritmo retorna ao passo anterior e uma alocação em um novo instante de início pode ser tentada.

Os instantes em que as próximas operações da rota podem ser iniciadas é conhecido pelo cálculo de suas janelas a partir do instante de início fornecido, calculadas pela função start_time chamada aqui. Assim sendo, se se desejar alocar uma operação antes de uma outra já alocada no mesmo processador, e se for necessário que o material por ela processado aguarde neste processador até que a próxima operação da rota de produção possa ser executada, a espera deve ser no máximo igual à "folga" delimitada pelo término do set up entre a operação candidata e a operação alocada, e o início desta operação alocada.

A figura abaixo exemplifica este procedimento. Os valores ao lado das operações candidatas representam os instantes de início mais cedo para cada uma destas operações. A

região marcada com "X" representa uma indisponibilidade de tempo. A operação 1 utiliza o mesmo processador da operação 20, já alocada na planta em t = 67. O tempo de processamento de op1 é igual a 4 e tempo de transferência igual a 1. O set up entre a operação 1 e a operação 20 é igual a 17. O tempo de processamento da operação 2, sucessora de 1 em sua rota de produção, é igual a 6 e o tempo de transferência é igual a 2. Supondo que seja escolhido t = 20 como instante de início para a operação 1, então:

candidatas: op1 = 10; op10 = 25; op15 = 8; op21 = 80.

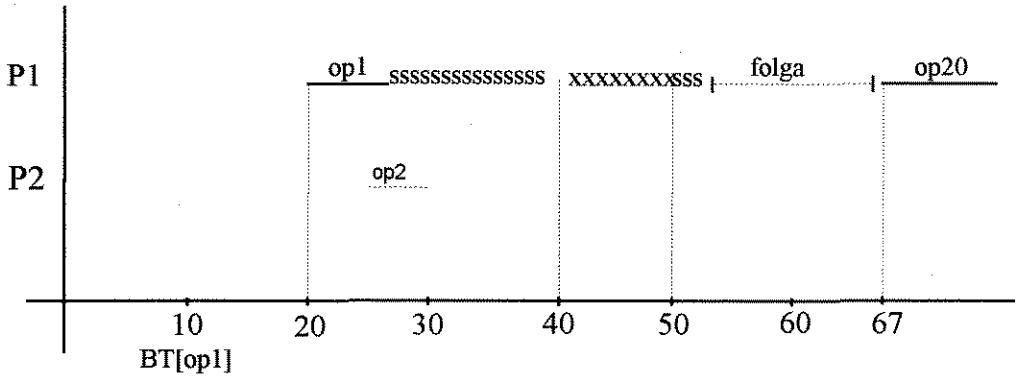


FIGURA 4.11: Teste de factibilidade em função da janela das posteriores - Instante factível

Pela figura acima, observa-se que este instante é factível pois permite a alocação da candidata op1 sem que haja conflito entre esta operação e a operação 20 no que se refere a ocupação do processador P1. A folga existente entre o término do set up entre op1 e op20 e o início de op20 é igual a 15. A operação 2 não está alocada mas foi representada na figura para se demonstrar que é possível sua alocação em t = 24, após a alocação da op1 em t = 20. Assim, o instante de início escolhido (t = 20) é aceito para a operação 1

Supondo agora que o instante de início escolhido para operação 1 seja t = 35. Observa-se que neste instante não é mais possível a alocação desta operação no processador 1, ainda que haja tempo suficiente, entre t = 35 e o início da operação 20, para o processamento de op1 mais o set up entre as duas operações. Porém, a alocação da operação 2 só é possível após o término da indisponibilidade de tempo, o que faz com que a operação 1 fique ocupando P1, no mínimo, até t = 51, o primeiro instante em que é possível ocorrer a transferência do material processado para P2. Só após este instante, é que o set up entre estas duas operações pode ser iniciado. Porém, a folga existente entre a liberação do processador por op1 e o início de op20 é de apenas 16 unidades de tempo enquanto que o set up entre elas é de 17, uma unidade de

tempo a mais que o intervalo de tempo disponível. Assim sendo, t = 35 é rejeitado e uma mensagem é enviada avisando que não é possível a alocação no instante escolhido. Na figura abaixo pode-se observar o caso descrito acima.

candidatas: op1 = 10; op10 = 25; op15 = 8; op21 = 80.

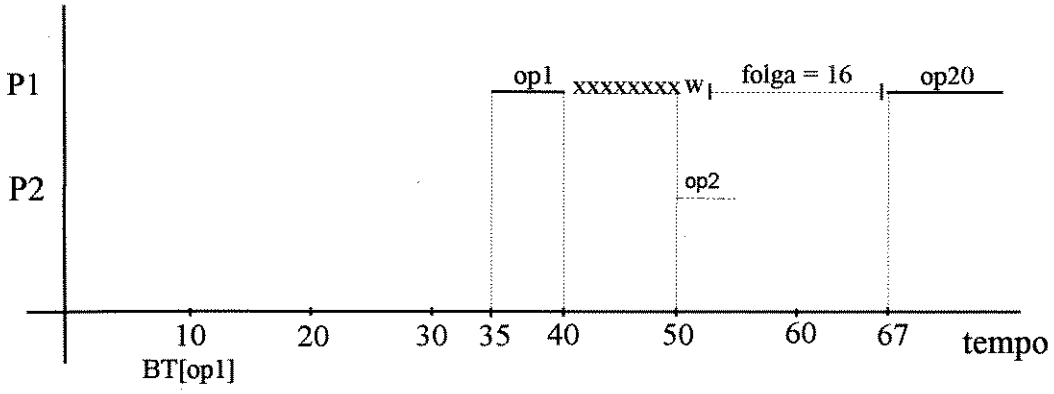


FIGURA 4.12: Teste de factibilidade em função da janela das posteriores - Instante infactível

Este teste é efetuado não apenas para a próxima operação daquela que se deseja alocar, mas também para as próximas operações da sua rota de produção, garantindo-se assim que não haverá interferência das operações posteriores no seqüenciamento da operação candidata. Um exemplo mais detalhado está no exemplo 7, no item 4.5 referente a exemplos

B) A operação anterior da operação candidata na rota de produção está alocada antes de outra operação no processador que a faz, (SBaux[ANT[op]]=1), e o instante escolhido torna infactível o processamento desta anterior no instante em que ela foi alocada.

O mesmo exemplo anterior pode ser utilizado para este caso onde o instante de início escolhido para uma dada operação é infactível em virtude de interferir na ocupação do processador da operação anterior em sua rota de produção. No exemplo acima, supondo que a operação 1 já esteja alocada em t = 20 mas o instante de início escolhido para operação 2 seja t = 50. Se op1 não for instável, este deslocamento é possível. Porém, da mesma forma que no exemplo acima, não haveria tempo suficiente entre a desocupação de P1 pela operação 1 e o início da operação 20 para se realizar o set up entre estas duas operações. Desta maneira, o

instante de início t = 50 para a operação 2 é rejeitado e uma mensagem avisa que operação infactibilizou o instante escolhido. O exemplo 8 no item 4.5 ilustra este caso.

É importante lembrar que a cada instante de início escolhido, o teste de factibilidade com respeito a recursos é também efetuado, sendo recusado se houver violação desta restrição.

C) <u>Instante de início escolhido é menor que o EBT</u>

O algoritmo permite também que um instante de tempo anterior ao EBT calculado seja escolhido. Se isto ocorrer, pode haver violação da oferta de recursos compartilhados, conflito de ocupação do processador ou violação de indisponibilidade de tempo. No caso de ocorrer violação de recursos, o algoritmo permite que a alocação seja efetuada. Porém se ocorrer conflito de ocupação do processador ou de indisponibilidade de tempo, a alocação no instante escolhido é proibida e o algoritmo retorna o valor inicial do instante de início para que uma nova alocação seja tentada.

A verificação de que se a partir do instante de início escolhido, o processamento de op invade uma indisponibilidade de tempo é efetuada pela função fim_de_semana que é chamada para fazer esta verificação.

Na figura abaixo, a notação "L" representa o limite máximo de demanda de um recurso compartilhado qualquer, que pode ser uma utilidade. A operação 36 não está alocada e é candidata a ser seqüenciada. Seu instante de início é calculado de forma que seja respeitada a limitação na oferta de recurso compartilhado, e é este valor que aparece no menu de escolha das operações candidatas. Para o exemplo apresentado, este instante equivale a t = 30, uma vez que a operação 7 consome toda a oferta de recurso disponível. Observa-se na figura que a operação 36 pode ser adiantada, sem conflito de ocupação do processador P1, até t =20, o instante em que termina o set up entre op1 e op36. Neste caso, se for tentada a alocação em t = 20, o algoritmo aceita este valor, sendo violada a restrição de oferta do recurso. Obviamente, se o instante de tempo escolhido para a op36 for menor que 20, o valor não é aceito e o algoritmo informa, através de uma mensagem, que o instante escolhido é infactível por rota, processador ou indisponibilidade de tempo, retornando no menu o valor anterior da janela, que neste caso é igual a 30.

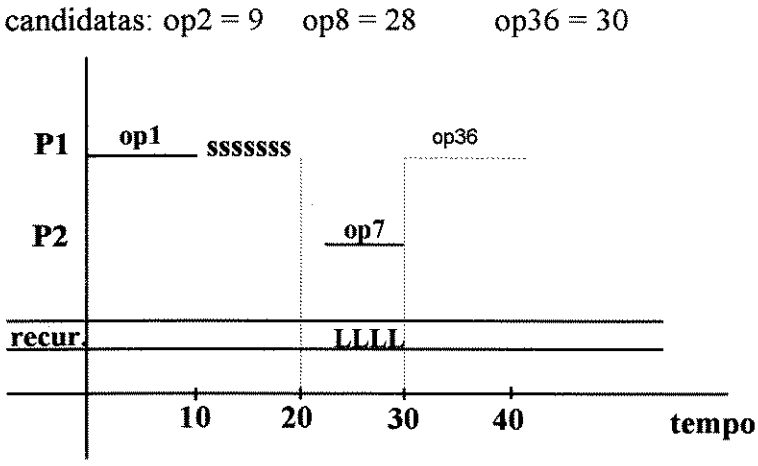


FIGURA 4.13: Teste de factibilidade por rota, processador ou indisponibilidade de tempo

O exemplo 4 do item 4.5 demonstra este procedimento.

D) Condição de instabilidade do material processado

Neste teste, também é verificada a condição de instabilidade da operação alocada. Se ela for instável e, a partir do start time escolhido, não for possível alocar a próxima operação logo após o seu término, o algoritmo proíbe sua alocação neste instante. Da mesma forma, se for escolhido para a posterior desta operação instável, um instante diferente daquele calculado pela sua janela, o algoritmo não aceitará esta alocação visto que ela deve ser iniciada logo após o término da primeira.

Na figura abaixo apenas a operação 12 está alocada e a operação instável op1 é candidata a ser seqüenciada. A operação 2 é sucessora de op1 em sua rota de produção. O tempo de processamento de op1 é de 8 unidades de tempo e seu tempo de transferência é de 1. O set up entre a operação 12 e a operação 2 no processador 2 é igual a 10. Observa-se que a operação 1 pode ser alocada, no mínimo, em t = 13. Se um valor menor for escolhido para o início de op1, a operação 2 não poderá ser alocada logo em seguida de seu processamento, em função da presença da operação 12 ocupando o processador 2. Isto não pode ser permitido visto que op1 é instável. Desta forma, o algoritmo rejeita o valor , e retorna o valor da janela (t =13). Da mesma maneira, se op1 for alocada em um dado instante "t", a operação 2 só poderá

ser alocada em t + 8 não sendo permitido nenhum outro valor. No item 4.5, o exemplo 2 ilustra este cálculo.

candidatas: op1 = 13.

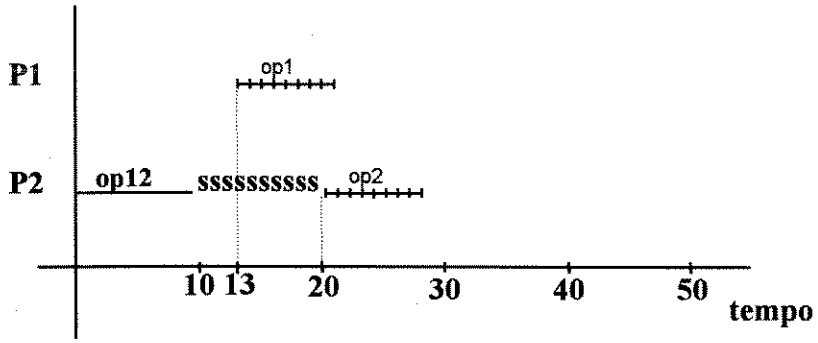


FIGURA 4.14: Teste de factibilidade por instabilidade do material processado.

E) Operações em paralelo

Uma outra proibição também é feita quanto as operações que operam em paralelo que devem possuir o mesmo instante de início não sendo permitido o deslocamento de seu start time, ou seja, se op2 opera em paralelo com op3 e op2 foi alocada em no instante "t", só é permitido para op3 ser alocada em "t". Se qualquer outro valor for tentado, o algoritmo o rejeitará.

F) Liberação do processador

Também não é permitida a alocação de op em um instante de tempo posterior ao processamento da última operação que utilizou o seu processador se a próxima operação desta última em sua rota de produção ainda não houver sido alocada, pois, desta forma, o processador ainda está ocupado com produto intermediário não sendo possível sua ocupação por outra operação. Para exemplificar este teste, tome-se como base o mesmo exemplo do caso A. Se for escolhido para o início da operação 1 um valor maior que 67, start time de op20, o algoritmo não permitirá a alocação até que a operação posterior à esta operação seja seqüenciada, liberando o processador 1.

G) Indisponibilidade de Matéria-Prima

Se o início de uma janela for definido por limitação no fornecimento de matéria-prima, o algoritmo proíbe que um instante de início anterior ao início desta janela seja escolhido. Se um isto acontecer, é emitida uma mensagem avisando que a tarefa não pode ser relaxada.

H) Seqüência Infactível

Uma seqüência é infactível se a alocação de uma operação infactibiliza o posicionamento de uma outra que já havia sido alocada. O algoritmo não permite que nenhuma operação seja desalocada e, por isso, uma mensagem informa que a seqüência é infactível e o programa é terminado. O exemplo 9 no item de exemplos ilustra este caso.

Após efetuado o teste de consistência do instante de início da última operação alocada, o algoritmo retorna a função simulação onde, após atualizado o estado da planta, é chamada a **função start_time** para ser iniciado o recorte das janelas de tempo das demais operações, em função do start time escolhido para esta última. O recorte das janelas é feito para todas as operações não alocadas, (PS[i]=0), com i variando de 1 a 45.

4.4.5 - DEFINIÇÃO DO INSTANTE DE INÍCIO MAIS CEDO DAS OPERAÇÕES NÃO ALOCADAS - FUNÇÃO START_TIME

A principal finalidade do algoritmo proposto é calcular os instantes de início mais cedo em que todas as operações podem ser iniciadas a partir de uma dada operação alocada, de maneira que estes instantes sejam factíveis considerando-se todos os fatores envolvidos no seu processamento.

Sete são os fatores utilizados para definir os instantes factíveis de início das operações ainda não alocadas, a saber:

1- Instante em que a batelada anterior da mesma tarefa estiver terminada

2- instante que a operação anterior da mesma tarefa estiver terminada

3- instante de liberação do processador responsável pela execução da operação i

4- instante de liberação do processador responsável pelo processamento da próxima operação da mesma tarefa, se i for instável

5-instante mais cedo em que o perfil de oferta dos recursos é capaz de acomodar a demanda da operação e, se for instável, de sua sucessora na rota de produção,

6- disponibilidade de matéria-prima, dada pelas janelas de tempo

7- instante de início e fim das indisponibilidades de tempo decorrentes da jornada de trabalho estabelecida para o funcionamento da planta

Todas estas considerações que influenciam no cálculo das janelas das operações não seqüenciadas estão descritos a seguir. A operação i que se está calculando a janela é denominada de op internamente nesta função (op = i).

A) Cálculo do instante de início das operações não alocadas em função da batelada anterior do mesmo produto

O primeiro passo quando se inicia o cálculo da janela de uma dada operação não seqüenciada, é se certificar se existe uma batelada anterior do mesmo produto a qual ela faz parte, (B_ANT[i] # 0). Se existir, esta operação só pode ser iniciada após a mesma operação da batelada anterior ter sido processada e o material processado ter sido transferido para a próxima operação liberando o processador. Assim, tem-se:

$$BT[op] = BT[B\_ANT[op]] + TT[ANT[B\_ANT[op]]] + TP[B\_ANT[op]] + TT[B\_ANT[op]] + SU[B\_ANT[op]][TAR[op]] \quad (4.10)$$

onde:

BT[op] = EBT de op

BT[B_ANT[op]] = EBT da mesma operação na batelada anterior de op

TT[ANT[B_ANT[op]]] = tempo de transferência da operação anterior da mesma operação na batelada anterior de op

TP[B_ANT[op]] = tempo de processamento da mesma operação na batelada anterior de op

TT[B_ANT[op]] = tempo de transferência da mesma operação na batelada anterior de op

SU[B_ANT[op]][TAR[op]] = set up entre a mesma operação na batelada anterior de op e op

B) Cálculo do instante de início da operações não alocadas via rota de produção

O segundo procedimento executado diz respeito ao cálculo do novo instante de início mais cedo (EBT) da operação op, definido em função da rota de produção. A relação que promove este cálculo é descrita abaixo:

$$st\_op = BT[ANT[op]] + TT[ANT[ANT[op]]] + TP[ANT[op]] \quad (4.11)$$

onde:

st_op = EBT de op

BT[ANT[op]] = instante de início da operação anterior de op na rota de produção

TT[ANT[ANT[op]]] = tempo de transferência da operação anterior à anterior de op

TP[ANT[op]] = tempo de processamento da operação anterior de op

Com este dois valores obtidos para o início da janela de op, tem-se que:

$$st\_op = máx\ (BT[op], st\_op) \quad (4.12)$$

Em seguida, deve-se verificar, a partir de st_op, em que instante o processador que executa op está disponível para uso.

C-) Cálculo do instante de início via liberação do processador que executa op

Um vetor que define a seqüência por equipamento, é utilizado para saber se uma dada operação não seqüenciada "op", cuja janela está sendo recortada, pode ser alocada em um instante de tempo factível entre duas operações que façam parte da seqüência do equipamento que a executa.

O vetor opp é construído de forma a guardar a seqüência no processador de op onde cada posição em opp, (opp[i]), representa todas as operações já seqüenciadas neste processador. Assim, se i já foi seqüenciada (PS[i] # 0), e o processador de i é igual ao processador de op (PRO[i] = PRO[op]), então opp[j] = i com i variando de 1 até o número total de operações. Para a operação op é feita uma pseudo alocação atribuindo-se para ela um valor grande de PS[op] para que ela possa fazer parte do vetor apesar de, a rigor, ela ainda não estar seqüenciada. O número máximo de operações que podem ser alocadas no processador é igual a número total de tarefas (T = 10), ou seja, opp[i] pode ter no máximo 10 operações alocadas.

Depois de construído, o vetor é então ordenado segundo a ordem crescente dos start times das operações que o compõem. O start time inicialmente atribuído para op é aquele calculado em função da rota de produção e da janela de sua batelada anterior, se houver, dado pela equação 4.12. A sua posição no vetor é guardada em uma variável denominada id_op para que se busque um EBT a partir de um valor inicialmente factível. Por exemplo, se op está ordenada na 3° posição do vetor, só será testada a factibilidade de seu EBT com respeito a ocupação do processador entre a 2° e a 4° operações alocadas no processador, pois um EBT calculado entre a 1° e a 2° já é, a princípio, infactível em função da sua rota de produção.

Definido o vetor seqüência por processador, tem-se que a equação que define o instante de início para op, respeitando-se o instante em que o processador que a executa está liberado, é escrita da seguinte forma, com i variando de id_op até o número máximo de posições (i = 10):

$$pro\_lib = BT[opp[i-1]] + TP[opp[i-1]] + TT[opp[i-1]] + TT[ANT[opp[i-1]]] \quad (4.13)$$

onde:

pro_lib = instante de liberação do processador de op

BT[opp[i -1]] = instante de início da operação que ocupa o processador de op

TP[opp[i -1]] = tempo de processamento da operação opp[i -1]

TT[opp[i -1]] = tempo de transferência da operação opp[i -1]

TT[ANT[opp[i -1]]] = tempo de transferência da operação anterior a opp[i -1] na rota de produção.

O instante que o processador está liberado depende não somente do completion time da tarefa que ele processou como também do instante de início da próxima operação da mesma tarefa na rota de produção, se ela já estiver seqüenciada, ou seja, PS[POS[opp[i -1]]] # 0. Este instante não é necessariamente igual ao completion time de opp[i -1] uma vez que, não existindo armazenagem intermediária, o material processado aguarda no próprio equipamento até que a próxima operação da mesma tarefa seja seqüenciada. A partir daí, o equipamento está liberado para o processamento de uma outra operação. Assim sendo, o instante de liberação do equipamento é deslocado até o instante de início de POS[opp[i -1]]. Assim; pro_lib passa a ser:

$$pro\_lib = BT[POS[opp[i-1]]] + TT[opp[i-1]] \tag{4.14}$$

onde:

BT[POS[opp[i -1]]] = instante de início da operação posterior a opp[i -1] na rota de produção

TT[opp[i -1]] = tempo de transferência de opp[i -1]

A este instante deve ser adicionado o setup entre a opp[i -1] e op para que seja definido o start time de op. Desta forma, pro_lib passa a ser:

$$pro\_lib = pro\_lib + SU[opp[i-1]][TAR[op]] \tag{4.15}$$

onde:

SU[opp[i -1]][TAR[op]] = setup entre a operação opp[i -1] e a tarefa a que pertence op

Definido pro_lib pela equação acima, deve-se verificar se, após adicionado o setup entre as duas operações, não ocorreu invasão em nenhuma indisponibilidade de tempo. Se isto acontecer, o setup deve ser interrompido no início da indisponibilidade para ser completado após o seu término. Desta forma, o período de duração da indisponibilidade de tempo em questão deve ser adicionado ao instante de liberação do processador. A figura abaixo exemplifica este procedimento.

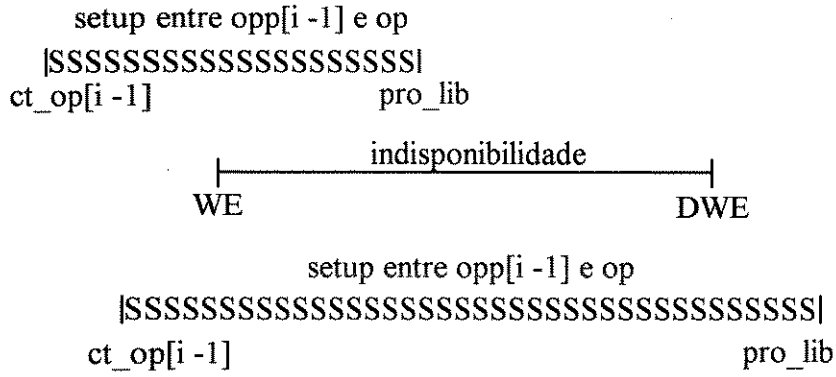


FIGURA 4.15: Set up invade indisponibilidade de tempo

Chamando de w[0] é a diferença entre pro_lib e WE, e w[1] é a diferença entre DWE e ct_op[opp[i -1]], se w[0] e w[1] forem maiores que zero, o setup entre opp[i -1] e op invadiu a indisponibilidade devendo ser interrompido para continuar após o fim da indisponibilidade. O novo pro_lib passa então a ser pro_lib+(DWE - WE), ou seja:

$$pro\_lib = pro\_lib + (DWE - WE) \tag{4.16}$$

onde:

DWE = instante de término da indisponibilidade de tempo invadida

WE = instante de início da indisponibilidade de tempo invadida

Com as informações do instante de início de op determinados pelo instante de início da batelada anterior e pela rota de produção dados pela equação 4.12, e pelo instante de liberação do processador, dado pelas equações 4.13 a 4.16, é possível, então, definir um primeiro valor que, a princípio, será o início da sua janela, considerando-se apenas a operação anterior a op no vetor opp, ou seja, opp[i -1]. Este instante será, então o máximo entre estes dois valores, ou seja:

$$st\_op = \max(st\_op, pro\_lib) \tag{4.17}$$

A figura abaixo dá uma representação esquemática de todas as variáveis envolvidas no cálculo do instante de liberação do processador

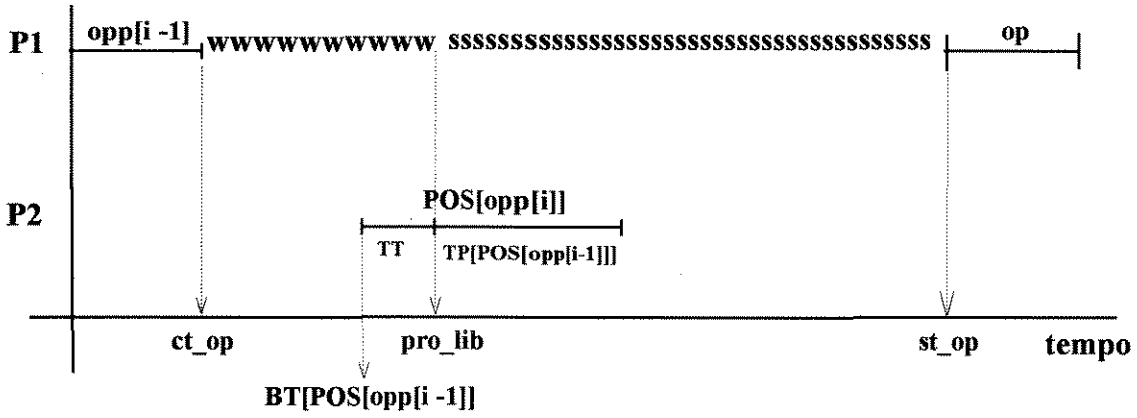


FIGURA 4.16: Representação das variáveis envolvidas no cálculo de pro_lib

D) Oferta de recursos (Função share) e indisponibilidade de tempo - (Função fim_semana)

Após determinado st_op, alguns fatores devem ser observados antes de se determinar o real instante de início de op. O primeiro diz respeito ao perfil de oferta de recursos compartilhados, que deve ser capaz de acomodar a nova demanda referente à esta operação sem que, em qualquer instante, a oferta de nenhum destes recursos seja violada. Desta forma, é definido para $t \geq st\_op$, o instante mais cedo em que é possível alocar a operação com vistas à oferta destes recursos. O procedimento para a determinação deste instante é encontrado em Rodrigues (1992). Se o teste com relação a recursos acarretar uma mudança no instante de início anteriormente calculado, um novo teste com relação à invasão de indisponibilidades de tempo é efetuado. Se houver nova mudança no instante de início em função desta indisponibilidade, novamente é testado recursos compartilhados. O procedimento fica em loop até que se obtenha um instante de início factível tanto com relação a recursos compartilhados como com relação a indisponibilidades de tempo.

No procedimento onde se verifica se houve invasão de indisponibilidade de tempo, deve-se se certificar se o intervalo de tempo disponível entre o st_op calculado, observando-se os fatores descritos acima, e o início da próxima indisponibilidade é suficiente para a execução da operação e da operação seguinte da mesma tarefa, se a primeira for instável. Se este intervalo não for suficiente, o início da operação deve ser deslocado e o novo st_op será o instante de término desta indisponibilidade. Este shift de tempo se faz necessário uma vez que a operação não pode ser interrompida após o início de sua execução.

Estas situação é resolvida utilizando-se um artificio de cálculo similar ao desenvolvido no cálculo de pro_lib quando o set up entre opp[i -1] e op passa por uma destas indisponibilidades. O esquema abaixo representa estes cálculos:

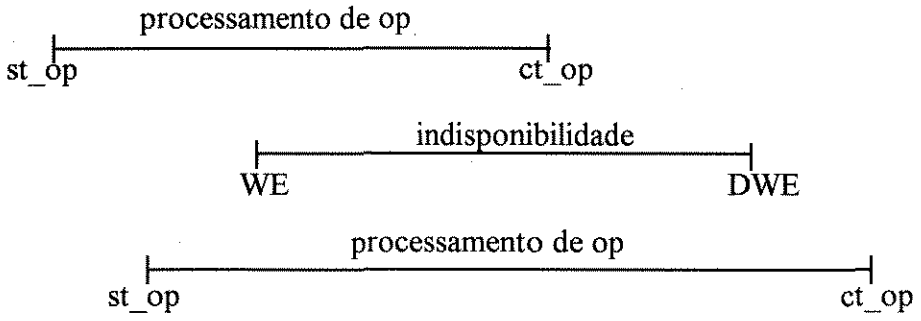


FIGURA 4.17: Processamento invade indisponibilidade de tempo

Chamando de w[0] a diferença entre ct_op e WE, e por w[1] a diferença entre DWE e st_op, tem-se que, se w[0] e w[1] forem maiores que 0, o processamento de i+1 invadiu a indisponibilidade de tempo e o novo start time passa a ser DWE. Desta forma st_op é dado por:

$$st\_op = máx \{ st\_op; DWE \} \tag{4.18}$$

E) Verificação de conflito de ocupação do processador de op com relação a opp[i+1]

Após efetuado os cálculos acima, deve-se verificar se existe uma operação alocada no processador em um instante de tempo mais tarde, ou seja, se opp[i +1] # 0 e, se existir, certifica-se se o instante de início calculado para op com relação a opp[i -1], será também factível com relação a opp[i +1]. Em outras palavras, se o intervalo de tempo disponível entre o st_op e o BT[opp[i+1]] é suficiente para comportar o tempo que op ocupa o processador mais o setup entre op e opp[i +1], ou seja:

$$st\_op + TP[op] + TT[op] + TT[ANT[op]] + SU[op][TAR[opp[i+1]]] \leq BT[opp[i+1]] \tag{4.19}$$

onde:

BT[opp[i+1]] = instante de início da operação opp[i+1]

Se isto se confirmar, a operação, a princípio, pode ser alocada entre opp[i -1] e opp[i+1] e st_op é definido como o início da janela da operação op. Se não, um incremento em i no vetor opp é feito e o cálculo se repete para o intervalo de tempo compreendido entre opp[i +1] e opp[i +2].

No exemplo a seguir, as operações 1, 20, 35 e ANT[op] já estão alocadas. É iniciado, então, o recorte da janela da operação não seqüenciada op. O primeiro passo é determinar o instante de início via rota de produção para esta operação, dado pelo fim do processamento da operação anterior a op em sua rota de produção, ANT[op]. É este valor que será utilizado para determinar a sua posição no vetor seqüência do processador P2, supondo que a operação op não possua batelada anterior. No exemplo dado, este instante é igual a 40. O vetor opp é construído segundo a ordem crescente dos start times das operações que o compõem e, inicialmente, possui a seguinte seqüência:

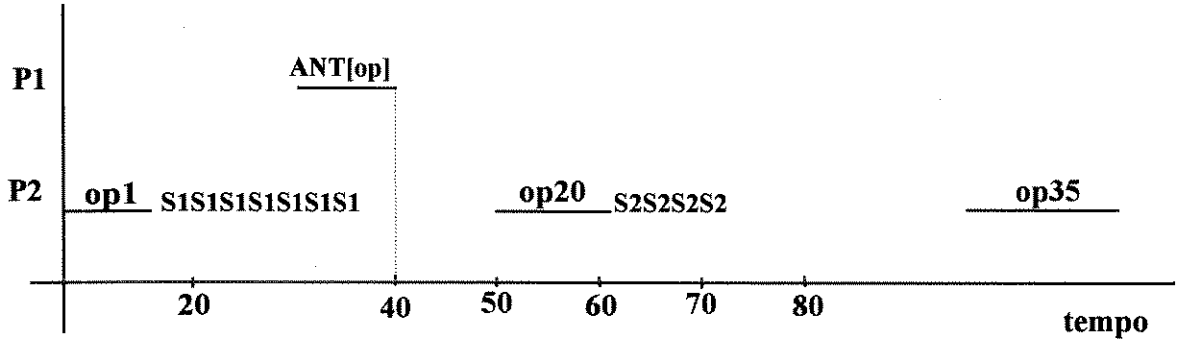
{ op1, op, op20, op35 }

Verifica-se a partir daí se é possível alocar op entre op1 e op20 levando-se em conta o set up entre op1 e op somado ao tempo de ocupação de op no processador P2 mais o set up entre op e op20. Supondo-se que o intervalo de tempo disponível entre o término de op1 e o início de op20 não seja suficiente para alocar op, esta operação troca de posição com op20 na seqüência do processador P2 e o teste é repetido para o intervalo de tempo compreendido entre o término de op20 e o início de op35. No exemplo abaixo, este posicionamento se confirma e o instante de início mais cedo calculado para op será 70. Se neste intervalo também não fosse suficiente para a alocação de op, seu instante de início mais cedo será, a princípio, definido pelo término de op35 somado ao set up entre op35 e op. A seqüência no vetor opp fica, então:

{ op1, op20, op, op35 }

O esquema abaixo exemplifica o cálculo da janela de op entre duas operações já seqüenciadas no seu processador.

1- Seqüência em P2 inicialmente composta por op1, op20 e op35:



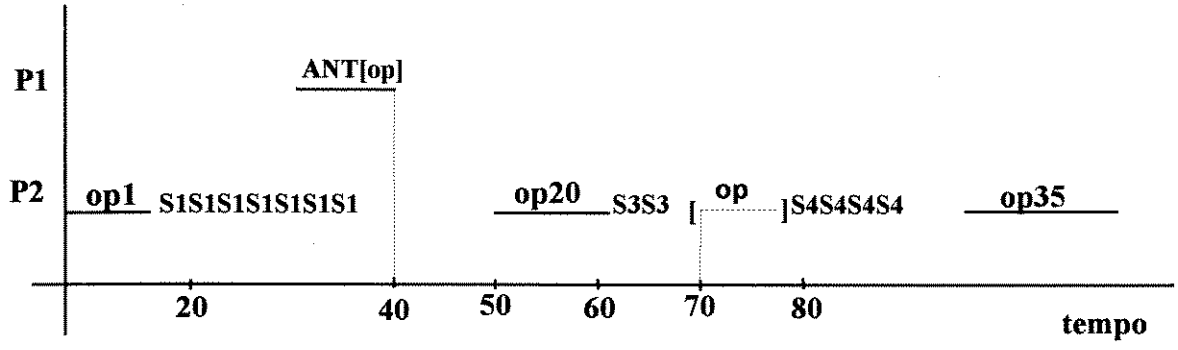
onde: ANT[op] = operação anterior a op em sua rota de produção

S1 = set up entre op1 e op20

S2 = set up entre op20 e op35

Seqüência em P2 { op1; op20. op35 }

2- Procura onde é possível alocar op:



onde: S3 = setup entre op20 e op

S4 = setup entre op e op35

Seqüência em P2 {op1, op20, op, op35}

FIGURA 4.18: Construção da seqüência por processador

F) Recálculo da janela de op em função das próximas operações na rota de produção - Retorno à função simulação

Terminados os cálculos descritos acima, o algoritmo retorna a função simulação, onde é incrementado o índice das operações. Inicia-se, então, o recorte da janela da próxima operação na rota de produção, se houver, ou da primeira operação de uma outra tarefa. Todos os cálculos são então repetidos para esta nova operação.

A partir do valor calculado para as próximas operações de op na rota de produção, o valor do EBT que havia sido calculado para ela pode ainda ser modificado. Esta modificação pode se dá em função do posicionamento do EBT de op em relação a outras operações previamente alocadas no processador que a executa, ou devido a condição de instabilidade do material processado por op, ou em função do EBT calculado para a operação paralela de op, se existir. Estas considerações estão descritas a seguir:

Se o EBT que foi calculado para op ficar posicionado antes de alguma operação previamente alocada em seu processador, (Sbaux[op] = 1), e o EBT calculado para a sua posterior na rota de produção seja tal que obrigue a operação op a aguardar no processador por um período maior que a folga existente entre op e opp[i+1], o algoritmo decrementa o índice das operações em 1 e recalcula a janela de op a partir de um instante qualquer localizado depois do EBT de opp[i + 1], repetindo todos os cálculos para esta e as próximas operações até que um instante realmente factível seja encontrado. O exemplo 6 demonstra o recálculo da janela de op em função do EBT obtido para operações posteriores.

Se o material processado em op for instável, (SB[op] = 1), o instante obtido deve ser factível não só para esta operação como também para sua posterior na rota de produção, pois, nestas condições, haveria degradação do material processado se o mesmo tivesse que aguardar no equipamento até que o processamento de sua sucessora seja possível. Em vista disto, após calculada a janela da operação posterior a op, o algoritmo verifica se o valor encontrado para esta operação somado ao tempo de transferência de op, é maior que o completion time de op. Se isto se confirmar, o EBT de op deve ser deslocado, até que o seu completion time coincida com o início do processamento da sua operação posterior na rota de produção. Assim, o novo EBT de op passa a ser, no mínimo, EBT[POS[op]] - TP[op] - TT[ANT[op]]. O algoritmo então decrementa o índice das operações em 1 e é calculado um novo start time para op a partir deste valor mínimo, determinando, se necessário, um novo instante de início. A verificação da factibilidade deste instante é efetuada pela função start_time repetindo-se o procedimento até que seja obtido um instante factível para as duas operações. Estes cálculos estão apresentados no exemplo 2.

No caso de op possuir uma paralela que não tenha o mesmo tempo de processamento, como é o caso das segunda e terceira batelada de H, esta paralela inicialmente recebe o mesmo start time de op, mas é verificada a factibilidade deste instante com respeito a processadores, recursos e indisponibilidade de tempo, pois, como as duas operações não possuem o mesmo tempo de processamento e compartilham os seus processadores com operações diferentes, não

é garantido que o EBT encontrado para uma seja também factível para a outra. O EBT da paralela de op é então calculado através da função start_time a partir do valor obtido para op. Se houver algum conflito e não for possível alocar PL[op] no mesmo instante encontrado para op, o índice das operações é decrementado de 1 e um novo instante de início para op é calculado por meio da função start_time, a partir do máximo entre os dois EBT's calculados para as operações paralelas. O cálculo é repetido até que seja obtido um EBT igual e factível para as duas operações.

Antes de se promover o recorte da janela da operação posterior à estas operações paralelas que possuem tempos de processamento diferentes, um cálculo à priori deve ser efetuado no cálculo do EBT desta operação posterior, no que se refere a determinação do seu instante de início em função da rota de produção. Neste caso, o start time da POS[op], onde op é a segunda operação paralela, não é obtido a partir da equação 4.11 como as demais mas, a esta equação, deve ainda ser adicionado o tempo de transferência da PL[op], e não só o tempo de transferência da anterior de op, para que o instante de início da operação em questão seja calculado corretamente. É este valor inicial que é levado à função start_time para, a partir dele, ser obtido o EBT da POS[op]. Assim, o BT[POS[op]] relativo à rota de produção para esta operação é dado por:

$$BT[POS[op]] = BT[op] + TT[ANT[op]] + TT[PL[op]] + TP\ [op] \tag{4.20}$$

onde:

BT[POS[op]] = EBT da operação posterior às operações paralelas com tempo de processamento diferentes.

BT[op] = EBT da segunda operação paralela.

TT[ANT[op]] = tempo de transferência da operação anterior às operações paralelas.

TT[PL[op]] = tempo de transferência da primeira operação paralela.

TP[op] = tempo de processamento da segunda operação paralela.

Por outro lado, se op possuir uma paralela, e esta paralela tiver o mesmo tempo de processamento de op, como acontece com as segunda e terceira operações de D, o algoritmo força que o start time da PL[op] seja igual ao calculado para op, e o índice das operações é incrementado em 1 sem que nenhum outro cálculo seja efetuado para esta paralela, passando-

se para o cálculo da janela da operação seguinte, pois este start time de op já está consistente no que se refere a processadores, recursos e indisponibilidades de tempo para as duas operações. O exemplo 3 representa o cálculo do EBT da paralela de op.

4.5-EXEMPLOS DE CONSTRUÇÃO DA SEQÜÊNCIA DE OPERAÇÕES

Neste item serão apresentadas algumas seqüências parciais para exemplificar os cálculos efetuados a fim de se determinar os instantes de início das operações não seqüenciadas bem como os diversos casos em que se verifica a consistência do instante de início escolhido para a operação candidata. Um exemplo do desenvolvimento de uma seqüência completa está demonstrado no Apêndice 4. Também é apresentada a carta de Gantt relativa a cada seqüência. As áreas delimitadas na carta representa as indisponibilidades de tempo. Cada página da carta contém 80 unidades de tempo, sendo que a última unidade de tempo é repetida na página seguinte. A notação escolhida para representar os produtos na carta de Gantt foi a mesma adotada pelo Rippin. Assim:

Produto D = A

Produto E1 = B

Produto F = C

Produto H = D

Produto E2 = E

EXEMPLO 1 - Janelas iniciais de cada operação.

Abaixo estão apresentadas as janelas iniciais das operações quando ainda não existe nenhuma operação alocada. As operações candidatas são as primeiras operações das primeiras bateladas de cada produto.

```
Candidatas: 1->0  21->0  36->0  40->0  43->402

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 66 210]  [ 70 225]  [ 70 225]  [ 85 235]  [ 95 248]
D3 [ 95 406]  [ 99 421]  [ 99 421]  [109 431]  [119 444]
D4 [124 478]  [128 493]  [128 493]  [143 503]  [153 516]
H1 [  0  62]  [  7  63]  [  7  72]  [ 15  80]  [ 66 114]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [ 84 468]
H3 [ 66 482]  [ 73 483]  [ 73 492]  [ 81 500]  [ 91 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [  0 181]  [  8 184]  [ 13 198]
E2 [402 429]  [410 432]  [414 444]
```

EXEMPLO 2 - Teste de factibilidade do instante de inicio proposto para operações posteriores à operações instáveis.

Quando uma operação op é instável, a operação que a sucede na rota de produção deve ser alocada no instante de tempo imediatamente após o término do processamento de op para que o material instável não tenha que aguardar no processador, podendo se degradar. Assim, quando a operação instável é alocada, a única operação candidata é a sua próxima na rota de produção e o único instante de início que é aceito para esta próxima é o início de sua janela. No exemplo abaixo, a operação 41 é instável e só é permitido que a operação 42 seja alocada no instante de início calculado para sua janela, que coincide com o término da operação 41.

```
Candidatas: 1->0  21->0  36->0  40->0  43->402
ITERACAO 1
OPERACAO:40   START TIME:0

D1 [  8  46]  [ 12  61]  [ 12  61]  [ 66  71]  [ 76  84]
D2 [ 84 210]  [ 88 225]  [ 88 225]  [113 235]  [123 248]
D3 [124 406]  [128 421]  [128 421]  [160 431]  [170 444]
D4 [164 478]  [168 493]  [168 493]  [282 503]  [292 516]
H1 [ 95  62]  [102  63]  [102  72]  [110  80]  [120 114]
H2 [113 416]  [123 417]  [123 426]  [131 434]  [282 468]
H3 [131 482]  [144 483]  [144 492]  [152 500]  [327 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [  0  11]  [  8 184]  [ 13 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->8  36->0  41->8  43->402
ITERACAO 2
OPERACAO:41   START TIME:8

D1 [  8  46]  [ 12  61]  [ 12  61]  [ 66  71]  [ 76  84]
D2 [ 84 210]  [ 88 225]  [ 88 225]  [113 235]  [123 248]
D3 [124 406]  [128 421]  [128 421]  [160 431]  [170 444]
D4 [164 478]  [168 493]  [168 493]  [282 503]  [292 516]
H1 [ 95  62]  [102  63]  [102  72]  [110  80]  [120 114]
H2 [113 416]  [123 417]  [123 426]  [131 434]  [282 468]
H3 [131 482]  [144 483]  [144 492]  [152 500]  [327 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [  0  11]  [  8  14]  [ 13 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 42->13
ITERACAO 3
OPERACAO:42   START TIME:15

A OPERACAO 42 TEM QUE COMECAR EM 13
```

```
Candidatas: 42->13
ITERACAO 3
OPERACAO:42    START TIME:10

                    A OPERACAO 42 TEM QUE COMECAR EM 13

Candidatas: 42->13
ITERACAO 3
OPERACAO:42    START TIME:13

D1 [  8  46]  [ 12  61]  [ 12  61]  [ 66  71]  [ 76  84]
D2 [ 84 210]  [ 88 225]  [ 88 225]  [113 235]  [123 248]
D3 [124 406]  [128 421]  [128 421]  [160 431]  [170 444]
D4 [164 478]  [168 493]  [168 493]  [282 503]  [292 516]
H1 [ 95  62]  [102  63]  [102  72]  [110  80]  [120 114]
H2 [113 416]  [123 417]  [123 426]  [131 434]  [282 468]
H3 [131 482]  [144 483]  [144 492]  [152 500]  [327 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 80 468]
E1 [  0  11]  [  8  14]  [ 13  28]
E2 [402 429]  [410 432]  [414 444]
```

CARTA DE GANTT

```
    |----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|
R1  ..............................|.....................................|.............
R2  ..............................|.....................................|.............
R3  BBBBBBBBBBB...................|.....................................|.............
R4  ..............................|.....................................|.............
R6  ..............................|.....................................|.............
R7  ..............................|.....................................|.............
F1  ..............................|.....................................|.............
FP1 ........BBBBBB................|.....................................|.............
FP2 ..............................|.....................................|.............
TRO ..............................|.....................................|.............
TRS .............BBBBBBBBBBBBBBB..|.....................................|.............
    |----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|
ELE 22222222333224444444444444440000000000000000000000000000000000000000000000000000
VAP 22222222000000000000000000000000000000000000000000000000000000000000000000000000
    |----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|
    0                           30                                  65            80
```

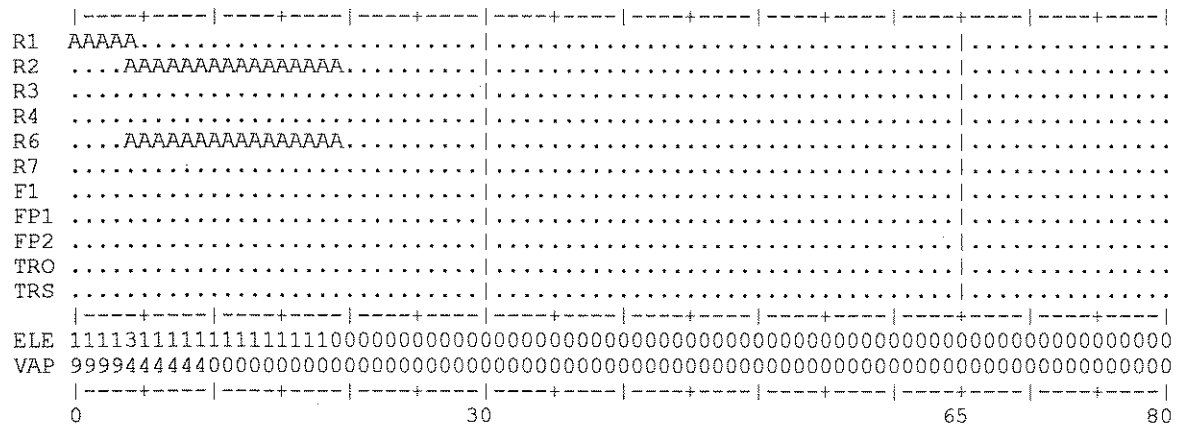
EXEMPLO 3 - Teste de factibilidade do instante de início proposto para operações em paralelo

As operações em paralelo têm obrigatoriamente que possuir o mesmo instante de início. Se for escolhido qualquer outro instante de início diferente para uma operação cuja paralela já tenha sido seqüenciada, o algoritmo rejeita este instante e não permite a alocação. Além disto, se existir uma operação paralela à operação que acabou de ser seqüenciada, apenas ela aparece como candidata, como pode ser visto no exemplo abaixo com as operações 2 e 3,

segunda e terceira operações de D, respectivamente. A operação 1 é instável e por isso, apenas a operação 2 é candidata a sucede-la na seqüência.

```
Candidatas: 1->0  21->0  36->0  40->0  43->402
ITERACAO 1
OPERACAO:1   START TIME:0

D1 [  0   5]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  4  62]  [ 11  63]  [ 11  72]  [ 19  80]  [ 66 114]
H2 [ 22 416]  [ 68 417]  [ 68 426]  [ 76 434]  [111 468]
H3 [ 76 482]  [ 89 483]  [ 89 492]  [ 97 500]  [156 534]
F  [  4 438]  [ 15 444]  [ 21 447]  [ 66 468]
E1 [  4 181]  [ 66 184]  [ 71 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 2->4
ITERACAO 2
OPERACAO:2   START TIME:4

D1 [  0   5]  [  4  20]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  4  62]  [ 11  63]  [ 11  72]  [ 19  80]  [ 66 114]
H2 [ 22 416]  [ 68 417]  [ 68 426]  [ 76 434]  [111 468]
H3 [ 76 482]  [ 89 483]  [ 89 492]  [ 97 500]  [156 534]
F  [  4 438]  [ 15 444]  [ 21 447]  [ 66 468]
E1 [  4 181]  [ 66 184]  [ 71 198]
E2 [402 429]  [410 432]  [414 444]

Candidatas: 3->4
ITERACAO 3
OPERACAO:3   START TIME:10

                    A OPERACAO 3 TEM QUE COMECAR EM 4

Candidatas: 3->4
ITERACAO 3
OPERACAO:3   START TIME:3

                    A OPERACAO 3 TEM QUE COMECAR EM 4

Candidatas: 3->4
ITERACAO 3
OPERACAO:3   START TIME:4

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  4  62]  [ 11  63]  [ 11  72]  [ 19  80]  [ 66 114]
H2 [ 22 416]  [ 68 417]  [ 68 426]  [ 76 434]  [111 468]
H3 [ 76 482]  [ 89 483]  [ 89 492]  [ 97 500]  [156 534]
F  [  4 438]  [ 15 444]  [ 21 447]  [ 66 468]
E1 [  4 181]  [ 66 184]  [ 71 198]
E2 [402 429]  [410 432]  [414 444]
```

CARTA DE GANTT



EXEMPLO 4 - Teste de factibilidade do instante de início proposto com respeito a ocupação do processador, onde o start time escolhido é menor que o calculado para op.

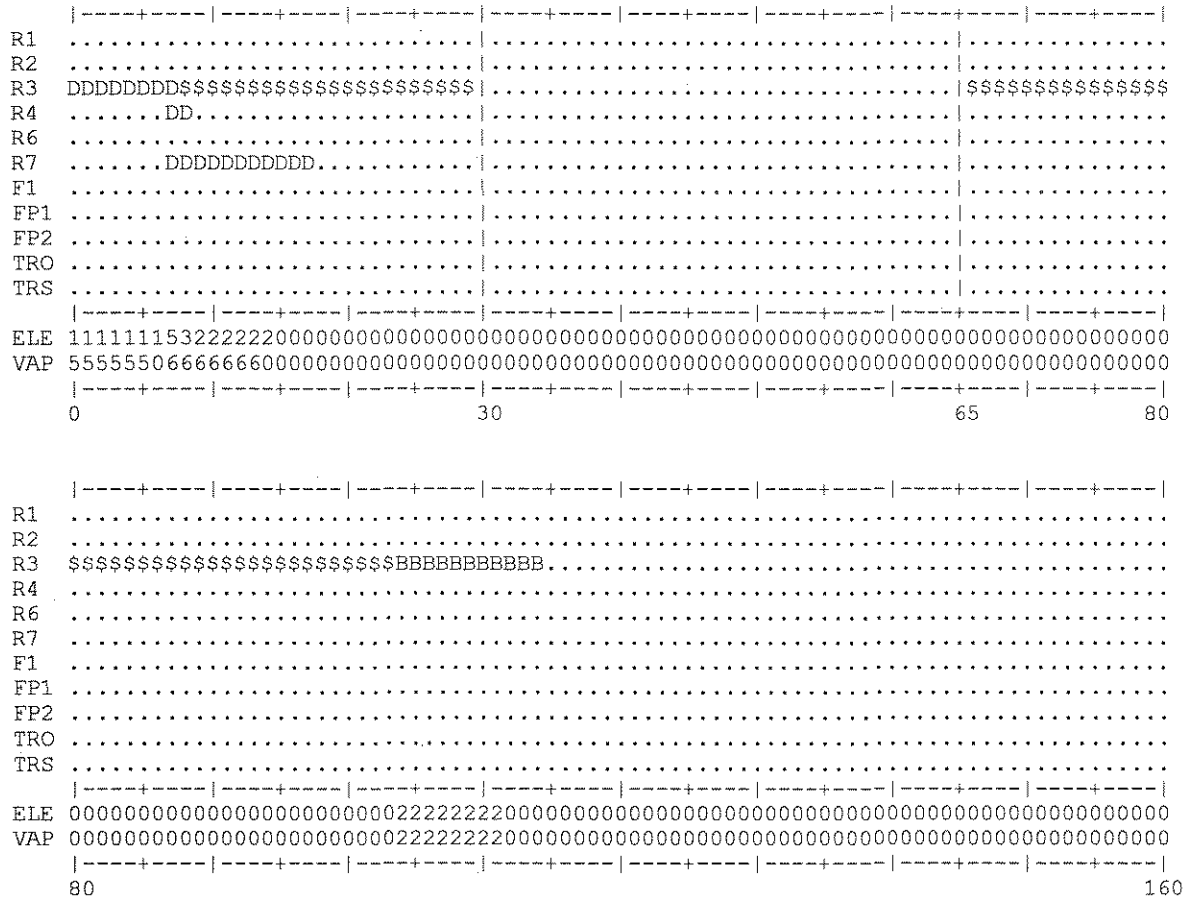
Para este exemplo a operação op é a operação 40. A primeira operação alocada na planta é a operação 21, primeira operação de H1, que utiliza o mesmo processador de op. O início de sua janela é calculado pelo instante em que a operação 21 libera o processador, em t = 104. Em seguida, a operação 40 é alocada e é escolhido um start time menor do que a sua janela (t = 100), provocando um conflito de ocupação do processador, sendo este instante de início rejeitado.

```
Candidatas: 1->0  21->0  36->0  40->0  43->402
ITERACAO 1
OPERACAO:21   START TIME:0
```

D1	[6 46]	[10 61]	[10 61]	[66 71]	[76 84]
D2	[82 210]	[86 225]	[86 225]	[113 235]	[123 248]
D3	[122 406]	[126 421]	[126 421]	[160 431]	[170 444]
D4	[162 478]	[166 493]	[166 493]	[282 503]	[292 516]
H1	[0 8]	[7 63]	[7 72]	[15 80]	[66 114]
H2	[18 416]	[66 417]	[66 426]	[74 434]	[111 468]
H3	[72 482]	[87 483]	[87 492]	[95 500]	[156 534]
F	[0 438]	[11 444]	[17 447]	[66 468]	
E1	[104 181]	[112 184]	[117 198]		
E2	[402 429]	[410 432]	[414 444]		

```
Candidatas: 1->6  22->7  36->0  43->402
ITERACAO 2
OPERACAO:22   START TIME:7

D1 [  8  46]  [ 12  61]  [ 12  61]  [ 66  71]  [ 76  84]
D2 [ 84 210]  [ 88 225]  [ 88 225]  [113 235]  [123 248]
D3 [124 406]  [128 421]  [128 421]  [160 431]  [170 444]
D4 [164 478]  [168 493]  [168 493]  [282 503]  [292 516]
H1 [  0   8]  [  7   9]  [  7  72]  [ 15  80]  [ 66 114]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [ 69 438]  [ 80 444]  [ 86 447]  [ 89 468]
E1 [104 181]  [112 184]  [117 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 23->7
ITERACAO 3
OPERACAO:23   START TIME:7

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [ 95  84]
D2 [106 210]  [110 225]  [110 225]  [132 235]  [142 248]
D3 [146 406]  [150 421]  [150 421]  [179 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
H1 [  0   8]  [  7   9]  [  7  18]  [ 15  80]  [ 66 114]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [ 69 438]  [ 80 444]  [ 86 447]  [ 89 468]
E1 [104 181]  [112 184]  [117 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->66  24->15  26->18  36->69  40->104  43->402
ITERACAO 4
OPERACAO:40   START TIME:100


INSTANTE INFACTIVEL POR ROTA, PROCESSADOR OU INDISPONIBILIDADE DE TEMPO


Candidatas: 1->66  24->15  26->18  36->69  40->104  43->402
ITERACAO 4
OPERACAO:40   START TIME:104

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [ 95  84]
D2 [112 210]  [116 225]  [116 225]  [132 235]  [142 248]
D3 [152 406]  [156 421]  [156 421]  [179 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
F  [  0   8]  [  7   9]  [  7  18]  [ 15  80]  [ 66 114]
H1 [163 416]  [170 417]  [170 426]  [178 434]  [282 468]
H2 [181 482]  [234 483]  [234 492]  [282 500]  [327 534]
H3 [ 69 438]  [ 80 444]  [ 86 447]  [112 468]
E1 [104 115]  [112 184]  [117 198]
E2 [402 429]  [410 432]  [414 444]
```

CARTA DE GANTT

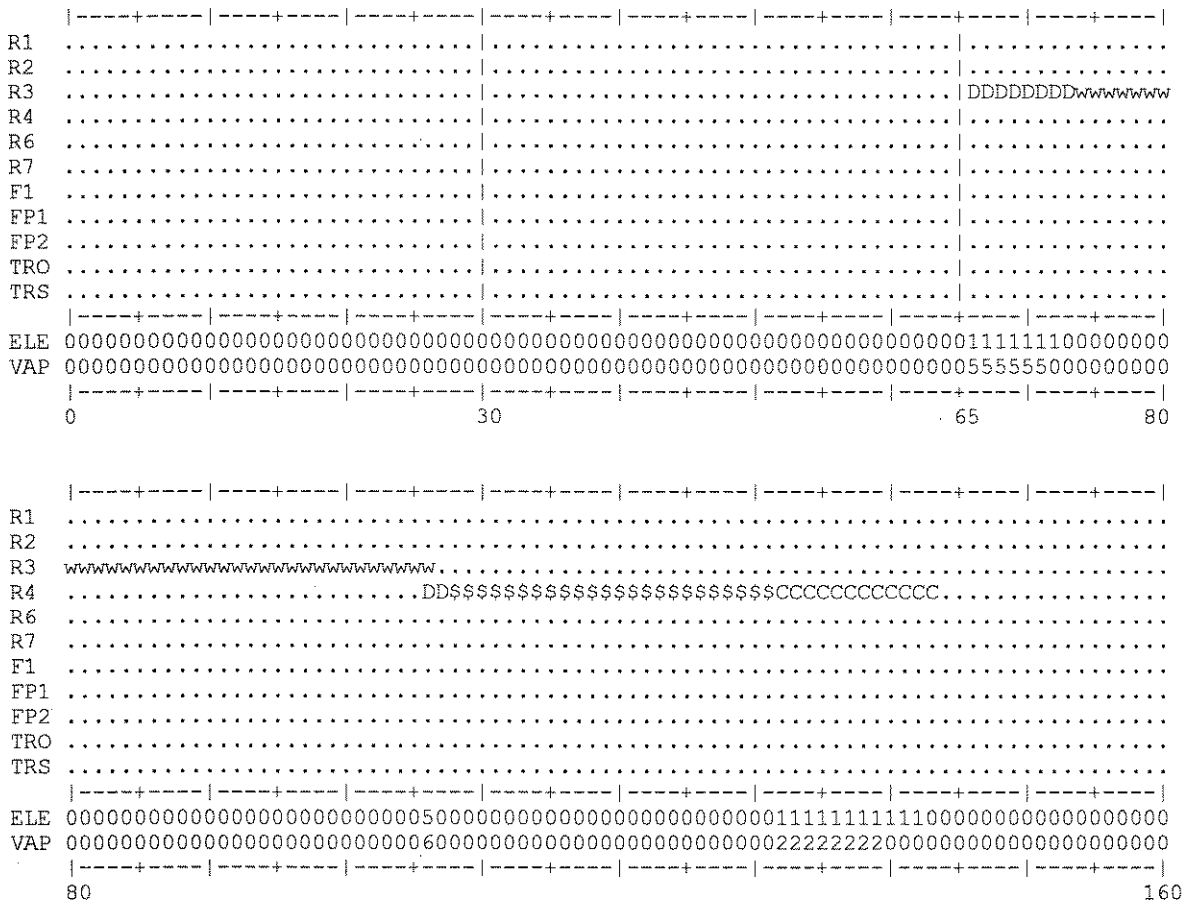


EXEMPLO 5 - Teste de factibilidade do instante de início proposto com respeito a ocupação do processador, onde o start time escolhido é maior que a janela de op.

A primeira operação alocada na planta é a operação 36 em t = 132, de forma que a operação 22, que compartilha o mesmo equipamento, possa ser alocada em um instante anterior a este. Porém, o primeiro instante escolhido para a operação 22 (t = 110) não foi aceito uma vez que o intervalo de tempo disponível entre ele e o início da operação 36 não é suficiente para o processamento da operação 22 mais o set up entre elas. O algoritmo emite uma mensagem informando que a operação em questão não pode ser iniciada no instante escolhido.

```
Candidatas: 1->0  21->0  36->0  40->0  43->402
ITERACAO 1
OPERACAO:36   START TIME:132

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  0  62]  [  7  63]  [  7  72]  [ 15  80]  [ 66 114]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [132 144]  [143 444]  [149 447]  [152 468]
E1 [  0 181]  [  8 184]  [ 13 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->0  21->0  37->143  40->0
ITERACAO 2
OPERACAO:21   START TIME:66

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 66  74]  [ 73  63]  [ 73  72]  [ 81  80]  [ 91 114]
H2 [ 84 416]  [ 94 417]  [ 94 426]  [102 434]  [136 468]
H3 [102 482]  [164 483]  [164 492]  [172 500]  [282 534]
F  [132 144]  [143 444]  [149 447]  [152 468]
E1 [134 181]  [142 184]  [147 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->0  22->73  37->143
ITERACAO 3
OPERACAO:22   START TIME:110


          A OPERACAO 22 NAO PODE COMECAR NESTE INSTANTE


Candidatas: 1->0  22->73  37->143
ITERACAO 3
OPERACAO:22   START TIME:106

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 66  74]  [106 108]  [106  72]  [114  80]  [124 114]
H2 [117 416]  [164 417]  [164 426]  [172 434]  [282 468]
H3 [135 482]  [185 483]  [185 492]  [234 500]  [327 534]
F  [132 144]  [143 444]  [149 447]  [152 468]
E1 [167 181]  [175 184]  [180 198]
E2 [402 429]  [410 432]  [414 444]
```

CARTA DE GANTT



EXEMPLO 6 - Recálculo de janelas em função dos start times calculados para as operações posteriores na rota de produção

Neste exemplo, pretende-se demonstrar como a janela calculada para uma dada operação pode provocar o recálculo das janelas das operações anteriores a ela na sua rota de produção. Para isto, deve-se observar o cálculo da janela da operação 30, última operação da batelada de H2, quando é alocada a operação 42 da batelada de E1.

O seqüenciamento inicia-se com a alocação das operações de H1 seguidas pelas primeiras operações de F e das operações 40 e 41 da batelada de E1, alocadas em instantes de início posteriores. Estas operações foram alocadas de forma a permitir que as operações da segunda batelada de H que utilizam os mesmos equipamentos que estas operações, possam ser alocadas entre elas e as operações de H1.

```
Candidatas: 1->0  21->0  36->0  40->0  43->402
ITERACAO 1
OPERACAO:21   START TIME:0

D1 [  6  46]  [ 10  61]  [ 10  61]  [ 66  71]  [ 76  84]
D2 [ 82 210]  [ 86 225]  [ 86 225]  [113 235]  [123 248]
D3 [122 406]  [126 421]  [126 421]  [160 431]  [170 444]
D4 [162 478]  [166 493]  [166 493]  [282 503]  [292 516]
H1 [  0   8]  [  7  63]  [  7  72]  [ 15  80]  [ 66 114]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [  0 438]  [ 11 444]  [ 18 447]  [ 66 468]
E1 [104 181]  [112 184]  [117 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->6  22->7  36->0  43->402
ITERACAO 2
OPERACAO:22   START TIME:7

D1 [  8  46]  [ 12  61]  [ 12  61]  [ 66  71]  [ 76  84]
D2 [ 84 210]  [ 88 225]  [ 88 225]  [113 235]  [123 248]
D3 [124 406]  [128 421]  [128 421]  [160 431]  [170 444]
D4 [164 478]  [168 493]  [168 493]  [282 503]  [292 516]
H1 [  0   8]  [  7   9]  [  7  72]  [ 15  80]  [ 66 114]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [ 69 438]  [ 80 444]  [ 86 447]  [ 89 468]
E1 [104 181]  [112 184]  [117 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 23->7
ITERACAO 3
OPERACAO:23   START TIME:7

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [ 95  84]
D2 [106 210]  [110 225]  [110 225]  [132 235]  [142 248]
D3 [146 406]  [150 421]  [150 421]  [179 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
H1 [  0   8]  [  7   9]  [  7  18]  [ 15  80]  [ 66 114]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [ 69 438]  [ 80 444]  [ 86 447]  [ 89 468]
E1 [104 181]  [112 184]  [117 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 1->66  24->15  26->18  36->69  40->104  43->402
ITERACAO 4
OPERACAO:24   START TIME:15

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [ 95  84]
D2 [106 210]  [110 225]  [110 225]  [132 235]  [142 248]
D3 [146 406]  [150 421]  [150 421]  [179 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
H1 [  0   8]  [  7   9]  [  7  18]  [ 15  26]  [ 66 114]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [ 69 438]  [ 80 444]  [ 86 447]  [ 89 468]
E1 [104 181]  [112 184]  [117 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->66  25->66  26->18  36->69  40->104  43->402
ITERACAO 5
OPERACAO:25   START TIME:66

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [101  84]
D2 [106 210]  [110 225]  [110 225]  [132 235]  [142 248]
D3 [146 406]  [150 421]  [150 421]  [179 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
H1 [  0   8]  [  7   9]  [  7  18]  [ 15  26]  [ 66 101]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [ 69 438]  [ 80 444]  [ 86 447]  [125 468]
E1 [104 181]  [120 184]  [125 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->66  26->18  36->69  40->104  43->402
ITERACAO 6
OPERACAO:36   START TIME:151

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [101  84]
D2 [106 210]  [110 225]  [110 225]  [132 235]  [142 248]
D3 [146 406]  [150 421]  [150 421]  [179 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
H1 [  0   8]  [  7   9]  [  7  18]  [ 15  26]  [ 66 101]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [151 163]  [162 444]  [168 447]  [171 468]
E1 [104 181]  [120 184]  [125 198]
E2 [402 429]  [410 432]  [414 444]
```

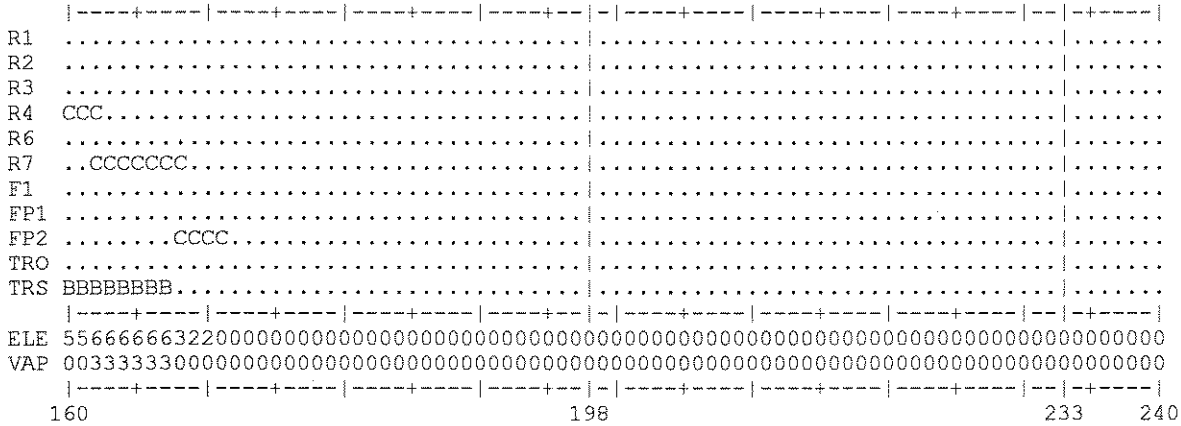
```
Candidatas: 1->66  26->18  37->162  40->104
ITERACAO 7
OPERACAO:37   START TIME:162

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [101  84]
D2 [106 210]  [110 225]  [110 225]  [132 235]  [142 248]
D3 [146 406]  [150 421]  [150 421]  [179 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
H1 [  0   8]  [  7   9]  [  7  18]  [ 15  26]  [ 66 101]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [151 163]  [162 169]  [168 447]  [171 468]
E1 [104 181]  [120 184]  [125 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 38->168
ITERACAO 8
OPERACAO:38   START TIME:168

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [101  84]
D2 [106 210]  [110 225]  [110 225]  [132 235]  [142 248]
D3 [146 406]  [150 421]  [150 421]  [179 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
H1 [  0   8]  [  7   9]  [  7  18]  [ 15  26]  [ 66 101]
H2 [ 18 416]  [ 66 417]  [ 66 426]  [ 74 434]  [111 468]
H3 [ 72 482]  [ 87 483]  [ 87 492]  [ 95 500]  [156 534]
F  [151 163]  [162 169]  [168 172]  [171 468]
E1 [104 181]  [120 184]  [125 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 1->66  26->18  39->171  40->104  43->402
ITERACAO 9
OPERACAO:40   START TIME:140

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [101  84]
D2 [106 210]  [110 225]  [110 225]  [132 235]  [148 248]
D3 [168 406]  [172 421]  [172 421]  [187 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [304 503]  [314 516]
H1 [  0   8]  [  7   9]  [  7  18]  [ 15  26]  [ 66 101]
H2 [235 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [287 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [151 163]  [162 169]  [168 172]  [171 468]
E1 [140 151]  [148 184]  [153 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 1->66  39->171  41->148  43->402
ITERACAO 10
OPERACAO:41   START TIME:148

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [101  84]
D2 [106 210]  [110 225]  [110 225]  [132 235]  [153 248]
D3 [168 406]  [172 421]  [172 421]  [187 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [304 503]  [314 516]
H1 [  0   8]  [  7   9]  [  7  18]  [ 15  26]  [ 66 101]
H2 [235 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [287 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [151 163]  [162 169]  [168 172]  [171 468]
E1 [140 151]  [148 154]  [153 198]
E2 [402 429]  [410 432]  [414 444]
```

Quando a operação 42 é alocada em t = 153, não é possível o posicionamento da operação 30 entre ela e a operação 25, sendo o início de seu processamento permitido, no mínimo, em t = 282, após o término da operação 40. Em vista disto, as operações anteriores a ela na rota de produção também não poderão ter o início de suas janelas posicionados antes das operações de E1 e F, com quem compartilham os equipamentos, apesar do intervalo de tempo disponível entre as operações de H1 e desta bateladas ser suficientes para o processamento das operações de H2 mais o set up. Porém, o deslocamento da janela da operação 30 para um instante de início muito posterior, impossibilita a alocação destas operações no instante anteriormente calculado para elas, uma vez que o material processado por estas operações teria que aguardar no processador até que seja efetuada a transferência para a próxima operação da rota de produção, o que provocaria um conflito de ocupação de processadores com as operações já alocadas em seus processadores. Assim, o algoritmo recalcula as janelas das operações de H2, e todas elas passam a ter seus instantes de início posicionados depois do processamento das operações de E1 e F.

```
Candidatas: 42->153
ITERACAO 11
OPERACAO:42   START TIME:153

D1 [ 66  46]  [ 70  61]  [ 70  61]  [ 85  71]  [101  84]
D2 [106 210]  [110 225]  [110 225]  [132 235]  [171 248]
D3 [168 406]  [172 421]  [172 421]  [187 431]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [304 503]  [314 516]
H1 [  0   8]  [  7   9]  [  7  18]  [ 15  26]  [ 66 101]
H2 [235 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [287 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [151 163]  [162 169]  [168 172]  [282 468]
E1 [140 151]  [148 154]  [153 168]
E2 [402 429]  [410 432]  [414 444]
```

CARTA DE GANTT



EXEMPLO 7- Janela calculada para as operações posteriores infactibiliza o instante de início proposto para op por conflito de ocupação de processador.

Neste exemplo, a janela calculada para uma ou mais das operações posteriores a op na rota, infactibiliza a alocação de op no instante dado. Neste caso, a operação op é a operação 21, primeira operação de H2. O seqüenciamento inicia-se alocando-se a batelada de E1 (operações de 40 a 42) e as primeiras operações da batelada de F (operações de 36 a 38), de forma tal que ainda é possível alocar as operações de H1 em instantes de início anteriores aos destas operações.

```
Candidatas: 1->0  21->0  36->0  40->0  43->402
ITERACAO 1
OPERACAO:40   START TIME:137

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  0  62]  [  7  63]  [  7  72]  [ 15  80]  [ 66 114]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [137 148]  [145 184]  [150 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->0  21->0  36->0  41->145  43->402
ITERACAO 2
OPERACAO:41   START TIME:145

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  0  62]  [  7  63]  [  7  72]  [ 15  80]  [ 66 114]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [137 148]  [145 151]  [150 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 42->150
ITERACAO 3
OPERACAO:42   START TIME:150

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [165 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  0  62]  [  7  63]  [  7  72]  [ 15  80]  [ 66 114]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [137 148]  [145 151]  [150 165]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->0  21->0  36->0  43->402
ITERACAO 4
OPERACAO:36   START TIME:140

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [165 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  0  62]  [  7  63]  [  7  72]  [ 15  80]  [ 66 114]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [140 152]  [151 444]  [157 447]  [282 468]
E1 [137 148]  [145 151]  [150 165]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->0  21->0  37->151
ITERACAO 5
OPERACAO:37   START TIME:151

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [165 444]
D4 [157 478]  [161 493]  [161 493]  [234 503]  [282 516]
H1 [  0  62]  [  7  63]  [  7  72]  [ 15  80]  [ 66 114]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [140 152]  [151 158]  [157 447]  [282 468]
E1 [137 148]  [145 151]  [150 165]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 38->157
ITERACAO 6
OPERACAO:38   START TIME:157

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [160 431]  [170 444]
D4 [157 478]  [161 493]  [161 493]  [282 503]  [292 516]
H1 [  0  62]  [  7  63]  [  7  72]  [ 15  80]  [ 66 114]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [140 152]  [151 158]  [157 161]  [282 468]
E1 [137 148]  [145 151]  [150 165]
E2 [402 429]  [410 432]  [414 444]
```

A operação 21 é então alocada em t = 67. Quando é efetuado o cálculo da janela da operação 25 a partir deste instante, observa-se que o intervalo de tempo disponível entre o início de sua janela, levando-se em conta apenas a rota de produção, e o início da operação 42, que utiliza o mesmo processador, não é suficiente para o seu processamento mais o set up necessário entre elas. Desta forma, só é possível aloca-la em t = 282, depois do processamento desta operação, o que obrigaria o material processado na operação 24 a aguardar no processador até que fosse transferido para a operação 25. Obviamente isto seria impossível visto que a operação 41 já está alocada em t = 145. Para eliminar este problema, o início da operação 24 é então recalculado para após a operação 41, sendo o mesmo procedimento efetuado para as operações 23, com relação à operação 37, e com a operação 22, com relação a operação 36. Porém, com a janela da operação 22 posicionada após a operação 36 e a operação 21 alocada em t = 67, o material processado em 21 teria que aguardar no equipamento até ser transferido para a operação 22 em t = 172. Estando a operação 40, que utiliza o mesmo processador que a operação 21, já alocada em t = 137, este posicionamento da operação 21 é impossível, e o instante de início proposto é rejeitado e uma mensagem avisa qual a primeira que operação que tornou o instante infactível.

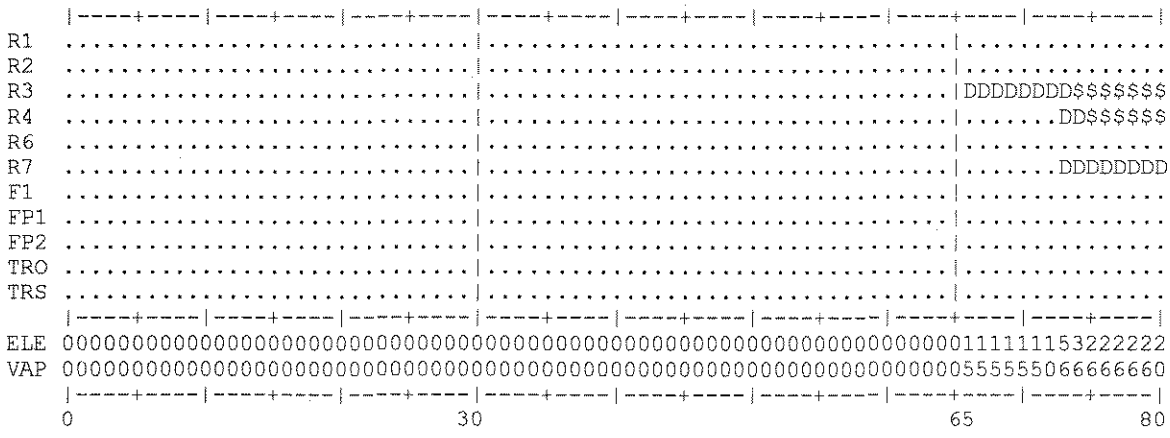
```
Candidatas: 1->0  21->0  39->282  43->402
ITERACAO 7
OPERACAO:21   START TIME:67

          INFACTIBILIDADE CAUSADA PELA OPERACAO 24
```

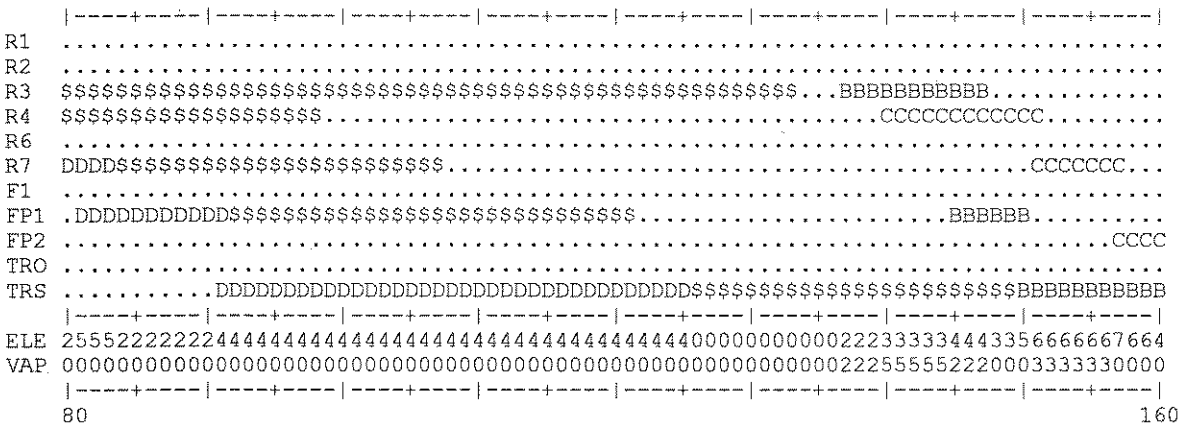
Porém, se a operação 21 for alocada em t = 66, o instante de início proposto é aceito pois é possível alocar todas as operações posteriores na rota de produção a partir deste instante, sem que ocorra conflito de ocupação de processador. Observa-se pela carta de Gantt que, com a operação 21 alocada em t=66, o intervalo de tempo disponível entre o início da janela da operação 25 e o início da operação 42 é apenas o suficiente para o seu processamento mais o set up entre elas.

```
Candidatas: 1->0  21->0  39->282  43->402
ITERACAO 7
OPERACAO:21   START TIME:66

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [160 431]  [170 444]
D4 [157 478]  [161 493]  [161 493]  [282 503]  [292 516]
H1 [ 66  74]  [ 73  63]  [ 73  72]  [ 81  80]  [ 91 114]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [140 152]  [151 158]  [157 161]  [282 468]
E1 [137 148]  [145 151]  [150 165]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->0  22->73  39->282  43->402
ITERACAO 8
OPERACAO:22   START TIME:73

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 74  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [160 431]  [170 444]
D4 [157 478]  [161 493]  [161 493]  [282 503]  [292 516]
H1 [ 66  74]  [ 73  75]  [ 73  72]  [ 81  80]  [ 91 114]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [140 152]  [151 158]  [157 161]  [282 468]
E1 [137 148]  [145 151]  [150 165]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 23->73
ITERACAO 9
OPERACAO:23   START TIME:73

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 81  84]
D2 [ 80 210]  [ 84 225]  [ 84 225]  [102 235]  [112 248]
D3 [120 406]  [124 421]  [124 421]  [160 431]  [170 444]
D4 [160 478]  [164 493]  [164 493]  [282 503]  [292 516]
H1 [ 66  74]  [ 73  75]  [ 73  84]  [ 81  80]  [ 91 114]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [140 152]  [151 158]  [157 161]  [282 468]
E1 [137 148]  [145 151]  [150 165]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 1->0  24->81  26->234  39->282  43->402
ITERACAO 10
OPERACAO:24   START TIME:81

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 91  84]
D2 [ 80 210]  [ 84 225]  [ 84 225]  [102 235]  [112 248]
D3 [120 406]  [124 421]  [124 421]  [160 431]  [170 444]
D4 [160 478]  [164 493]  [164 493]  [282 503]  [292 516]
H1 [ 66  74]  [ 73  75]  [ 73  84]  [ 81  92]  [ 91 114]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [140 152]  [151 158]  [157 161]  [282 468]
E1 [137 148]  [145 151]  [150 165]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->0  25->91  26->234  39->282  43->402
ITERACAO 11
OPERACAO:25   START TIME:91


D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [126  84]
D2 [ 80 210]  [ 84 225]  [ 84 225]  [102 235]  [165 248]
D3 [120 406]  [124 421]  [124 421]  [160 431]  [179 444]
D4 [160 478]  [164 493]  [164 493]  [282 503]  [292 516]
H1 [ 66  74]  [ 73  75]  [ 73  84]  [ 81  92]  [ 91 126]
H2 [234 416]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [286 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [140 152]  [151 158]  [157 161]  [282 468]
E1 [137 148]  [145 151]  [150 165]
E2 [402 429]  [410 432]  [414 444]
```

CARTA DE GANTT





EXEMPLO 8 - Start time escolhido para op infactibiliza o valor do start time da operação anterior na rota de produção.

Para este exemplo, op é a operação 27, segunda operação de H2, e a anterior de op é a operação 26, já alocada.

O seqüenciamento é iniciado com a primeira batelada de H (operações de 21 a 25) e a primeira operação de E1 (operação 40) sendo posicionadas nos instantes dados. A operação 26 é alocada no processador que a faz, entre as operações 21 e 40, em t = 170.

```
Candidatas: 1->0  21->0  36->0  40->0  43->402
ITERACAO 1
OPERACAO:21   START TIME:22

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 22  30]  [ 66  63]  [ 66  72]  [ 74  80]  [ 84 114]
H2 [ 76 416]  [ 87 417]  [ 87 426]  [ 95 434]  [129 468]
H3 [ 94 482]  [108 483]  [108 492]  [116 500]  [282 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [126 181]  [134 184]  [139 198]
E2 [402 429]  [410 432]  [414 444]
```

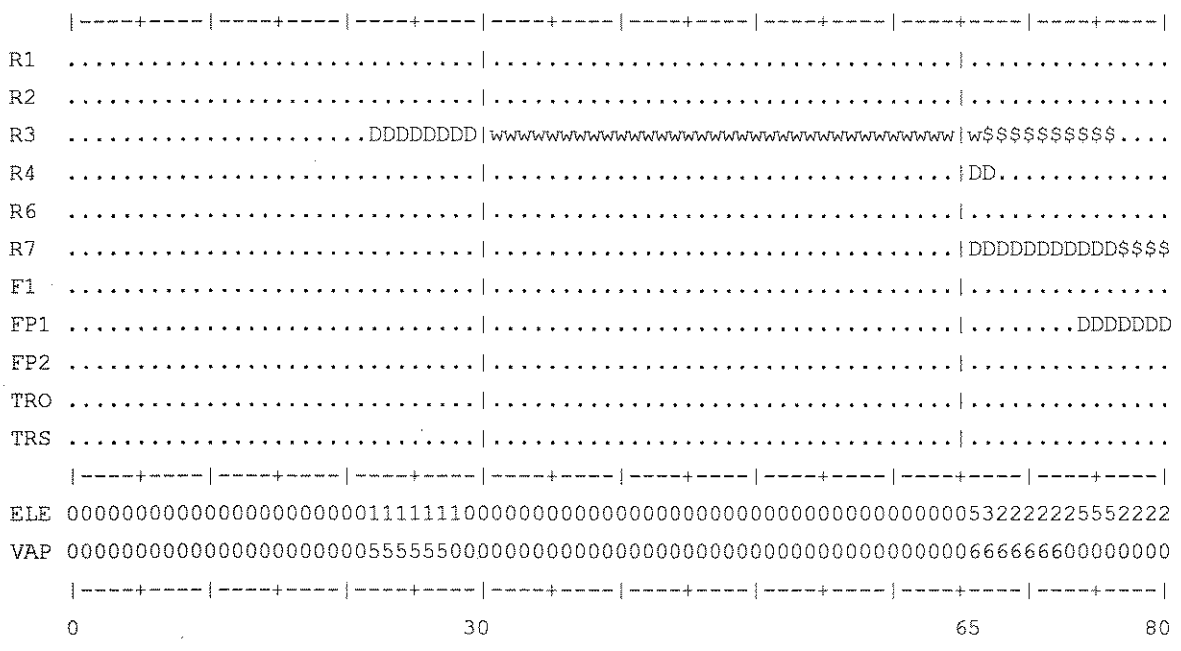
```
Candidatas: 1->0  22->66  36->0  43->402
ITERACAO 2
OPERACAO:22   START TIME:66

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 67  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 22  30]  [ 66  68]  [ 66  72]  [ 74  80]  [ 84 114]
H2 [ 77 416]  [ 87 417]  [ 87 426]  [ 95 434]  [129 468]
H3 [ 95 482]  [108 483]  [108 492]  [116 500]  [282 534]
F  [ 92 438]  [103 444]  [109 447]  [112 468]
E1 [127 181]  [135 184]  [140 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 23->66
ITERACAO 3
OPERACAO:23   START TIME:66

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 74  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 22  30]  [ 66  68]  [ 66  77]  [ 74  80]  [ 84 114]
H2 [ 77 416]  [ 87 417]  [ 87 426]  [ 95 434]  [129 468]
H3 [ 95 482]  [108 483]  [108 492]  [116 500]  [282 534]
F  [ 92 438]  [103 444]  [109 447]  [112 468]
E1 [127 181]  [135 184]  [140 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->0  24->74  26->77  36->92  40->127  43->402
ITERACAO 4
OPERACAO:24   START TIME:74

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 84  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 22  30]  [ 66  68]  [ 66  77]  [ 74  85]  [ 84 114]
H2 [ 77 416]  [ 87 417]  [ 87 426]  [ 95 434]  [129 468]
H3 [ 95 482]  [108 483]  [108 492]  [116 500]  [282 534]
F  [ 92 438]  [103 444]  [109 447]  [112 468]
E1 [127 181]  [135 184]  [140 198]
E2 [402 429]  [410 432]  [414 444]
```

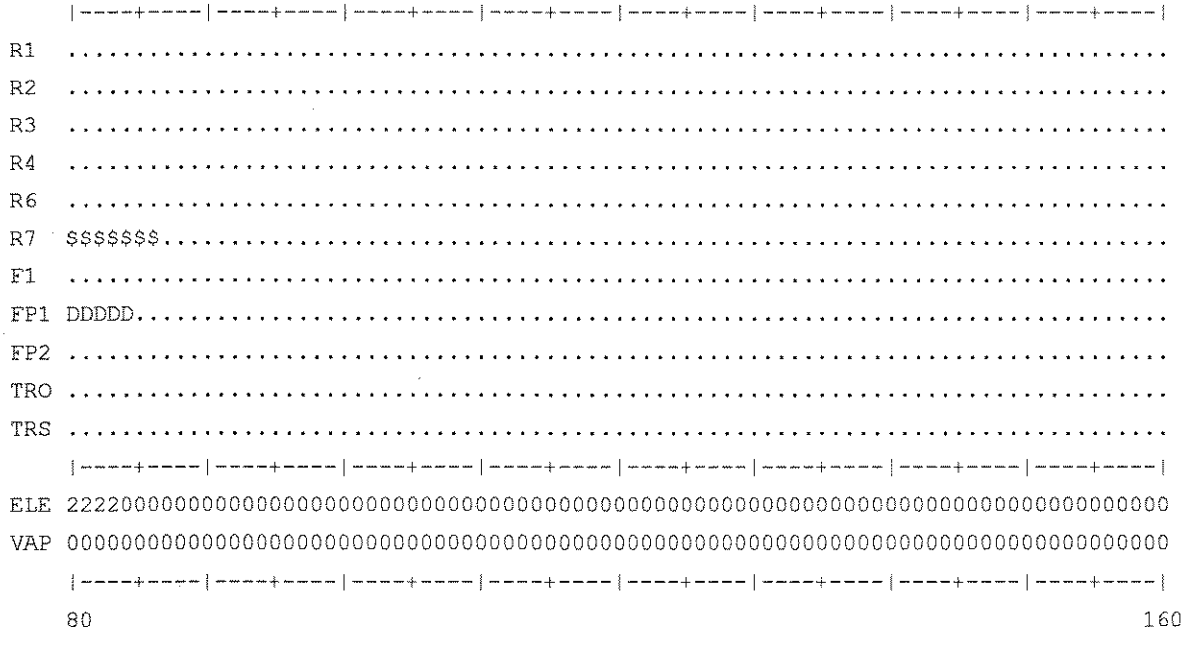
```
Candidatas: 1->0  25->84  26->77  36->92  40->127  43->402
ITERACAO 5
OPERACAO:40   START TIME:320

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 84  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 22  30]  [ 66  68]  [ 66  77]  [ 74  85]  [ 84 114]
H2 [ 77 416]  [ 87 417]  [ 87 426]  [ 95 434]  [129 468]
H3 [ 95 482]  [108 483]  [108 492]  [116 500]  [282 534]
F  [ 92 438]  [103 444]  [109 447]  [112 468]
E1 [320 331]  [328 184]  [333 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->0  25->84  26->77  36->92  43->402
ITERACAO 6
OPERACAO:26   START TIME:170

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 84  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 22  30]  [ 66  68]  [ 66  77]  [ 74  85]  [ 84 114]
H2 [170 178]  [177 417]  [177 426]  [185 434]  [282 468]
H3 [415 482]  [422 483]  [422 492]  [430 500]  [440 534]
F  [ 92 438]  [103 444]  [109 447]  [112 468]
E1 [320 331]  [328 184]  [333 198]
E2 [402 429]  [410 432]  [414 444]
```

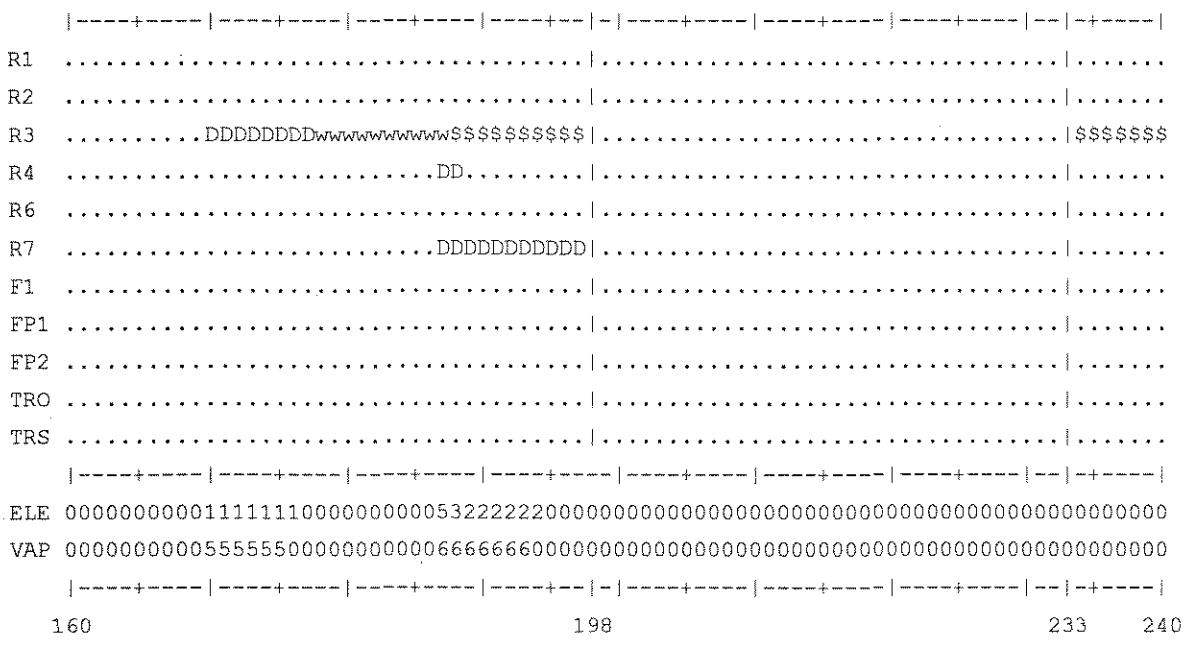
É escolhido para a operação 27 um start time igual a 234. Neste instante, porém, a alocação não é permitida visto que isto infactibilizaria o posicionamento da operação 26 no instante em que ela foi alocada, já que o material processado nesta operação teria que aguardar no equipamento até ser transferido, o que provocaria conflito de ocupação no processador entre as operações 26 e 40. Assim, só é possível alocar a operação 27, no máximo, em t = 187 para que este conflito não ocorra. Como existe duas horas ainda disponíveis depois do fim do set up entres as operações 26 e 40, como pode ser observado pela carta de Gantt, seria razoável supor que o posicionamento de op em 189 poderia ser permitido, porém isto não ocorre, pois em t = 198 inicia-se uma indisponibilidade de tempo e, como a operação 28 é paralela com a operação 27 mas possui tempo de processamento maior, não há tempo suficiente para terminar seu processamento, iniciando em de t = 189, antes do início desta indisponibilidade. Por isso, este instante de início para a operação 27 não é possível.

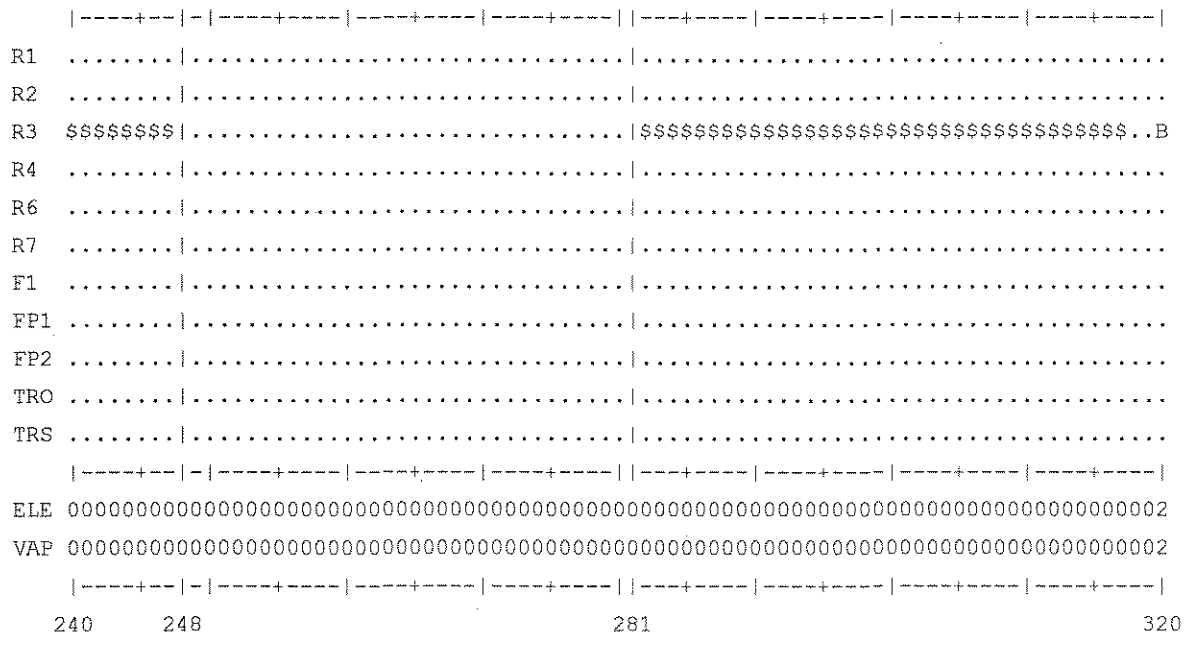
```
Candidatas: 1->0  25->84  27->177  36->92  43->402
ITERACAO 7
OPERACAO:27   START TIME:234

                INFACTIBILIDADE CAUSADA PELA OPERACAO 27

Candidatas: 1->0  25->84  27->177  36->92  43->402
ITERACAO 7
OPERACAO:27   START TIME:189

                INFACTIBILIDADE CAUSADA PELA OPERACAO 27

Candidatas: 1->0  25->84  27->177  36->92  43->402
ITERACAO 7
OPERACAO:27   START TIME:187

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 84  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 22  30]  [ 66  68]  [ 66  77]  [ 74  85]  [ 84 114]
H2 [170 178]  [187 189]  [187 426]  [234 434]  [282 468]
H3 [415 482]  [422 483]  [422 492]  [430 500]  [440 534]
F  [ 92 438]  [103 444]  [109 447]  [112 468]
E1 [320 331]  [328 184]  [333 198]
E2 [402 429]  [410 432]  [414 444]

Candidatas: 28->187
ITERACAO 8
OPERACAO:28   START TIME:187

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 84  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 22  30]  [ 66  68]  [ 66  77]  [ 74  85]  [ 84 114]
H2 [170 178]  [187 189]  [187 198]  [234 434]  [282 468]
H3 [415 482]  [422 483]  [422 492]  [430 500]  [440 534]
F  [ 92 438]  [103 444]  [109 447]  [112 468]
E1 [320 331]  [328 184]  [333 198]
E2 [402 429]  [410 432]  [414 444]
```

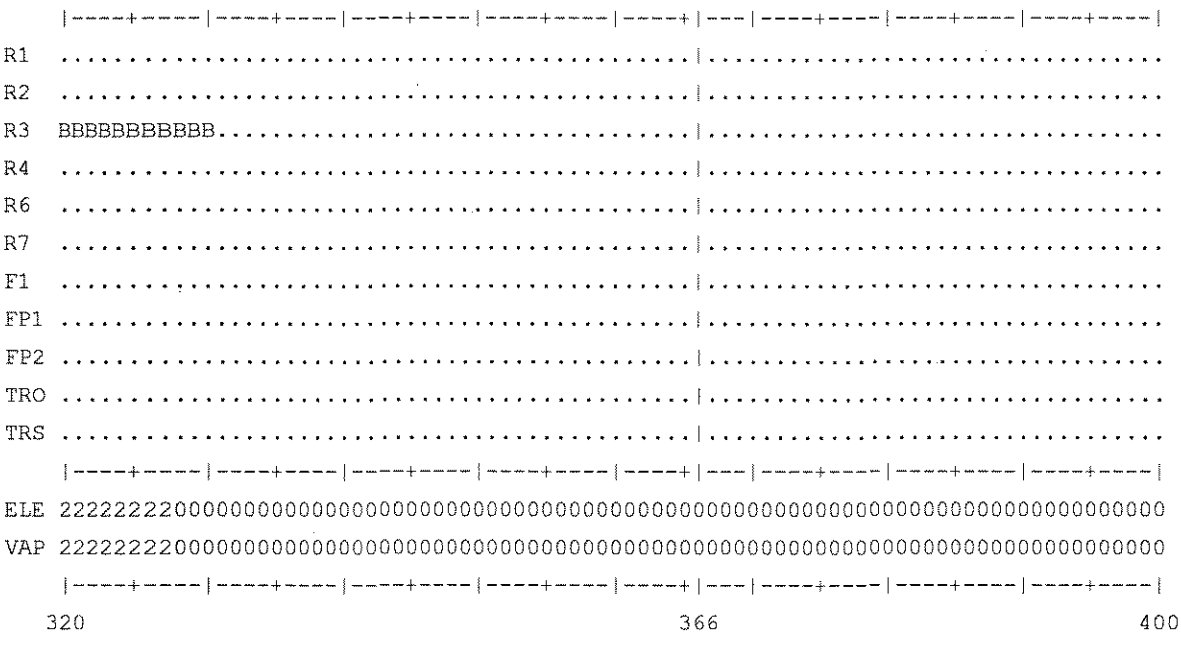
CARTA DE GANTT






R1
R2
R3 DDDDDDDDwwwwwwwwww$$$$$$$$$$
R4 DD
R6
R7 DDDDDDDDDDD
F1
FP1
FP2
TRO
TRS
ELE 0000000000111111100000000005322222200000000000000000000000000000000000000000000000
VAP 0000000000555555000000000066666660000000000000000000000000000000000000000000000000
160
198
233
240


R1
R2
R3 $$$$$$$$ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$..B
R4
R6
R7
F1
FP1
FP2
TRO
TRS
ELE 0000000000000000000000000000000000000000000000000000000000000000000000000000000002
VAP 0000000000000000000000000000000000000000000000000000000000000000000000000000000002
240
248
281
320



EXEMPLO 9 - Seqüência Infactível

Este exemplo demonstra como a alocação de uma dada operação pode infactibilizar o posicionamento de uma outra no instante de início em que ela já havia sido alocada. Se isto ocorrer, a seqüência é infactível pois o algoritmo não permite que nenhuma operação já alocada seja desalocada.

Na seqüência proposta foi feita a alocação da primeira batelada de H nos instantes de início dados, e em seguida foram alocadas as operações 40, primeira operação de E1, e operações 36, 37 e 38, pertencentes a batelada do produto F, em instantes de inicio posteriores, de forma que seja possível alocar as primeiras operações da segunda batelada de H entre estas já alocadas.

```
Candidatas: 1->0  21->0  36->0  40->0  43->402
ITERACAO 1
OPERACAO:21   START TIME:110

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [110 118]  [117  63]  [117  72]  [125  80]  [135 114]
H2 [128 416]  [138 417]  [138 426]  [146 434]  [282 468]
H3 [146 482]  [159 483]  [159 492]  [167 500]  [327 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [  0 181]  [  8 184]  [ 13 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 1->0  22->117  36->0  40->0  43->402
ITERACAO 2
OPERACAO:22   START TIME:117

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [118 248]
D3 [118 406]  [122 421]  [122 421]  [149 431]  [159 444]
D4 [158 478]  [162 493]  [162 493]  [234 503]  [282 516]
H1 [110 118]  [117 119]  [117  72]  [125  80]  [135 114]
H2 [128 416]  [138 417]  [138 426]  [146 434]  [282 468]
H3 [146 482]  [159 483]  [159 492]  [167 500]  [327 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [  0 181]  [  8 184]  [ 13 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 23->117
ITERACAO 3
OPERACAO:23   START TIME:117

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [125 248]
D3 [124 406]  [128 421]  [128 421]  [149 431]  [159 444]
D4 [164 478]  [168 493]  [168 493]  [234 503]  [282 516]
H1 [110 118]  [117 119]  [117 128]  [125  80]  [135 114]
H2 [128 416]  [138 417]  [138 426]  [146 434]  [282 468]
H3 [146 482]  [159 483]  [159 492]  [167 500]  [327 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [  0 181]  [  8 184]  [ 13 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 1->0  24->125  26->128  36->0  40->0  43->402
ITERACAO 4
OPERACAO:24   START TIME:125

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [135 248]
D3 [124 406]  [128 421]  [128 421]  [149 431]  [159 444]
D4 [164 478]  [168 493]  [168 493]  [234 503]  [282 516]
H1 [110 118]  [117 119]  [117 128]  [125 136]  [135 114]
H2 [128 416]  [138 417]  [138 426]  [146 434]  [282 468]
H3 [146 482]  [159 483]  [159 492]  [167 500]  [327 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [  0 181]  [  8 184]  [ 13 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 1->0  25->135  26->128  36->0  40->0  43->402
ITERACAO 5
OPERACAO:25   START TIME:135

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [170 248]
D3 [124 406]  [128 421]  [128 421]  [149 431]  [184 444]
D4 [164 478]  [168 493]  [168 493]  [234 503]  [282 516]
H1 [110 118]  [117 119]  [117 128]  [125 136]  [135 170]
H2 [128 416]  [138 417]  [138 426]  [146 434]  [282 468]
H3 [146 482]  [159 483]  [159 492]  [167 500]  [327 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [  0 181]  [  8 184]  [ 13 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 1->0  26->128  36->0  40->0  43->402
ITERACAO 6
OPERACAO:40   START TIME:282

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [170 248]
D3 [124 406]  [128 421]  [128 421]  [149 431]  [184 444]
D4 [164 478]  [168 493]  [168 493]  [234 503]  [290 516]
H1 [110 118]  [117 119]  [117 128]  [125 136]  [135 170]
H2 [128 416]  [138 417]  [138 426]  [146 434]  [282 468]
H3 [341 482]  [348 483]  [348 492]  [402 500]  [412 534]
F  [  0 438]  [ 11 444]  [ 17 447]  [ 66 468]
E1 [282 293]  [290 184]  [295 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 1->0  26->128  36->0  41->290  43->402
ITERACAO 7
OPERACAO:36   START TIME:174

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [170 248]
D3 [124 406]  [128 421]  [128 421]  [149 431]  [184 444]
D4 [164 478]  [168 493]  [168 493]  [234 503]  [290 516]
H1 [110 118]  [117 119]  [117 128]  [125 136]  [135 170]
H2 [128 416]  [138 417]  [138 426]  [146 434]  [282 468]
H3 [341 482]  [348 483]  [348 492]  [402 500]  [412 534]
F  [174 186]  [185 444]  [191 447]  [290 468]
E1 [282 293]  [290 184]  [295 198]
E2 [402 429]  [410 432]  [414 444]
```

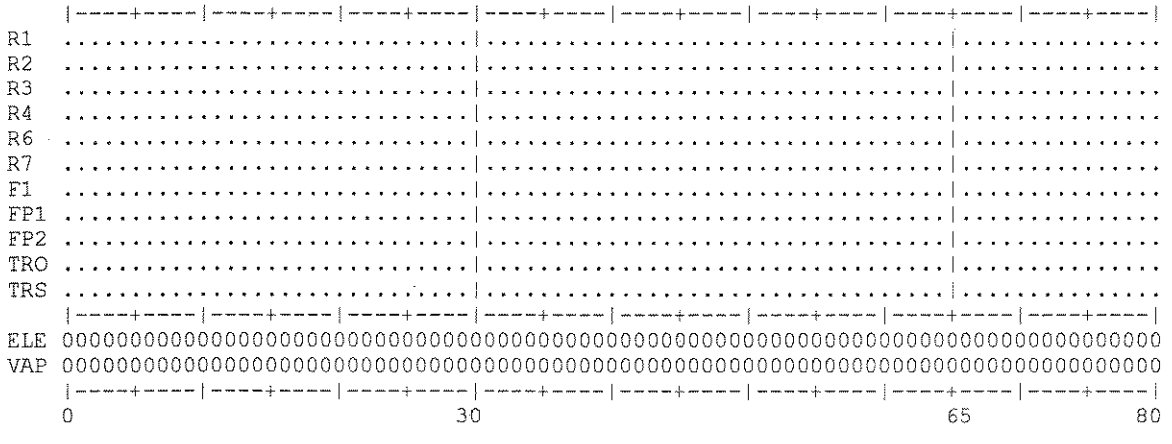
```
Candidatas: 1->0  26->128  37->185  41->290
ITERACAO 8
OPERACAO:37   START TIME:185

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [170 248]
D3 [124 406]  [128 421]  [128 421]  [149 431]  [184 444]
D4 [164 478]  [168 493]  [168 493]  [234 503]  [290 516]
H1 [110 118]  [117 119]  [117 128]  [125 136]  [135 170]
H2 [128 416]  [138 417]  [138 426]  [146 434]  [282 468]
H3 [341 482]  [348 483]  [348 492]  [402 500]  [412 534]
F  [174 186]  [185 192]  [191 447]  [290 468]
E1 [282 293]  [290 184]  [295 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 38->191
ITERACAO 9
OPERACAO:38   START TIME:191

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [170 248]
D3 [124 406]  [128 421]  [128 421]  [149 431]  [184 444]
D4 [164 478]  [168 493]  [168 493]  [234 503]  [290 516]
H1 [110 118]  [117 119]  [117 128]  [125 136]  [135 170]
H2 [128 416]  [138 417]  [138 426]  [146 434]  [282 468]
H3 [341 482]  [348 483]  [348 492]  [402 500]  [412 534]
F  [174 186]  [185 192]  [191 195]  [290 468]
E1 [282 293]  [290 184]  [295 198]
E2 [402 429]  [410 432]  [414 444]
```

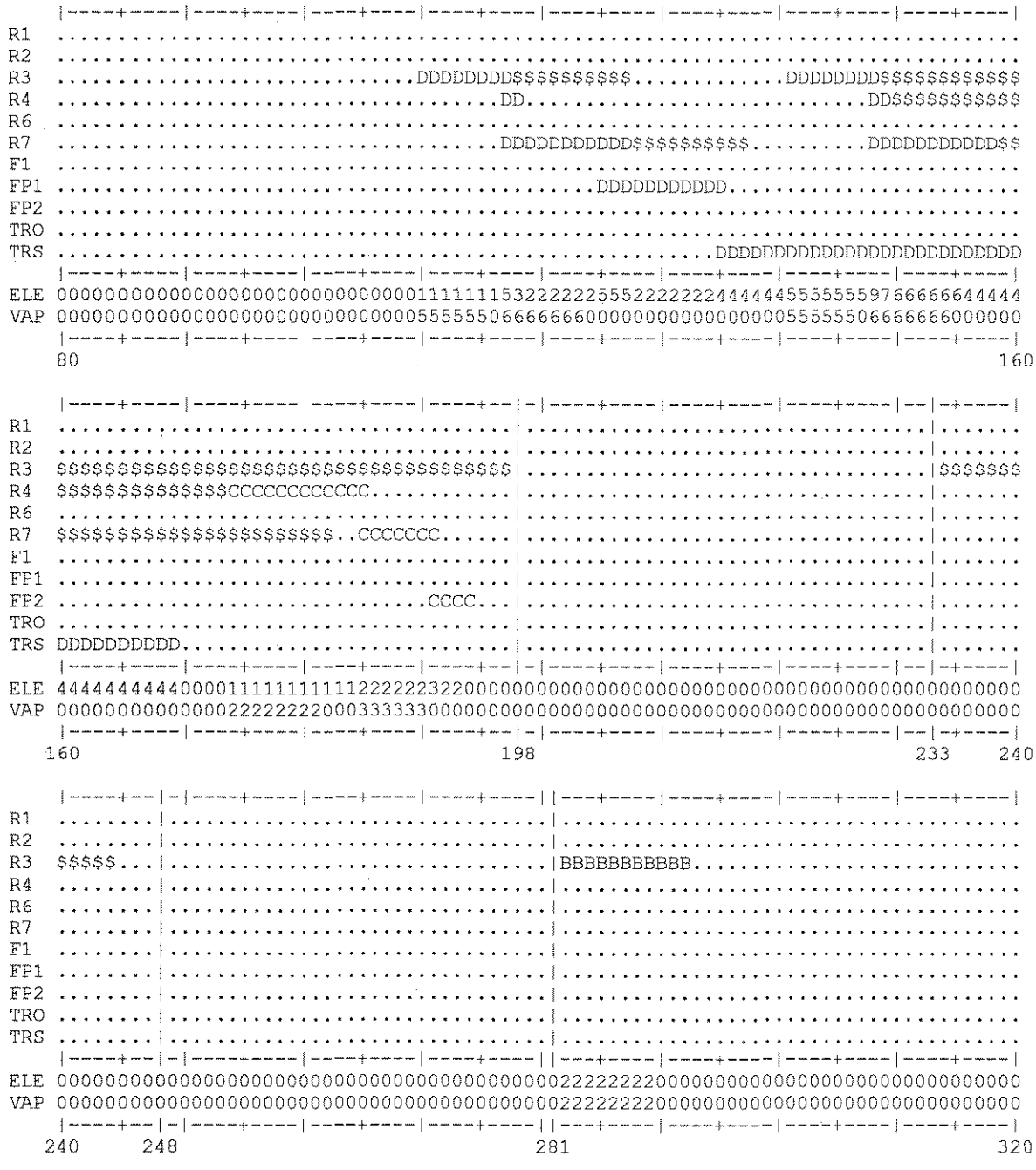
É iniciado o seqüenciamento da segunda batelada de H alocando a operação 26 entre as operações 21 e 40, a operação 27 entre as operações 22 e 36 e a operação 28 entre as operações 23 e 37. A carta de Gantt que representa o seqüenciamento até aqui é apresentada em seguida.

```
Candidatas: 1->0  26->128  39->290  41->290  43->402
ITERACAO 10
OPERACAO:26   START TIME:141

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [170 248]
D3 [124 406]  [128 421]  [128 421]  [149 431]  [184 444]
D4 [164 478]  [168 493]  [168 493]  [234 503]  [290 516]
H1 [110 118]  [117 119]  [117 128]  [125 136]  [135 170]
H2 [141 149]  [148 417]  [148 426]  [156 434]  [282 468]
H3 [341 482]  [348 483]  [348 492]  [402 500]  [412 534]
F  [174 186]  [185 192]  [191 195]  [290 468]
E1 [282 293]  [290 184]  [295 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 1->0  27->148  39->290  41->290  43->402
ITERACAO 11
OPERACAO:27   START TIME:148


D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [170 248]
D3 [124 406]  [128 421]  [128 421]  [149 431]  [184 444]
D4 [164 478]  [168 493]  [168 493]  [234 503]  [290 516]
H1 [110 118]  [117 119]  [117 128]  [125 136]  [135 170]
H2 [141 149]  [148 150]  [148 426]  [156 434]  [282 468]
H3 [341 482]  [348 483]  [348 492]  [402 500]  [412 534]
F  [174 186]  [185 192]  [191 195]  [290 468]
E1 [282 293]  [290 184]  [295 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 28->148
ITERACAO 12
OPERACAO:28   START TIME:148

D1 [  0  46]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [170 248]
D3 [124 406]  [128 421]  [128 421]  [156 431]  [184 444]
D4 [164 478]  [168 493]  [168 493]  [282 503]  [292 516]
H1 [110 118]  [117 119]  [117 128]  [125 136]  [135 170]
H2 [141 149]  [148 150]  [148 159]  [156 434]  [282 468]
H3 [341 482]  [348 483]  [348 492]  [402 500]  [412 534]
F  [174 186]  [185 192]  [191 195]  [290 468]
E1 [282 293]  [290 184]  [295 198]
E2 [402 429]  [410 432]  [414 444]
```

CARTA DE GANTT





A operação 41 é então alocada no início de sua janela, porém isto infactibiliza o processamento das operações já alocadas da segunda batelada de H, uma vez que a operação 41 sendo alocada neste instante, força a operação 29 a ser iniciada depois de sua execução pois não há tempo disponível entre as operações 24 e 41, nos instantes em que elas foram alocadas, para o processamento da operação 29 mais os tempos de preparação e limpeza entre as operações. A operação 29, tendo o seu instante de inicio mais cedo deslocado para depois da operação 41, obrigaria ao material processado na operação 28, alocada em t = 148, a aguardar

no equipamento até que pudesse ser transferido, só desocupando o processador em t = 335. Como a operação 37 já está alocada em t = 185, esta seqüência não é possível de ser realizada.

```
Candidatas: 1->0  29->156  39->290  41->290  43->402
ITERACAO 13
OPERACAO:41   START TIME:290

                FIM DO PROGRAMA: SEQUENCIA INFACTIVEL!
```

5 - CONCLUSÃO

Neste trabalho foi desenvolvida uma estratégia de simulação capaz de indicar, para um dado cenário, quais as próximas operações que devem ser alocadas para manter a factibilidade da solução, ao mesmo tempo em que determina qual o instante mais cedo em que as operações ainda não alocadas podem ser iniciadas. Este procedimento visa a minimização do tempo total de produção como o critério de desempenho a ser otimizado. O simulador foi desenvolvido com o objetivo de ser incorporado a uma estratégia tipo BAB.

No caso de plantas multiprodutos cuja solução não esteja restrita às seqüências de permutação, o número de ramos candidatos a participar da solução parcial do nó é, em geral, muito grande. Pode-se, é claro, recorrer à estratégias heurísticas para reduzir o espaço de busca, neste caso abdicando-se não só da otimalidade, mas podendo também prejudicar a factibilidade.

Egli e Rippin propuseram uma estratégia deste tipo onde não se garantia a obtenção da solução ótima para o problema, mas era escolhida dentre algumas soluções factíveis, determinadas segundo um critério que levava em conta minimização dos estoques dos produtos, qual a que apresentava menor custo total de produção. Na estratégia utilizada por estes autores, a abertura da árvore de busca era efetuada por tarefa e não por operação.

A utilidade do simulador desenvolvido é evidenciada pela redução significativa de nós candidatos obtida a partir da análise do conjunto de restrições que compõem o problema, mantendo a factibilidade. Fica evidente que, durante este processo, pode acontecer de não haver solução factível. Neste caso, é permitida a intervenção do usuário buscando-se a melhor solução de compromisso.

Como proposta de continuidade deste trabalho, sugere-se o desenvolvimento de uma função de custo para otimizar a busca automática de soluções factíveis, utilizando-se a estratégia adotada no desenvolvimento do simulador de busca orientada por restrições para redução do espaço de busca pesquisado.

APÊNDICE 1

DADOS DO PROBLEMA:

Parâmetros

Nº Tarefas	Nº Operações	Nº Processad.	Nº Recursos	Horizonte (h)
10	45	11	2	516

Índices

TAREFAS (TAR[i])	D1	D2	D3	D4	H1	H2	H3	F	E1	E2
ÍNDICES	1	2	3	4	5	6	7	8	9	10

PROCES.(PRO[i])	R1	R2	R3	R4	R6	R7	F1	FP1	FP2	TRO	TRS
ÍNDICES	1	2	3	4	5	6	7	8	9	10	11

RECURSOS	ENER. ELÉT.	VAPOR
ÍNDICE	1	2

Quantidade de produto por batelada (Batch Size).

PRODUTO	BATCH SIZE (Kg)
D	400
E1	300
E2	240
F	600
H	300

Tempos de processamento, tempos de transferência e condições de estabilidade

0 → intermediário estável

1 → intermediário instável

1- Produto D

Operação	Índice das oper(*).	TP	TT	INST
D.R1	1, 6, 11, 15	4	1	1
D.R2	2, 7, 12, 16	14	1	0
D.R6	3, 8, 13, 17	14	1	0
D.F1	4, 9, 14, 18	9	1	0
D.TRO	5, 10, 15, 20	13	0	0

(*) representa os índices de cada operação das bateladas de D1 a D4

2- Produto H

Operação	Índice das oper(*).	TP	TT	INST
H.R3	21, 26, 31	7	1	0
H.R4	22, 27, 32	0	1	1
H.R7	23, 28, 33	7	3	0
H.FP1	24, 29, 34	7	1	0
H.TRS	25, 30, 35	34	0	0

(*)representa os índices de cada operação das bateladas de H1 a H3

3- Produto F

Operação	Índice das oper.	TP	TT	INST
F.R4	36	11	1	0
F.R7	37	5	1	1
F.FP2	38	2	1	0
F.TRS	39	21	0	0

4- Produto E1

Operação	Índice das oper.	TP	TT	INST
E1.R3	40	8	3	0
E1.FP1	41	2	1	1
E1.TRS	42	14	0	0

5- Produto E2

Operação	Índice das oper.	TP	TT	INST
E2.R4	43	8	2	0
E2.FP1	44	2	1	1
E2.TRS	45	12	0	0

Setup entre tarefas por processador

R1

DE	PARA	SETUP
D	D	24

R2

DE	PARA	SETUP
D	D	24

R3

DE	PARA	SETUP
H	H	10
H	E	60
E	H	48
E	E	10

R4

DE	PARA	SETUP
H	H	0
H	F	24
H	E	12
F	H	20
F	F	10
F	E	16
E	H	48
E	F	20
E	E	10

R6

DE	PARA	SETUP
D	D	24

R7

DE	PARA	SETUP
H	H	10
H	F	24
F	H	12
F	F	8

F1

DE	PARA	SETUP
D	D	36

FP1

DE	PARA	SETUP
H	H	0
H	E	30
E	H	36
E	E	24

FP2

DE	PARA	SETUP
F	F	12

TRO

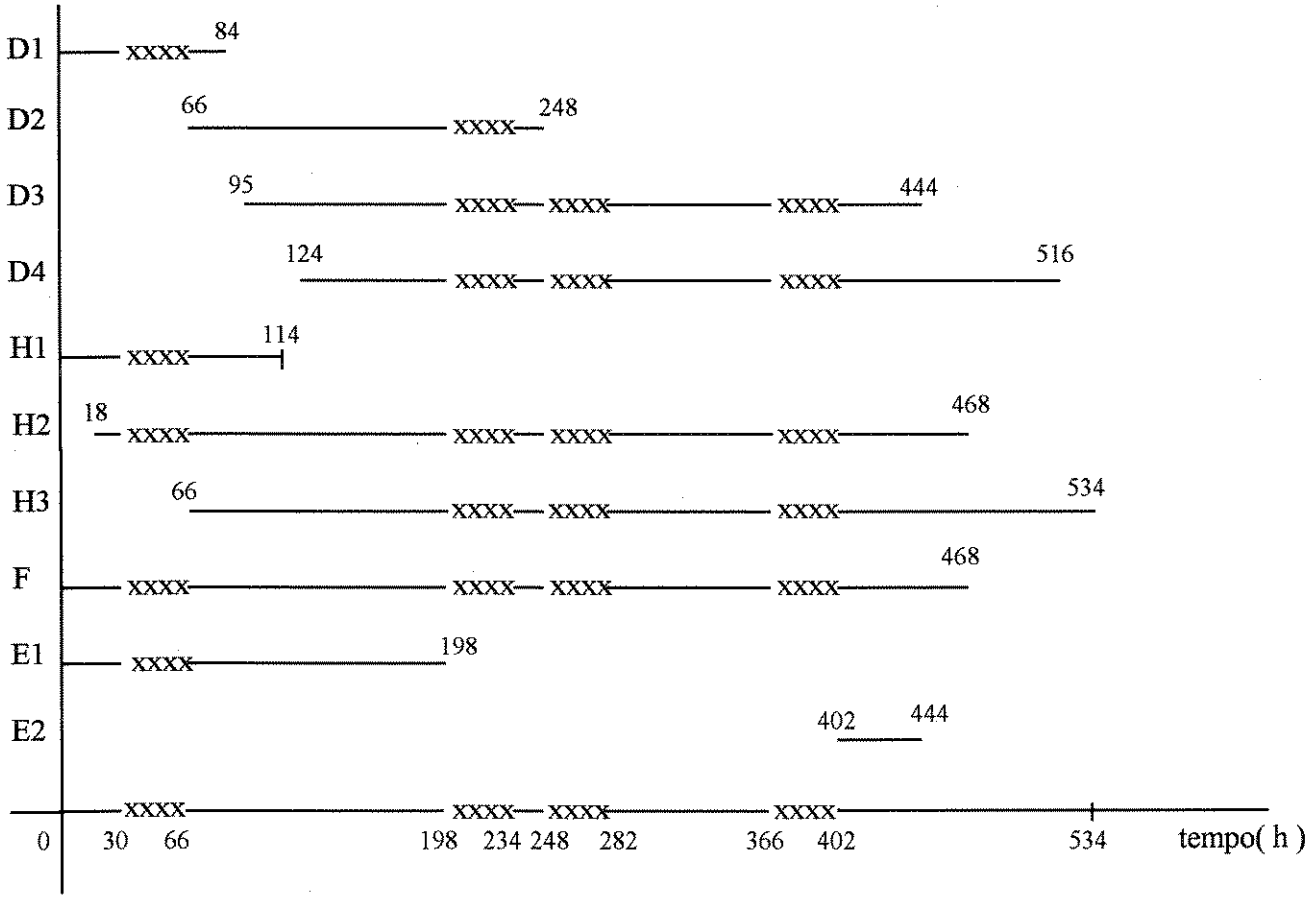
DE	PARA	SETUP
D	D	0

TRS

DE	PARA	SETUP
H	H	10
H	F	24
H	E	24
F	H	20
F	F	6
F	E	20
E	H	16
E	F	16
E	E	8

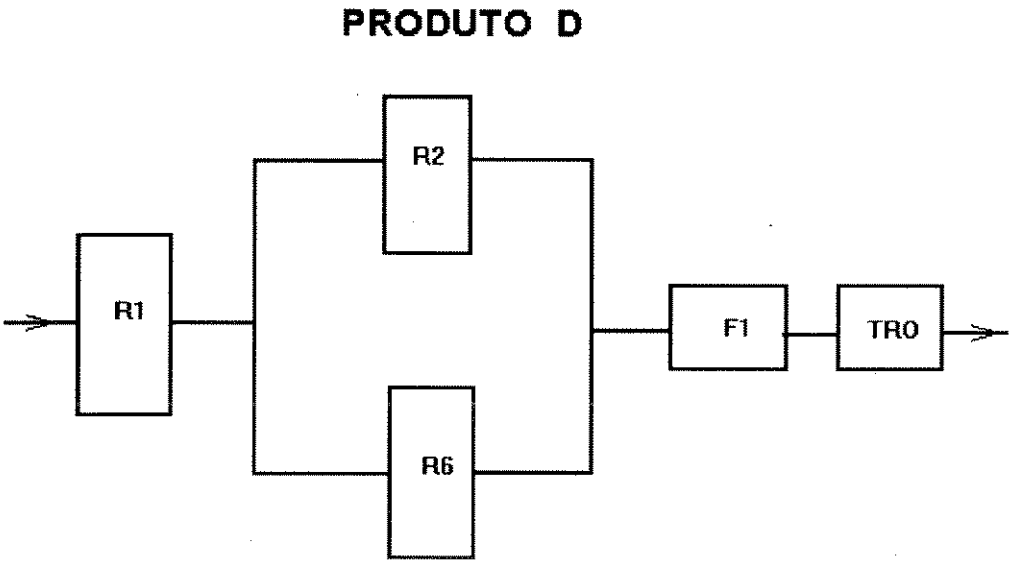
Janelas de tempo de cada batelada

obs: o símbolo "x" representa as indisponibilidades de tempo estabelecidas em virtude da jornada de trabalho padrão.



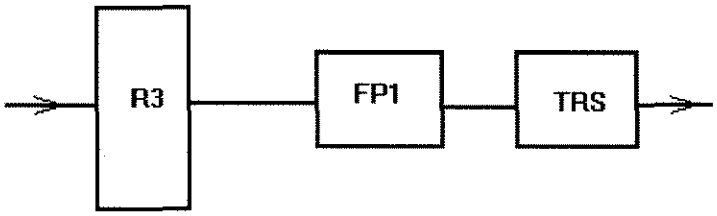
APÊNDICE 2

Rotas dos produtos



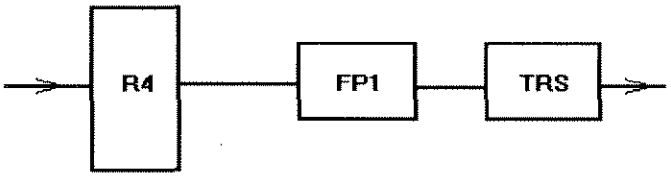
R1 - REATOR 1
R2 - REATOR 2
R6 - REATOR 6
F1 - FILTRO 1
TRO - SECADOR

PRODUTO E - variante 1



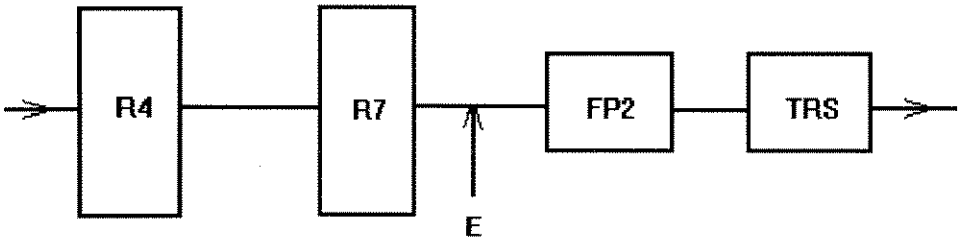
R3 - REATOR 3
FP1 - FILTRO PRENSA 1
TRS - CABINE DE SECAGEM

PRODUTO E - variante 2



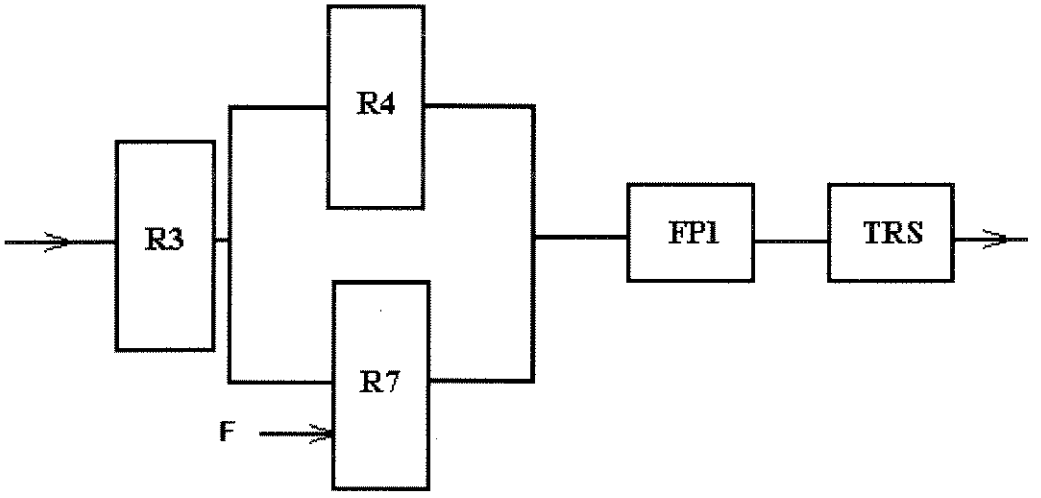
R4 - REATOR 4
FP1 - FILTRO PRENSA 1
TRS - CABINE DE SECAGEM

PRODUTO F



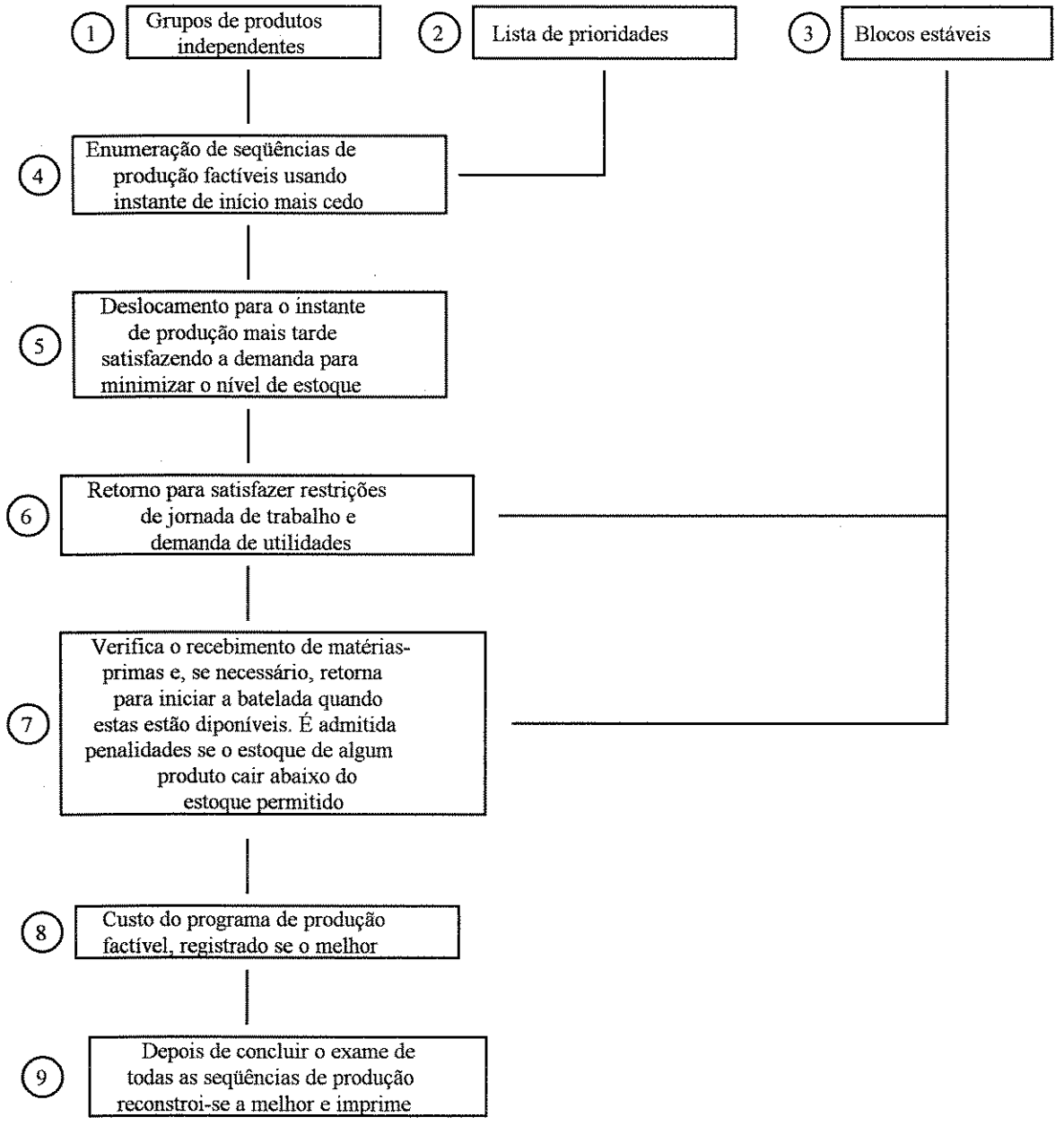
R4 - REATOR 4
R7 - REATOR 7
FP2 - FILTRO PRENSA 2
TRS - CABINE DE SECAGEM

PRODUTO H

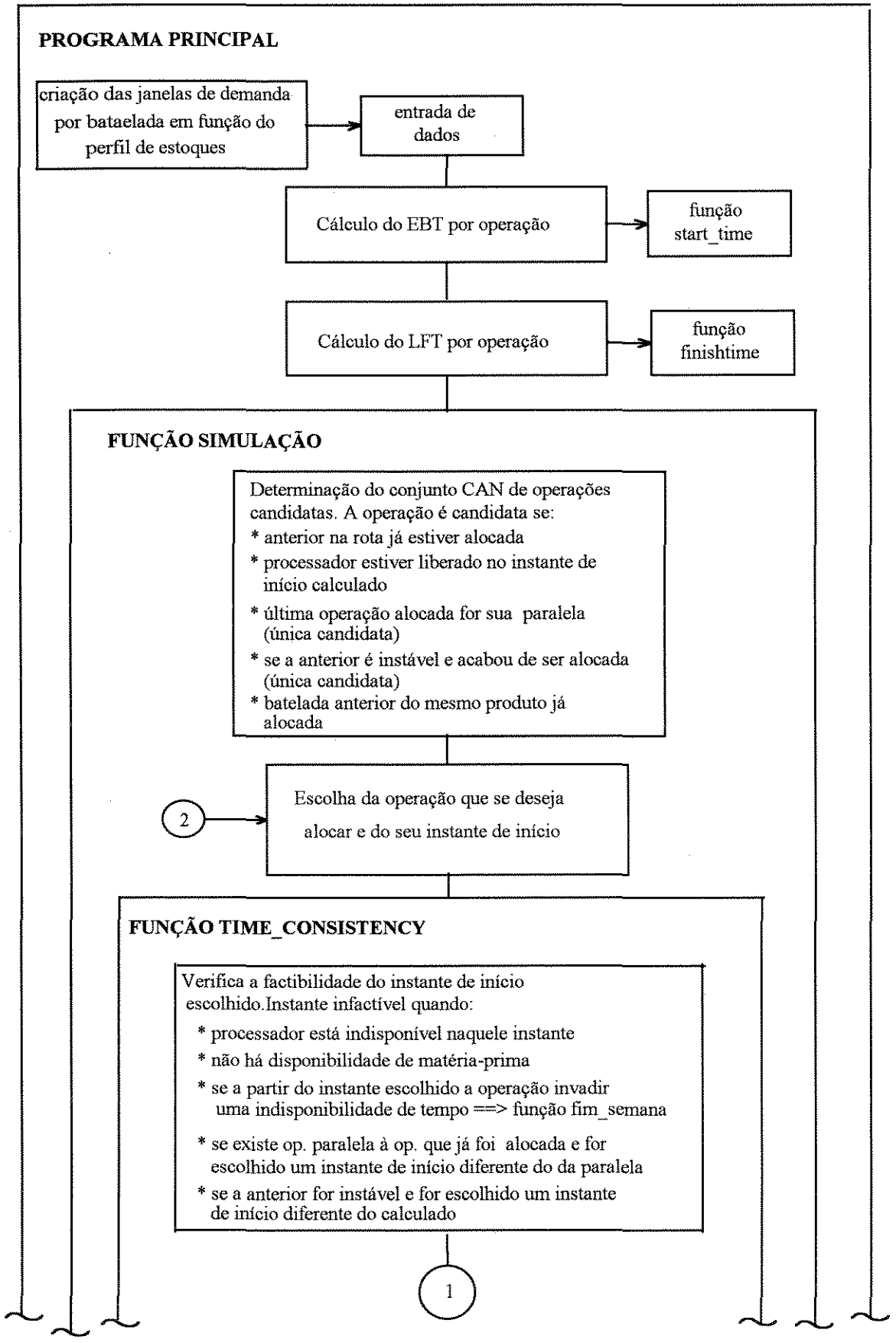


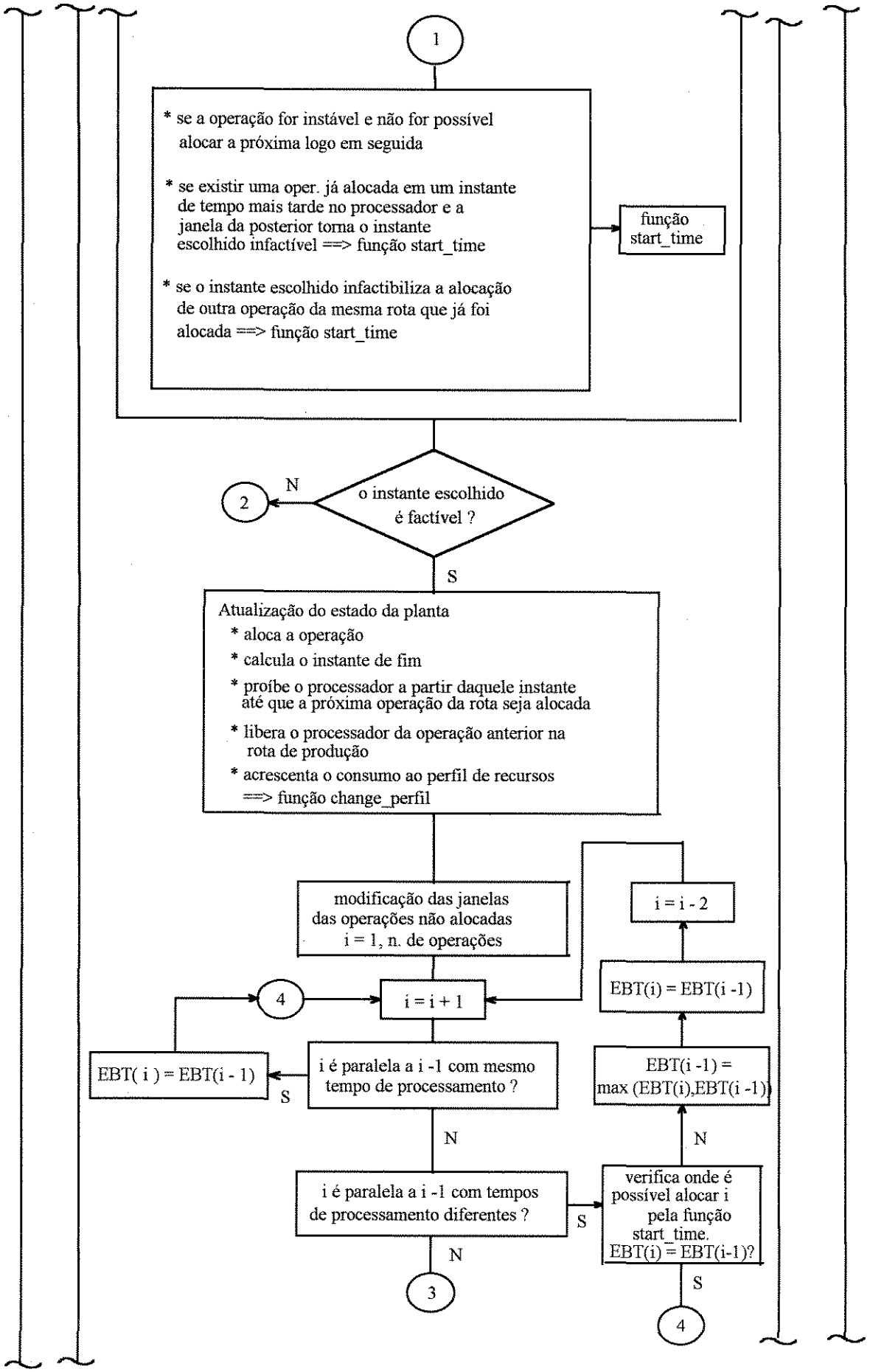
R3 - REATOR 3
R4 - REATOR 4
R7 - REATOR 7
FP1 - FILTRO PRENSA 1
TRS - CABINE DE SECAGEM

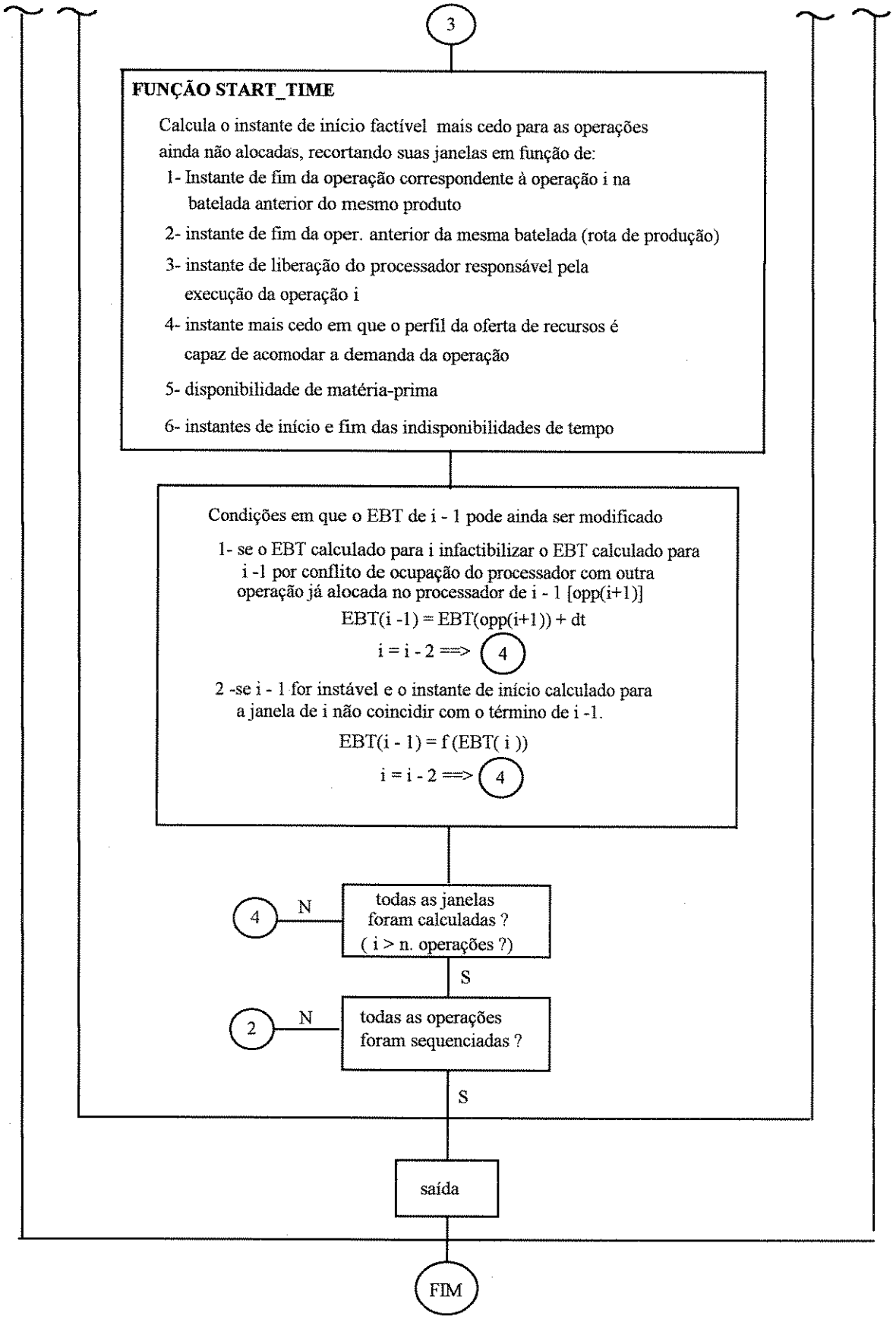
APÊNDICE 3


1
Grupos de produtos independentes
2
Lista de prioridades
3
Blocos estáveis
4
Enumeração de seqüências de produção factíveis usando instante de início mais cedo
5
Deslocamento para o instante de produção mais tarde satisfazendo a demanda para minimizar o nível de estoque
6
Retorno para satisfazer restrições de jornada de trabalho e demanda de utilidades
7
Verifica o recebimento de matérias-primas e, se necessário, retorna para iniciar a batelada quando estas estão diponíveis. É admitida penalidades se o estoque de algum produto cair abaixo do estoque permitido
8
Custo do programa de produção factível, registrado se o melhor
9
Depois de concluir o exame de todas as seqüências de produção reconstroi-se a melhor e imprime

APÊNDICE 4


PROGRAMA PRINCIPAL
criação das janelas de demanda por bataelada em função do perfil de estoques
entrada de dados
Cálculo do EBT por operação
função start_time
Cálculo do LFT por operação
função finishtime
FUNÇÃO SIMULAÇÃO
Determinação do conjunto CAN de operações candidatas. A operação é candidata se:
* anterior na rota já estiver alocada
* processador estiver liberado no instante de início calculado
* última operação alocada for sua paralela (única candidata)
* se a anterior é instável e acabou de ser alocada (única candidata)
* batelada anterior do mesmo produto já alocada
2
Escolha da operação que se deseja alocar e do seu instante de início
FUNÇÃO TIME_CONSISTENCY
Verifica a factibilidade do instante de início escolhido.Instante infactível quando:
* processador está indisponível naquele instante
* não há disponibilidade de matéria-prima
* se a partir do instante escolhido a operação invadir uma indisponibilidade de tempo ==> função fim_semana
* se existe op. paralela à op. que já foi alocada e for escolhido um instante de início diferente do da paralela
* se a anterior for instável e for escolhido um instante de início diferente do calculado
1


1
* se a operação for instável e não for possível alocar a próxima logo em seguida
* se existir uma oper. já alocada em um instante de tempo mais tarde no processador e a janela da posterior torna o instante escolhido infactível ==> função start_time
* se o instante escolhido infactibiliza a alocação de outra operação da mesma rota que já foi alocada ==> função start_time
função start_time
2
N
o instante escolhido é factível ?
S
Atualização do estado da planta
* aloca a operação
* calcula o instante de fim
* proíbe o processador a partir daquele instante até que a próxima operação da rota seja alocada
* libera o processador da operação anterior na rota de produção
* acrescenta o consumo ao perfil de recursos ==> função change_perfil
modificação das janelas das operações não alocadas i = 1, n. de operações
4
i = i + 1
EBT(i) = EBT(i - 1)
S
i é paralela a i -1 com mesmo tempo de processamento ?
N
i é paralela a i -1 com tempos de processamento diferentes ?
N
3
S
verifica onde é possível alocar i pela função start_time. EBT(i) = EBT(i-1)?
N
EBT(i -1) = max (EBT(i),EBT(i -1))
EBT(i) = EBT(i -1)
i = i - 2
S
4


3
FUNÇÃO START_TIME
Calcula o instante de início factível mais cedo para as operações ainda não alocadas, recortando suas janelas em função de:
1- Instante de fim da operação correspondente à operação i na batelada anterior do mesmo produto
2- instante de fim da oper. anterior da mesma batelada (rota de produção)
3- instante de liberação do processador responsável pela execução da operação i
4- instante mais cedo em que o perfil da oferta de recursos é capaz de acomodar a demanda da operação
5- disponibilidade de matéria-prima
6- instantes de início e fim das indisponibilidades de tempo
Condições em que o EBT de i - 1 pode ainda ser modificado
1- se o EBT calculado para i infactibilizar o EBT calculado para i -1 por conflito de ocupação do processador com outra operação já alocada no processador de i - 1 [opp(i+1)]
EBT(i -1) = EBT(opp(i+1)) + dt
i = i - 2 ==> 4
2 -se i - 1 for instável e o instante de início calculado para a janela de i não coincidir com o término de i -1.
EBT(i - 1) = f (EBT(i))
i = i - 2 ==> 4
4
N
todas as janelas foram calculadas ? (i > n. operações ?)
S
2
N
todas as operações foram sequenciadas ?
S
saída
FIM

SEQÜÊNCIA COMPLETA

```
Candidatas: 1->0  21->0  36->0  40->0  43->402
ITERACAO 1
OPERACAO:1   START TIME:0

D1 [  0   5]  [  4  61]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  4  62]  [ 11  63]  [ 11  72]  [ 19  80]  [ 66 114]
H2 [ 22 416]  [ 68 417]  [ 68 426]  [ 76 434]  [111 468]
H3 [ 76 482]  [ 89 483]  [ 89 492]  [ 97 500]  [156 534]
F  [  4 438]  [ 15 444]  [ 21 447]  [ 66 468]
E1 [  4 181]  [ 66 184]  [ 71 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 2->4
ITERACAO 2
OPERACAO:2   START TIME:4

D1 [  0   5]  [  4  20]  [  4  61]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  4  62]  [ 11  63]  [ 11  72]  [ 19  80]  [ 66 114]
H2 [ 22 416]  [ 68 417]  [ 68 426]  [ 76 434]  [111 468]
H3 [ 76 482]  [ 89 483]  [ 89 492]  [ 97 500]  [156 534]
F  [  4 438]  [ 15 444]  [ 21 447]  [ 66 468]
E1 [  4 181]  [ 66 184]  [ 71 198]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 3->4
ITERACAO 3
OPERACAO:3   START TIME:4

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [  4  62]  [ 11  63]  [ 11  72]  [ 19  80]  [ 66 114]
H2 [ 22 416]  [ 68 417]  [ 68 426]  [ 76 434]  [111 468]
H3 [ 76 482]  [ 89 483]  [ 89 492]  [ 97 500]  [156 534]
F  [  4 438]  [ 15 444]  [ 21 447]  [ 66 468]
E1 [  4 181]  [ 66 184]  [ 71 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 4->19  21->4  36->4  40->4  43->402
ITERACAO 4
OPERACAO:21   START TIME:12


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  71]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 12  20]  [ 19  63]  [ 19  72]  [ 66  80]  [ 76 114]
H2 [ 66 416]  [ 76 417]  [ 76 426]  [ 84 434]  [121 468]
H3 [ 84 482]  [ 97 483]  [ 97 492]  [105 500]  [282 534]
F  [  4 438]  [ 15 444]  [ 21 447]  [ 66 468]
E1 [116 181]  [124 184]  [129 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 4->19  22->19  36->4  43->402
ITERACAO 5
OPERACAO:4   START TIME:19


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 12  20]  [ 19  63]  [ 19  72]  [ 66  80]  [ 76 114]
H2 [ 66 416]  [ 76 417]  [ 76 426]  [ 84 434]  [121 468]
H3 [ 84 482]  [ 97 483]  [ 97 492]  [105 500]  [282 534]
F  [  4 438]  [ 15 444]  [ 21 447]  [ 66 468]
E1 [116 181]  [124 184]  [129 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 5->66  6->76  22->19  36->4  43->402
ITERACAO 6
OPERACAO:22   START TIME:19


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  72]  [ 66  80]  [ 76 114]
H2 [ 66 416]  [ 76 417]  [ 76 426]  [ 84 434]  [121 468]
H3 [ 84 482]  [ 97 483]  [ 97 492]  [105 500]  [282 534]
F  [ 81 438]  [ 92 444]  [ 98 447]  [101 468]
E1 [116 181]  [124 184]  [129 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 23->19
ITERACAO 7
OPERACAO:23   START TIME:19


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 66  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 66  80]  [ 76 114]
H2 [ 66 416]  [ 76 417]  [ 76 426]  [ 84 434]  [121 468]
H3 [ 84 482]  [ 97 483]  [ 97 492]  [105 500]  [282 534]
F  [ 81 438]  [ 92 444]  [ 98 447]  [101 468]
E1 [116 181]  [124 184]  [129 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 5->66  6->76  24->66  26->66  36->81  40->116  43->402
ITERACAO 8
OPERACAO:24   START TIME:69


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 79  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [102 235]  [112 248]
D3 [116 406]  [120 421]  [120 421]  [149 431]  [159 444]
D4 [156 478]  [160 493]  [160 493]  [234 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [ 66 416]  [ 82 417]  [ 82 426]  [ 90 434]  [124 468]
H3 [ 84 482]  [103 483]  [103 492]  [111 500]  [282 534]
F  [ 81 438]  [ 96 444]  [102 447]  [105 468]
E1 [116 181]  [124 184]  [129 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 5->79  6->76  25->79  26->66  36->81  40->116  43->402
ITERACAO 9
OPERACAO:5   START TIME:70


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 76 210]  [ 80 225]  [ 80 225]  [107 235]  [117 248]
D3 [116 406]  [120 421]  [120 421]  [154 431]  [164 444]
D4 [156 478]  [160 493]  [160 493]  [237 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [ 72 416]  [ 84 417]  [ 84 426]  [ 92 434]  [124 468]
H3 [ 90 482]  [105 483]  [105 492]  [113 500]  [282 534]
F  [ 81 438]  [ 96 444]  [102 447]  [105 468]
E1 [116 181]  [124 184]  [129 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 6->76  25->79  26->72  36->81  40->116  43->402
ITERACAO 10
OPERACAO:25   START TIME:79


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84 210]  [ 88 225]  [ 88 225]  [107 235]  [117 248]
D3 [124 406]  [128 421]  [128 421]  [154 431]  [164 444]
D4 [164 478]  [168 493]  [168 493]  [237 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [ 72 416]  [ 84 417]  [ 84 426]  [ 92 434]  [124 468]
H3 [ 90 482]  [105 483]  [105 492]  [113 500]  [282 534]
F  [ 84 438]  [ 96 444]  [102 447]  [138 468]
E1 [116 181]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 6->84  26->72  36->84  40->116  43->402
ITERACAO 11
OPERACAO:6   START TIME:84


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 225]  [ 88 225]  [107 235]  [117 248]
D3 [124 406]  [128 421]  [128 421]  [154 431]  [164 444]
D4 [164 478]  [168 493]  [168 493]  [237 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [ 72 416]  [ 88 417]  [ 88 426]  [ 96 434]  [124 468]
H3 [ 90 482]  [109 483]  [109 492]  [117 500]  [282 534]
F  [ 88 438]  [ 99 444]  [105 447]  [138 468]
E1 [116 181]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 7->88
ITERACAO 12
OPERACAO:7   START TIME:88


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 225]  [107 235]  [117 248]
D3 [124 406]  [128 421]  [128 421]  [154 431]  [164 444]
D4 [164 478]  [168 493]  [168 493]  [237 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [ 72 416]  [ 94 417]  [ 94 426]  [102 434]  [124 468]
H3 [ 90 482]  [115 483]  [115 492]  [123 500]  [282 534]
F  [ 88 438]  [ 99 444]  [105 447]  [138 468]
E1 [116 181]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 8->88
ITERACAO 13
OPERACAO:8   START TIME:88


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 235]  [117 248]
D3 [124 406]  [128 421]  [128 421]  [154 431]  [164 444]
D4 [164 478]  [168 493]  [168 493]  [237 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [ 72 416]  [ 94 417]  [ 94 426]  [102 434]  [124 468]
H3 [ 90 482]  [115 483]  [115 492]  [123 500]  [282 534]
F  [ 88 438]  [ 99 444]  [105 447]  [138 468]
E1 [116 181]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 9->107  26->72  36->88  40->116  43->402
ITERACAO 14
OPERACAO:36   START TIME:88


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 235]  [117 248]
D3 [124 406]  [128 421]  [128 421]  [154 431]  [164 444]
D4 [164 478]  [168 493]  [168 493]  [237 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [ 72 416]  [120 417]  [120 426]  [128 434]  [138 468]
H3 [ 94 482]  [141 483]  [141 492]  [149 500]  [282 534]
F  [ 88 100]  [ 99 444]  [105 447]  [138 468]
E1 [116 181]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 9->107  26->72  37->99  40->116
ITERACAO 15
OPERACAO:9   START TIME:107


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [117 248]
D3 [128 406]  [132 421]  [132 421]  [154 431]  [164 444]
D4 [168 478]  [172 493]  [172 493]  [237 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [ 72 416]  [120 417]  [120 426]  [128 434]  [138 468]
H3 [ 94 482]  [141 483]  [141 492]  [149 500]  [282 534]
F  [ 88 100]  [108 444]  [114 447]  [138 468]
E1 [116 181]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 10->117  11->128  26->72  37->108  40->116
ITERACAO 16
OPERACAO:37   START TIME:108


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [117 248]
D3 [128 406]  [132 421]  [132 421]  [154 431]  [164 444]
D4 [168 478]  [172 493]  [172 493]  [237 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [ 72 416]  [129 417]  [129 426]  [137 434]  [147 468]
H3 [ 94 482]  [150 483]  [150 492]  [158 500]  [282 534]
F  [ 88 100]  [108 115]  [114 447]  [138 468]
E1 [116 181]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 38->114
ITERACAO 17
OPERACAO:38   START TIME:114


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [117 248]
D3 [128 406]  [132 421]  [132 421]  [154 431]  [164 444]
D4 [168 478]  [172 493]  [172 493]  [237 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [ 72 416]  [129 417]  [129 426]  [137 434]  [147 468]
H3 [ 94 482]  [150 483]  [150 492]  [158 500]  [282 534]
F  [ 88 100]  [108 115]  [114 118]  [138 468]
E1 [116 181]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 10->117  11->128  26->72  39->138  40->116  43->402
ITERACAO 18
OPERACAO:40   START TIME:116


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 248]
D3 [128 406]  [132 421]  [132 421]  [154 431]  [164 444]
D4 [168 478]  [172 493]  [172 493]  [237 503]  [282 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [175 416]  [182 417]  [182 426]  [234 434]  [282 468]
H3 [234 482]  [282 483]  [282 492]  [290 500]  [327 534]
F  [ 88 100]  [108 115]  [114 118]  [138 468]
E1 [116 127]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 10->124  11->128  39->138  41->133  43->402
ITERACAO 19
OPERACAO:10   START TIME:124


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 406]  [132 421]  [132 421]  [161 431]  [171 444]
D4 [168 478]  [172 493]  [172 493]  [282 503]  [292 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [175 416]  [182 417]  [182 426]  [234 434]  [282 468]
H3 [234 482]  [282 483]  [282 492]  [290 500]  [327 534]
F  [ 88 100]  [108 115]  [114 118]  [138 468]
E1 [116 127]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 11->128  39->138  41->133  43->402
ITERACAO 20
OPERACAO:11   START TIME:128


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 421]  [132 421]  [161 431]  [171 444]
D4 [168 478]  [172 493]  [172 493]  [282 503]  [292 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [175 416]  [182 417]  [182 426]  [234 434]  [282 468]
H3 [234 482]  [282 483]  [282 492]  [290 500]  [327 534]
F  [ 88 100]  [108 115]  [114 118]  [138 468]
E1 [116 127]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 12->132
ITERACAO 21
OPERACAO:12   START TIME:132


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 421]  [161 431]  [171 444]
D4 [168 478]  [172 493]  [172 493]  [282 503]  [292 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [175 416]  [182 417]  [182 426]  [234 434]  [282 468]
H3 [234 482]  [282 483]  [282 492]  [290 500]  [327 534]
F  [ 88 100]  [108 115]  [114 118]  [138 468]
E1 [116 127]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 13->132
ITERACAO 22
OPERACAO:13   START TIME:132


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 431]  [171 444]
D4 [168 478]  [172 493]  [172 493]  [282 503]  [292 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [175 416]  [182 417]  [182 426]  [234 434]  [282 468]
H3 [234 482]  [282 483]  [282 492]  [290 500]  [327 534]
F  [ 88 100]  [108 115]  [114 118]  [138 468]
E1 [116 127]  [133 184]  [138 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 14->161  39->138  41->133  43->402
ITERACAO 23
OPERACAO:41   START TIME:133


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 431]  [171 444]
D4 [168 478]  [172 493]  [172 493]  [282 503]  [292 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 416]  [234 417]  [234 426]  [282 434]  [292 468]
H3 [238 482]  [289 483]  [289 492]  [297 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [138 468]
E1 [116 127]  [133 139]  [138 198]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 42->138
ITERACAO 24
OPERACAO:42   START TIME:138


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 431]  [171 444]
D4 [168 478]  [172 493]  [172 493]  [282 503]  [292 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 416]  [234 417]  [234 426]  [282 434]  [292 468]
H3 [238 482]  [289 483]  [289 492]  [297 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [169 468]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 14->161  26->184  39->169  43->402
ITERACAO 25
OPERACAO:14   START TIME:161

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [171 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 416]  [234 417]  [234 426]  [282 434]  [292 468]
H3 [238 482]  [289 483]  [289 492]  [297 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 468]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 15->171  16->282  26->184  39->171  43->402
ITERACAO 26
OPERACAO:26   START TIME:184

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [171 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [234 417]  [234 426]  [282 434]  [292 468]
H3 [238 482]  [289 483]  [289 492]  [297 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 468]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 15->171  16->282  27->234  39->171  43->402
ITERACAO 27
OPERACAO:39   START TIME:171

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 444]
D4 [282 478]  [286 493]  [286 493]  [301 503]  [311 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [234 417]  [234 426]  [282 434]  [292 468]
H3 [238 482]  [289 483]  [289 492]  [297 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 15->234  16->282  27->234  43->402
ITERACAO 28
OPERACAO:15   START TIME:234


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [282 478]  [286 493]  [286 493]  [305 503]  [315 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 417]  [282 426]  [290 434]  [300 468]
H3 [238 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 16->282  27->282  43->402
ITERACAO 29
OPERACAO:27   START TIME:282


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [283 478]  [287 493]  [287 493]  [305 503]  [315 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 426]  [290 434]  [300 468]
H3 [293 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 28->282
ITERACAO 30
OPERACAO:28   START TIME:282


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 478]  [293 493]  [293 493]  [308 503]  [318 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 434]  [300 468]
H3 [293 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 16->289  29->290  31->293  43->402
ITERACAO 31
OPERACAO:29   START TIME:290

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 478]  [293 493]  [293 493]  [308 503]  [318 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 468]
H3 [293 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 16->289  30->300  31->293  43->402
ITERACAO 32
OPERACAO:30   START TIME:300

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 478]  [293 493]  [293 493]  [308 503]  [335 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 16->289  31->293  43->402
ITERACAO 33
OPERACAO:16   START TIME:289

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 493]  [293 493]  [308 503]  [335 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]
```

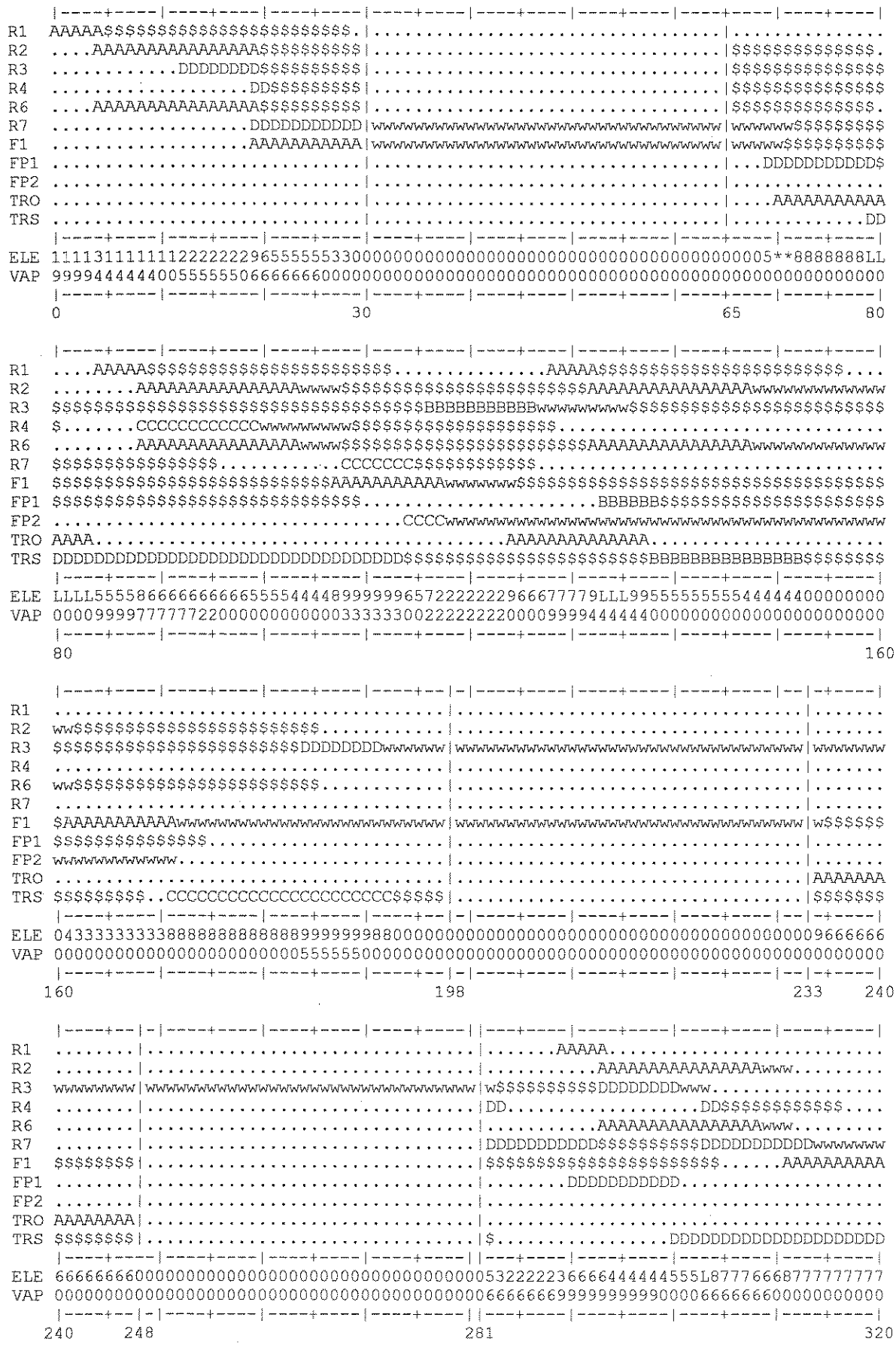
```
Candidatas: 17->293
ITERACAO 34
OPERACAO:17   START TIME:293


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 493]  [308 503]  [335 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 482]  [303 483]  [303 492]  [311 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 18->293
ITERACAO 35
OPERACAO:18   START TIME:293


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [308 503]  [335 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 482]  [303 483]  [303 492]  [311 500]  [402 534]
F [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 19->308  31->293  43->402
ITERACAO 36
OPERACAO:31   START TIME:293


D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [308 503]  [335 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 301]  [303 483]  [303 492]  [311 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]
```

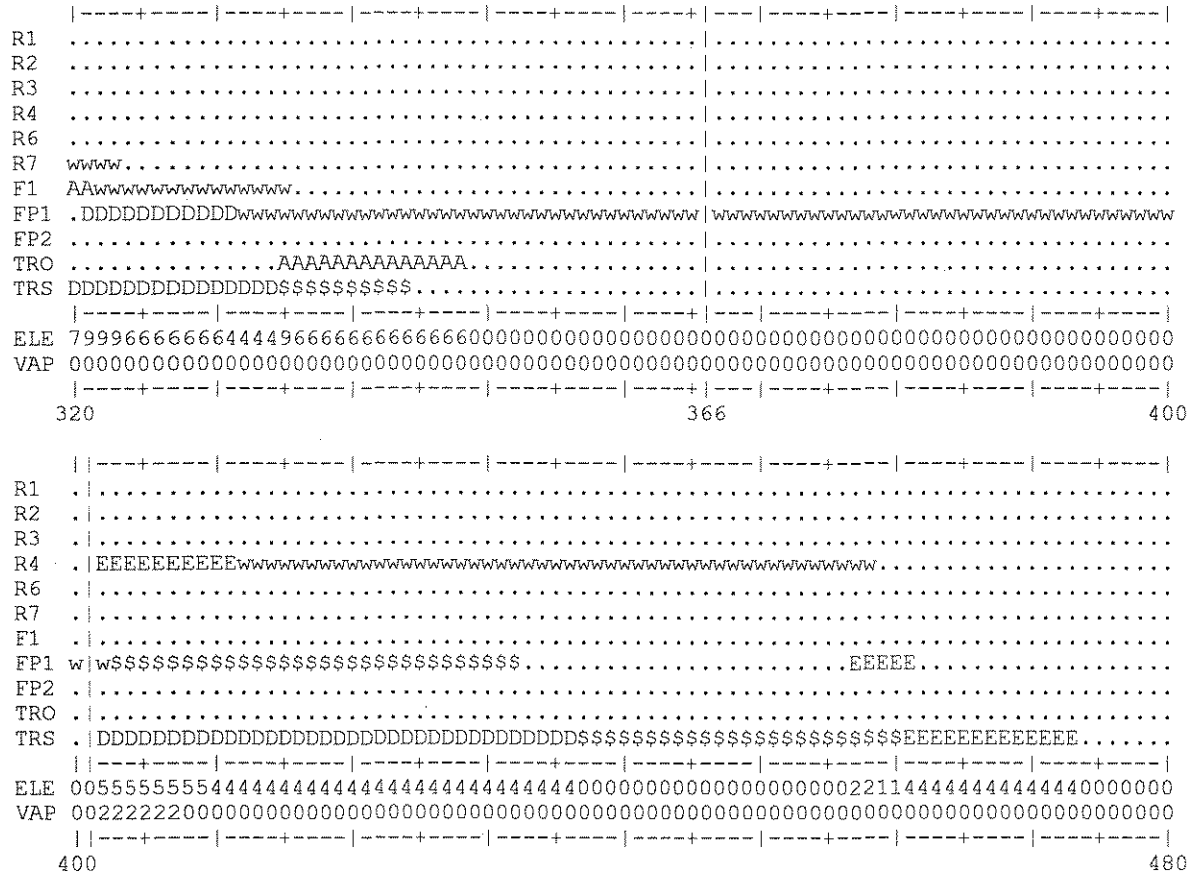
```
Candidatas: 19->308  32->303  43->402
ITERACAO 37
OPERACAO:32   START TIME:303

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [308 503]  [335 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 301]  [303 305]  [303 492]  [311 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 33->303
ITERACAO 38
OPERACAO:33   START TIME:303

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [311 503]  [335 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 301]  [303 305]  [303 314]  [311 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]


Candidatas: 19->311  34->311  43->402
ITERACAO 39
OPERACAO:19   START TIME:311

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [311 322]  [335 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 301]  [303 305]  [303 314]  [321 500]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]
```

```
Candidatas: 20->335  34->321  43->402
ITERACAO 40
OPERACAO:34   START TIME:321

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [311 322]  [335 516]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 301]  [303 305]  [303 314]  [321 332]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 20->335  35->402  43->402
ITERACAO 41
OPERACAO:20   START TIME:335

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [311 322]  [335 349]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 301]  [303 305]  [303 314]  [321 332]  [402 534]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [410 432]  [414 444]



Candidatas: 35->402  43->402
ITERACAO 42
OPERACAO:35   START TIME:402

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [311 322]  [335 349]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 301]  [303 305]  [303 314]  [321 332]  [402 437]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 429]  [457 432]  [461 444]
```

```
Candidatas: 43->402
ITERACAO 43
OPERACAO:43   START TIME:402

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [311 322]  [335 349]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 301]  [303 305]  [303 314]  [321 332]  [402 437]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 412]  [457 432]  [461 444]



Candidatas: 44->457
ITERACAO 44
OPERACAO:44   START TIME:457

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [311 322]  [335 349]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 301]  [303 305]  [303 314]  [321 332]  [402 437]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 412]  [457 462]  [461 444]



Candidatas: 45->461
ITERACAO 45
OPERACAO:45   START TIME:461

D1 [  0   5]  [  4  20]  [  4  20]  [ 19  30]  [ 70  84]
D2 [ 84  89]  [ 88 104]  [ 88 104]  [107 118]  [124 138]
D3 [128 133]  [132 148]  [132 148]  [161 172]  [234 248]
D4 [289 294]  [293 309]  [293 309]  [311 322]  [335 349]
H1 [ 12  20]  [ 19  21]  [ 19  30]  [ 69  80]  [ 79 114]
H2 [184 192]  [282 284]  [282 293]  [290 301]  [300 335]
H3 [293 301]  [303 305]  [303 314]  [321 332]  [402 437]
F  [ 88 100]  [108 115]  [114 118]  [171 193]
E1 [116 127]  [133 139]  [138 153]
E2 [402 412]  [457 462]  [461 474]
```

CARTA DE GANTT




R1
R2
R3
R4
R6
R7
F1
FP1
FP2
TRO
TRS
ELE
VAP
320
366
400
R1
R2
R3
R4
R6
R7
F1
FP1
FP2
TRO
TRS
ELE
VAP
400
480

APÊNDICE 5

```
/*=============================================================*/
/*EXEMPLO PROPOSTO POR EGLI & RIPPIN - 1986                    */
/*PROGRAMA DE SIMULACAO                                        */
/*=============================================================*/

/*=============================================================*/
/*                DEFINIÇÃO DAS VARIÁVEIS                      */
/*=============================================================*/


/* HEADERS */
#include <stdio.h>

/* DEFINICOES DE OPERADORES */
#ifndef ALLOC
#define ALLOC extern
#endif

/* PARAMETROS */
/* Numero maximo de tarefas (+1) */
#define T 11
/* Numero maximo de processadores (+1) */
#define P 12
/* Numero maximo de recursos */
#define R 2
/* Horizonte maximo de sequenciamento */
#define H 560
/* Numero maximo de operacoes (+1) */
#define O 46

/* MACROS */
#define max(A,B) ( (A)>(B)?(A):(B) )
#define min(A,B) ( (A)<(B)?(A):(B) )

void dados(void);
FILE *g_fopen(char* filename, char* mode);
void printgantt(int fim);
char produto(int op);

int fim_semana(int op);
void simulacao(void);
int starttime(int op, int inicio);

void change_perfil(char *mode, int op, int st, int pvt);
int in_op_cand(int c, int op);

void update_oper_win(char* extremo, int nop, int new_value);
```

```
/* VARIAVEIS GLOBAIS */

/* Numero de operacoes de cada tarefa */
ALLOC int NOP[T];
/* PLB[i]=1 -> Proc.i liberado; PLB[i]=0 -> Proc.i ocupado */
ALLOC int PLB[P];

/* Janelas atribuidas inicialmente a cada operacao */
ALLOC int W[T][2];
/* Capacidade dos recursos */
ALLOC int C[R];
/* Perfil de disponibilidade dos recursos */
ALLOC int A[R][H];
/* Consumo dos recursos por cada operacao */
ALLOC int CR[R][O][2];
/* Duracao do consumo dos recursos */
ALLOC int DR[R][O][2];
/* Tarefa a que pertence cada operacao */
ALLOC int TAR[O];
/* Numero da operacao na rota da tarefa */
ALLOC int OPE[O];
/* Processador responsavel por cada operacao */
ALLOC int PRO[O];
/* Operacao posterior de cada operacao */
ALLOC int POS[O];
/* Operacao anterior de cada operacao */
ALLOC int ANT[O];
/* Tempo de processamento de cada operacao */
ALLOC int TP[O];
/* Tempo de transferencia de cada produto */
ALLOC int TT[O];
/* Tempo de setup dos processadores-produtos */
ALLOC int SU[O][T];
/* Instante de inicio de cada operacao */
ALLOC int BT[O];
/* Instante de fim de cada operacao */
ALLOC int FT[O];
/* Condicao de relaxacao do FT de cada operacao */
ALLOC int RL[O];
/* Condicao de estabilidade de cada operacao */
ALLOC int SB[O];
/* Condicao de estabilidade auxiliar */
ALLOC int SBaux[O];
/* Folga de cada operacao */
ALLOC int FOLGA[O];
/* conjunto de operacoes sequenciadas */
ALLOC int PS[O];
/* Operacoes em Paralelo */
ALLOC int PL[O];
/* Inicio dos finais de semana */
ALLOC int WE[4];
/* Fim dos finais de semana */
ALLOC int DWE[4];
```

```
/*Variavel que diz se a janela de uma tarefa pode ser relaxada
*/
ALLOC int RELAX[T];

/* Batelada anterior de cada operacao */
ALLOC int B_ANT[O];

/* Ultima operacao realizada em cada processador */
ALLOC int UOP[P];

/* FIM DO RIPPIN.H */
```

```
/*=============================================================*/
/*                        ENTRADA DE DADOS                     */
/*=============================================================*/


/* INOUT.C */
/* FUNCOES DE ENTRADA E SAIDA DE DADOS */

/* INCLUDES */

#include "rippin.h"

/* CODIGO DAS FUNCOES */

FILE *g_fopen(char* filename, char* mode)
{
  FILE *fopen(), *fp;
  if((fp = fopen(filename,mode)) == NULL){
    printf("\n\nCannot open %s.\n\n",filename);
    exit(1);
  }
  return fp;
}/* FIM */

void dados(void)
{
  int i,j,k,l,ativ,tar,setup;
  FILE *filein;

  /* ENTRADA DE DADOS */

  /* abre o arquivo de dados */
  filein = g_fopen("egli.dat", "rt");

     /* carrega a capacidade dos recursos */
  for( k=0; k<R; k=k+1 )
    fscanf(filein,"%d",&C[k]);

  /* carrega os tempos da operacoes */
  l = 1;
  for( i=1; i<T; i=i+1 ){
    fscanf(filein,"%d",&NOP[i]);
    fscanf(filein,"%d %d %d",&W[i][0],&W[i][1],&RELAX[i]);
    for( j=1; j<=NOP[i]; j=j+1 ){
      fscanf(filein,"%d",&PRO[l]);
      fscanf(filein,"%d",&TP[l]);
      fscanf(filein,"%d",&TT[l]);
      fscanf(filein,"%d",&SB[l]);
      fscanf(filein,"%d",&B_ANT[l]);
      l = l + 1;
    }
  }
```

```
  /* verifica o numero de operacoes */

  if( (l-1)!=(O-1) ){
    printf("Inconsistencia dos dados de entrada!");
    exit(1);
  }

     /* carrega o consumo de recursos das operacoes */

  for( i=1; i<O; i=i+1 )
    for( j=0; j<=1; j=j+1 )
     for( k=0; k<R; k=k+1 ){
            fscanf(filein,"%d",&CR[k][i][j]);
            fscanf(filein,"%d",&DR[k][i][j]);
     }

     /* carrega os setup's das operacoes */

  while( 1 ){
    fscanf(filein,"%d %d %d",&ativ,&tar,&setup);
    if( ativ==-1 )
              break;
    else
      SU[ativ][tar] = setup;
  }

     /* fecha o arquivo de dados */

     fclose(filein);

  l = 1;

  /* dependencias entre operacoes */

  for( i=1; i<T; i=i+1 ){
    for( j=1; j<=NOP[i]; j=j+1 ){
      TAR[l] = i;
      OPE[l] = j;
      ANT[l] = l-1;
      POS[l] = l+1;
      if( j==1 )
        ANT[l] = 0;
      if( j==NOP[i] )
        POS[l] = 0;
      l = l + 1;
    }
  }

  return;


}/* FIM */
```

```
void printgantt(int fim)
{
  int i,j,k,t,s[P],ip,opp[T],flag,aux;
  char gantt[12][481];
  char equip[14][4];
  FILE *arq;

  for( i=0; i<P; i=i+1 )
     s[i] = 0;

  strcpy(equip[ 0],"    ");
  strcpy(equip[ 1],"R1  ");
  strcpy(equip[ 2],"R2  ");
  strcpy(equip[ 3],"R3  ");
  strcpy(equip[ 4],"R4  ");
  strcpy(equip[ 5],"R6  ");
  strcpy(equip[ 6],"R7  ");
  strcpy(equip[ 7],"F1  ");
  strcpy(equip[ 8],"FP1 ");
  strcpy(equip[ 9],"FP2 ");
  strcpy(equip[10],"TRO ");
  strcpy(equip[11],"TRS ");
  strcpy(equip[12],"ELE ");
  strcpy(equip[13],"VAP ");

  for( t=0; t<=fim; t=t+1)
    if( (t%10)==0 )
     gantt[0][t]='|';
    else if( (t%5)==0 )
     gantt[0][t]='+';
    else
     gantt[0][t]='-';
  for( j=1; j<P; j=j+1 ){
     k = 1;
     opp[0] = 0;
          for( i=1; i<O; i=i+1)
               if( PS[i]!=0 && PRO[i]==j ){
                    opp[k] = i;
                    k = k + 1;
               }
          for( i=k; i<T; i=i+1 )
               opp[i] = 0;
          flag = 1;

          /* ordenamento do vetor */
          while( flag==1 ){
               flag = 0;
               for( i=1; i<T-1 && opp[i+1]!=0; i=i+1 )
                    if( BT[opp[i+1]]<BT[opp[i]] ){
                         aux = opp[i+1];
                         opp[i+1] = opp[i];
                         opp[i] = aux;
                         flag = 1;
                    }
          }
```

```
    for( i=1; i<T && opp[i]!=0; i=i+1 ){
    for( t=s[j]; t<s[j]+SU[opp[i-1]][TAR[opp[i]]]; t=t+1 ){
         for( k=0; k<4; k=k+1 ){
              if( t>=WE[k] && t<DWE[k] ){
                   while( t<DWE[k] ){
                        gantt[j][t]='.';
                        t = t + 1;
                   }
                   s[j] = s[j] + (DWE[k]-WE[k]);
              }
         }
         gantt[j][t]='$';
    }
    s[j] = s[j]+SU[opp[i-1]][TAR[opp[i]]];
     for( t=s[j]; t<BT[opp[i]]; t=t+1 )
       gantt[j][t]='.';
     s[j] = BT[opp[i]];
     if(ANT[opp[i]]!=opp[i]-1 && TP[opp[i]]!=TP[PL[opp[i]]] ){
       for( t=s[j]; t<s[j] + TP[opp[i]] + TT[opp[i]] +
       TT[ANT[opp[i]]] + TT[PL[opp[i]]]; t=t+1 ){
         gantt[j][t]=produto(opp[i]);
       }
       s[j] = s[j] + TP[opp[i]] + TT[opp[i]] + TT[ANT[opp[i]]]
       + TT[PL[opp[i]]];
     }
     else{
       for( t=s[j]; t < s[j] + TP[opp[i]] + TT[opp[i]] +
       TT[ANT[opp[i]]]; t=t+1 ){
         gantt[j][t]=produto(opp[i]);
       }
       s[j] = s[j]+TP[opp[i]]+TT[opp[i]]+TT[ANT[opp[i]]];
     }
     ip = 0;
     for( k=i; k<O; k=k+1 )
         if( PS[k]>0 && k==POS[opp[i]] ){
         ip = k;
                        break;
                   }
     if( ip!=0 ){
       for( t=s[j]; t<BT[ip]+TT[opp[i]]; t=t+1 )
         gantt[j][t]='w';
         s[j] = max(s[j],BT[ip]+TT[opp[i]]);
       }
   }
   for( t=s[j]; t<=fim; t=t+1 )
    gantt[j][t]='.';
}

printf("\n");
printf("\nCARTA DE GANTT \n");

for( i=0; i<fim/80; i=i+1 ){
```

```
      for( t=0; t<4; t=t+1 )
           putc(equip[0][t],stdout);
      for( t=i*80; t<=(i+1)*80; t=t+1 )
           putc(gantt[0][t],stdout);
printf("\n");

      for( j=1; j<P; j=j+1 ){
           for( t=0; t<4; t=t+1 )
                putc(equip[j][t],stdout);
           for( t=i*80; t<=(i+1)*80; t=t+1 )
                putc(gantt[j][t],stdout);
      printf("\n");
      }

      for( t=0; t<4; t=t+1 )
           putc(equip[0][t],stdout);
      for( t=i*80; t<=(i+1)*80; t=t+1 )
           putc(gantt[0][t],stdout);
printf("\n");

      for( t=0; t<4; t=t+1 )
           putc(equip[12][t],stdout);
      for( t=i*80; t<=(i+1)*80; t=t+1 )
           if( (A[0][t]/5)<=9 )
                printf("%d",A[0][t]/5);
           else if( (A[0][t]/5)==10 )
                printf("%s","L");
           else
                printf("%s","*");
printf("\n");

      for( t=0; t<4; t=t+1 )
           putc(equip[13][t],stdout);
      for( t=i*80; t<=(i+1)*80; t=t+1 )
           if( (A[1][t]/40)<=9 )
                printf("%d",A[1][t]/40);
           else if( (A[1][t]/40)==10 )
                printf("%s","L");
           else
                printf("%s","*");
printf("\n");

      for( t=0; t<4; t=t+1 )
           putc(equip[0][t],stdout);
      for( t=i*80; t<=(i+1)*80; t=t+1 )
           putc(gantt[0][t],stdout);
printf("\n");
printf("\n");

}
```

```
   /* Inicio da impressao em arquivo */
arq = g_fopen("egli.out", "a+");
fprintf(arq,"\n");
fprintf(arq,"\nCARTA DE GANTT \n");
for( i=0; i<fim/80; i=i+1 ){

        for( t=0; t<4; t=t+1 )
             putc(equip[0][t],arq);
        for( t=i*80; t<=(i+1)*80; t=t+1 )
             putc(gantt[0][t],arq);
   fprintf(arq,"\n");

        for( j=1; j<P; j=j+1 ){
             for( t=0; t<4; t=t+1 )
                  putc(equip[j][t],arq);
             for( t=i*80; t<=(i+1)*80; t=t+1 )
                  putc(gantt[j][t],arq);
        fprintf(arq,"\n");
        }

        for( t=0; t<4; t=t+1 )
             putc(equip[0][t],arq);
        for( t=i*80; t<=(i+1)*80; t=t+1 )
             putc(gantt[0][t],arq);
   fprintf(arq,"\n");

        for( t=0; t<4; t=t+1 )
             putc(equip[12][t],arq);
        for( t=i*80; t<=(i+1)*80; t=t+1 )
             if( (A[0][t]/5)<=9 )
                  fprintf(arq,"%d",A[0][t]/5);
             else if( (A[0][t]/5)==10 )
                  fprintf(arq,"%s","L");
             else
                  fprintf(arq,"%s","*");
   fprintf(arq,"\n");

        for( t=0; t<4; t=t+1 )
             putc(equip[13][t],arq);
        for( t=i*80; t<=(i+1)*80; t=t+1 )
             if( (A[1][t]/40)<=9 )
                  fprintf(arq,"%d",A[1][t]/40);
             else if( (A[1][t]/40)==10 )
                  fprintf(arq,"%s","L");
             else
                  fprintf(arq,"%s","*");
   fprintf(arq,"\n");

        for( t=0; t<4; t=t+1 )
             putc(equip[0][t],arq);
        for( t=i*80; t<=(i+1)*80; t=t+1 )
             putc(gantt[0][t],arq);
   fprintf(arq,"\n");
   fprintf(arq,"\n");
   }
```

```
  /* Fim da impressao em arquivo */
     fclose(arq);
  return;
}

char produto(int op)
{
     if( TAR[op]==1)
          return 'A';
     if( TAR[op]==2)
          return 'A';
     if( TAR[op]==3)
          return 'A';
     if( TAR[op]==4)
          return 'A';
     if( TAR[op]==5)
          return 'D';
     if( TAR[op]==6)
          return 'D';
     if( TAR[op]==7)
          return 'D';
     if( TAR[op]==8)
          return 'C';
     if( TAR[op]==9)
          return 'B';
     if( TAR[op]==10)
          return 'E';
}

/* FIM DO ARQUIVO INOUT.C */
```

```
/*=================================================================*/
/*                        PROGRAMA PRINCIPAL                       */
/*=================================================================*/


/* DEFINICAO DE CONTROLE DE COMPILACAO */
#define ALLOC

/* BIBILIOTECAS */
#include "rippin.h"

/* CODIGO DAS FUNCOES */

int starttime(int op, int inicio)
/* Instantes de inicio das operacoes das tarefas nao-alocadas*/
{
     int i, j, flag, aux, st_op, st_op1, ct_op, pro_lib, w[2],
         opp[T], id_op;
     int ct_bat, bt_jan;
     /* um numero grande, so pra dizer que ta alocada */
     PS[op] = 1000;
     /* inicializacao de variaveis */
     SBaux[op] = 0;
     FOLGA[op] = 0;
     bt_jan = BT[op];
     if( inicio!=0 )
          BT[op] = inicio;

     /* Instante de inicio calculado em funcao da batela
        anterior
     */

     if( B_ANT[op]!=0 ){
          if( PL[op]!=0 && SB[PL[op]]==1 )
               ct_bat = BT[B_ANT[op]] + TP[B_ANT[op]] +
               TT[B_ANT[op]] + TT[B_ANT[op]-1] +
               SU[B_ANT[op]][TAR[op]] + TT[PL[op]];

          else
               ct_bat = BT[B_ANT[op]]+TP[B_ANT[op]] +
               TT[B_ANT[op]] + TT[ANT[B_ANT[op]]] +
               SU[B_ANT[op]][TAR[op]];


               /* Verifica se o completion time da batelada
anterior invadiu o fim de semana */
               for( j=0; j<4; j=j+1 ){
                    w[0] = ct_bat-WE[j];
                    w[1] = DWE[j]-BT[B_ANT[op]];
                    if( w[0]>0 && w[1]>0 )

               /* setup invadiu o i-esimo fim de semana */
                         ct_bat = ct_bat + (DWE[j]-WE[j]);
               }
```

```
            BT[op] = max(BT[op],ct_bat);
      }


      /* Instante de inicio calculado pela rota do produto */

      st_op=max(BT[ANT[op]]+TP[ANT[op]]+TT[ANT[ANT[op]]],BT[op])
;

      /* Ordena as operacoes que utilizam o processador de op */
      j = 1;
      opp[0] = 0;
      for( i=1; i<O; i=i+1)
            if( PS[i]!=0 && PRO[i]==PRO[op] ){
                  opp[j] = i;
                  j = j + 1;
            }
      for( i=j; i<T; i=i+1 )
            opp[i] = 0;
      flag = 1;
      while( flag==1 ){
            flag = 0;
            for( i=1; i<T-1 && opp[i+1]!=0; i=i+1 )
                  if( BT[opp[i+1]]<BT[opp[i]] ){
                        aux = opp[i+1];
                        opp[i+1] = opp[i];
                        opp[i] = aux;
                        flag = 1;
                  }
      }
      /* Acha a posica de op no vetor opp */
      id_op = 0;
      for( i=1; i<=T; i=i+1 )
            if( opp[i]==op ){
                  id_op = i;
                  break;
            }
      for( i=id_op; i<=T; i=i+1 ){
            /* Calcula o instante de liberacao do processador */
            pro_lib = 0;
            if( opp[i]!=0 ){
                  pro_lib = BT[opp[i-1]]+TP[opp[i-1]] +
                              TT[opp[i-1]]+TT[ANT[opp[i-1]]];
                  if( PS[POS[opp[i-1]]]!=0 )
                        pro_lib = max ( pro_lib,
                                    BT[POS[opp[i-1]]]+TT[opp[i-1]]);

                  /* Adiciona o setup */
                  pro_lib = pro_lib + SU[opp[i-1]][TAR[op]];

            /* Verifica se o setup invadiu o fim de semana */
                  for( j=0; j<4; j=j+1 ){
                        w[0] = pro_lib-WE[j];
                        w[1]=DWE[j] - (pro_lib -
                        SU[opp[i-1]][TAR[op]]);
```

```
                    if( w[0]>0 && w[1]>0 )
               /* setup invadiu o i-esimo fim de semana */
                         pro_lib = pro_lib + (DWE[j]-WE[j]);
               }
          }
          st_op = max(st_op,pro_lib);
          BT[op] = st_op;
          flag = 0;
          while( flag==0 ){
               st_op1 = max(st_op,fim_semana(op));
               if( inicio==0 || inicio>=bt_jan )
                    st_op = in_op_cand(st_op1,op);
               else
                    st_op = st_op1;
               if( st_op1==st_op && st_op1==BT[op])
                    flag = 1;
               else{
                    st_op1 = st_op;
                    BT[op] = st_op;
               }
          }

     /*Retorna com o valor original da janela (aquele que
estava no menu)*/
          BT[op] = bt_jan;

          if( opp[i+1]!=0 ){
          /* Operacao pode ser alocada entre duas outras */
               SBaux[op] = 1;

          /* Verifica se em st_op e' possivel alocar op sem
             empurrar i+1*/

               /* calcula o completion time de op */
               ct_op = st_op+TP[op]+TT[op]+TT[ANT[op]]
               + SU[op][TAR[opp[i+1]]];

          /* Verifica se o setup invadiu o fim de semana */
               for( j=0; j<4; j=j+1 ){
                    w[0] = ct_op-WE[j];
                    w[1] = DWE[j]-st_op;
                    if( w[0]>0 && w[1]>0 )

                    /* setup invadiu o i-esimo fim de semana */
                         ct_op = ct_op + (DWE[j]-WE[j]);
               }

               /* calculo da folga da operacao i */
               FOLGA[op] = BT[opp[i+1]] - ct_op;
               if( ct_op <= BT[opp[i+1]] ){
                    PS[op] = 0;
                    return st_op;
               }
```

```
                if( inicio!=0 ){
                     PS[op] = 0;
                     return bt_jan;
                }
           }
           else{
                PS[op] = 0;
                return st_op;
           }
           /* Se nao e' possivel alocar op entre i-1 e i+1,
              troca op por i+1 */

           aux = opp[i];
           opp[i] = opp[i+1];
           opp[i+1] = aux;

           /* pode ser que op nao esteja mais entre duas
              operacoes */

           SBaux[op] = 0;
      }
      PS[op] = 0;
      return st_op;
}/* FIM */

int finishtime(int op)
/* Instantes de fim das operacoes das tarefas nao-alocadas */
{
      int ft_op;
      ft_op = min(FT[POS[op]]-TP[POS[op]]-TT[POS[op]],FT[op]);
      return ft_op;
}/* FIM */


void simulacao()
/* Simulador da Planta */
{
      int
iteracao,i,j,k,op,inicio,opcao,fim,aux,flag,st,ct_ant,st_aux;
      int bt_ant,can[O];
      FILE *arq;

      /* Zera perfil de consumo de recursos */
      for( k=0; k<R; k=k+1 )
           for( j=0; j<H; j=j+1 )
                A[k][j] = 0;

      /* Zera o conjunto de operacoes candidatas */
      for( i=0; i<O; i=i+1 )
           can[i] = 0;

      for( i=0; i<O; i=i+1 ){
           /* Posicao inicial das operacoes na sequencia */
           PS[i] = 0;
      }
```

```
op = 0;
aux = 0;
iteracao = 1;
while( iteracao<=0){

     if( aux==0 ){
          printf("\nEntre com uma das opcoes abaixo: ");
          printf("\n     0->terminar ");
          printf("\n     1->alocar operacao ");
          printf("\n     2->imprimir ");
          printf("\nOpcao Escolhida: ");
          scanf("%d",&opcao);
     }
     if( opcao==0 )
          break;

     if( opcao==2 ){
          printf("\nEntre com o fim do intervalo (multiplo
          de 80): ");
          scanf("%d",&fim);
          printgantt(fim);
          printf("\nTermino da Impressao. \n");
          continue;
     }

     if( opcao==1 ){
          /* Conjunto de operacoes candidatas */
          arq = g_fopen("egli.out","a+");
          printf("\nPara retornar ao menu principal tecle
          '0'. \n");
          printf("\nCandidatas: ");
          fprintf(arq,"\nCandidatas: ");
          if( op!=0 && SB[op]==1 ){
               can[POS[op]] = 1;
               for( i=1; i<O; i=i+1)
               if( i!=POS[op] )
                    can[i] = 0;
               printf("%d->%d  ",POS[op],BT[POS[op]]);
               fprintf(arq,"%d->%d ",POS[op],BT[POS[op]]);
          }
          else if( op!=0 && PL[op]!=0 && PS[PL[op]]==0 ){
               can[PL[op]] = 1;
               for( i=1; i<O; i=i+1)
               if( i!=PL[op] )
                    can[i] = 0;
               printf("%d->%d  ",PL[op],BT[PL[op]]);
               fprintf(arq,"%d->%d  ",PL[op],BT[PL[op]]);
          }
          else{
               for( i=1; i<O; i=i+1 )
               if( PS[i]==0 ){
                         can[i] = 0;
```

```
/* verifica se o processador de i esta liberado */
                    if(SB[i]==1&&PLB[PRO[POS[i]]]==0)
                         can[i] = 0;
                    else if( PLB[PRO[i]]==0  ){
                         for( k=1; k<O; k=k+1 )
                         if(k!=i&&PRO[k]==PRO[i]&&
                         PS[k]!=0&&BT[i]<BT[k]&&
                         (ANT[i]==0 || PS[ANT[i]]!=0)
                         &&(B_ANT[i]==0||
                         PS[B_ANT[i]]!=0)&&
                         (POS[B_ANT[i]]==0||
                         PS[POS[B_ANT[i]]]!=0) ){
                              can[i] = 1;
                              printf("%d->%d  ", i,
                              BT[i]);
                              fprintf(arq,"%d->%d  ",
                              i, BT[i]);
                         }
                    }
                    else if((ANT[i]==0||PS[ANT[i]]>0)
                         &&(B_ANT[i]==0||
                         PS[B_ANT[i]]!=0) &&
                         (PL[i]==0||ANT[i]==i-1)){
                              can[i] = 1;
                              printf("%d->%d  ", i,
                              BT[i]);
                              fprintf(arq,"%d->%d  ",
                              i, BT[i]);
                    }
                    else
                         can[i] = 0;
               }
          }
          fclose(arq);
          printf("\nEntre com a %da. operacao:",iteracao);
          scanf("%d",&aux);
          if( aux==0 )
               continue;
}

/* Verifica se a operacao e' candidata */
if( can[aux]==0 ){
     printf("\nERRO: Operacao Invalida!\n");
     continue;
}
else{
     printf("Entre com o inicio %da.operacao:",
     iteracao);
     scanf("%d",&inicio);
     if( inicio<BT[aux] && RELAX[TAR[aux]]==0 ){
          printf("\nEsta tarefa nao pode ser
          relaxada.\n");
          continue;
     }
```

```
     st_aux = BT[aux];
     flag = 0;
     for( i=1; i<O; i=i+1 )
          if(aux!=i && PRO[aux]==PRO[i] && PS[i]!=0
          && POS[i]!=0 &&
          PS[POS[i]]==0 &&  inicio>BT[i] ){
               flag = 1;

/* Encontrou uma operacao que proibe o processador,
entao volta com o BT original de aux e quebra o loop
*/
               printf("\nEsta operacao nao pode ser
               alocada neste instante!\n");

               break;
          }


/* Volta para o inicio do While, pois esta operacao
nao pode ser alocada neste instante */
          if( flag== 1 )
               continue;
}

/* verifica se a operacao invadiu o final de semana*/
if( fim_semana(aux)!=0 ){
     printf("\nERRO: Operacao Invadiu o fim de
     semena!\n");
     BT[aux] = st_aux;
     continue;
}
else{

     /* Testa consistencia do instante de inicio
     proposto
     */

     if( inicio != BT[aux] )
          i = time_consistency(aux,inicio);
     else
          i = 1;
     if( i==1 )
          op = aux;
     else
          continue;
}

/* Atualizacao do Estado da planta */
PS[op] = iteracao;
BT[op] = inicio;
FT[op] = inicio + TP[op] + TT[op] + TT[ANT[op]];
if(PL[op]!=0&&TP[op]!=TP[PL[op]] && ANT[op]!=op-1 )
     FT[op] = FT[op] + TT[PL[op]];
if( POS[op]!=0 )
     PLB[PRO[op]] = 0;
```

```
     else
          PLB[PRO[op]] = 1;

     /* condicoes para liberacao do processador de op */
     if( ANT[op]!=0 ){
          PLB[PRO[ANT[op]]] = 1;
          for( k=1; k<O; k=k+1 )
               if(k!=ANT[op] && PRO[k]==PRO[ANT[op]] &&
               PS[k]!=0 && PS[POS[k]]==0 )
                    PLB[PRO[ANT[op]]] = 0;
          if( PL[ANT[op]]!=0 ){
               PLB[PRO[PL[ANT[op]]]] = 1;
               for( k=1; k<O; k=k+1 )
                    if(k!=PL[ANT[op]]&&
                    PRO[k]==PRO[PL[ANT[op]]]&&
                    PS[k]!=0 && PS[POS[k]]==0 )
                         PLB[PRO[PL[ANT[op]]]] = 0;
          }
     }

     if(PL[op]!=0 && PS[PL[op]]!=0 && TP[PL[op]]!=TP[op])
          PLB[PRO[PL[op]]] = 1;

     /* Acrescentar o consumo ao perfil de recursos */

     /* caso geral e caso 2 e 3 */
     if( PL[op]==0 || (PL[op]!=0 && TP[PL[op]]==TP[op]) )
          change_perfil("add",op,BT[op]+TT[ANT[op]],0);
     else /* caso 22 e 23 */
          change_perfil("add", op, BT[op] + TT[ANT[op]] +
          TT[PL[op]],0);

     if( PL[op]==0 || ( PL[op]!=0 && ANT[op]==op-1 ) )
          change_perfil("add",ANT[op],BT[op],1);
     if(PL[op]!=0&& TP[op]!=TP[PL[op]] && ANT[op]!=op-1)
          change_perfil ("add",PL[op], BT[op] +
          TT[ANT[op]], 1);

     /* Modificacao das janelas de tempo */
     for( i=1; i<O; i=i+1 )
/*Recalcula o BT[op] apenas se PL[op] nao foi sequenciada
*/
     if(  PS[i]==0 ){
          if(ANT[i]!=0&&ANT[i]==ANT[i-1]&&TP[i-1]==TP[i]){
               BT[i] = BT[i-1];
          }
          else if( ANT[i]!=0 && ANT[i]==ANT[i-1]&&
               TP[i-1]!=TP[i] ){
               if( PS[PL[i]]==0 )
                    BT[i] = starttime(i,0);
               if( BT[i]!=BT[i-1] ){
                    BT[i-1] = max(BT[i-1],BT[i]);
                    BT[i] = BT[i-1];
                    i = i - 2;
               }
```

```
                else{
                    BT[i+1]=BT[i] + TP[i] + TT[ANT[i]] +
                    TT[PL[i]];
                    BT[i+1] = starttime(i+1,BT[i+1]);
                    flag = 0;
                    while( flag==0 ){
                        BT[i+1]= max (BT[i+1],
                        fim_semana(i+1));
                        st = in_op_cand(BT[i+1],i+1);
                        if( BT[i+1]==st )
                            flag = 1;
                        else
                            BT[i+1] = st;
                    }
                    i = i + 1;
                }
            }
            else
                BT[i] = starttime(i,0);

            if( (SB[ANT[i]]==1 || SBaux[ANT[i]]==1) ){
                    ct_ant = BT[ANT[i]] + TP[ANT[i]] +
                    TT[ANT[ANT[i]]] + TT[ANT[i]];
                    if( SB[ANT[i]]==1 ){
                        if( BT[i]+TT[ANT[i]]>ct_ant ){
                            BT[ANT[i]] = BT[i] -
                            TP[ANT[i]]-TT[ANT[ANT[i]]];
                            BT[ANT[i]]= max (BT[ANT[i]],
                            fim_semana(ANT[i]));
                            i = i - 2;
                        }
                    }
                    else if(BT[i] + TT[ANT[i]] > ct_ant +
                        FOLGA[ANT[i]] ){

                /* qq valor muito grande */
                            bt_ant = 1000;

/* encontra a operacao que era feita imediatamente apos
   ANT[i] */
                        for( j=1; j<O; j=j+1 )
                            if( PRO[j]==PRO[ANT[i]] &&
                            PS[j]!=0&& BT[j]>BT[ANT[i]])

            /* O bt_ant tem que ser maior do que BT[j] para
               que a ANT[op] passe na frente de j no vetor
               de processador por isso soma-se TP[j] (um
               valor pequeno qualquer */

                                bt_ant= min (bt_ant,
                                BT[j]+TP[j]);
                        BT[ANT[i]]= max (bt_ant,
                        fim_semana(ANT[i]));
                        i = i - 2;
                    }
```

```
     /* A operacao i ja esta sequenciada, portanto se start
        time nao pode ser modificado. Esta situacao caracteriza
        uma sequencia infactivel. O programa portanto termina
        com infactibilidade */
                        if( PS[i+1]!=0 ){
                             printf("\n FIM DO PROGRAMA:
                             SEQUENCIA INFACTIVEL!\n");
                             exit(0);
                        }
                }
                if( PL[i]!=0 && PL[i]==(i-1) && TP[PL[i]]!=
                TP[i] ){
                     if(BT[i] + TT[ANT[i]] + TP[i] + TT[i] +
                     TT[PL[i]] > FT[i] )
                          printf("\n COMPLETION TIME DA TAREFA
                          %d MAIOR QUE SUA JANELA!\n",i);
                }
                else if( BT[i]+TT[ANT[i]]+TP[i]+TT[i]>FT[i] )
                     printf("\n COMPLETION TIME DA TAREFA %d
                     MAIOR QUE SUA JANELA!\n",i);

          }

          /* Imprime janelas das operacoes */
          for( i=1; i<O; i=i+1 ){
               printf("[%3d %3d]  ",BT[i],FT[i]);
               if( TAR[i]!=TAR[i+1] )
                    printf("\n");
          }
          /* Imprime janelas em arquivo */
          arq = g_fopen("egli.out","a+");
          fprintf(arq,"\nITERACAO %d",iteracao);
          fprintf(arq,"\nOPERACAO:%d START TIME: %d\n\n", op,
          BT[op]);
          for( i=1; i<O; i=i+1 ){
               fprintf(arq,"[%3d %3d]  ",BT[i],FT[i]);
               if( TAR[i]!=TAR[i+1] )
                    fprintf(arq,"\n");
          }
          fclose(arq);

          /* Atualiza o numero de iteracoes */
          iteracao = iteracao + 1;
     }
}

int time_consistency(int aux, int inicio)
{
          int i, j, flag, st, st_aux, st_pl, bt_aux, result,
          prox, bt_prox;
          int ct_atual, ct_ant, w[2];
```

```
/* Se aux possui uma paralela que ja foi alocada,
   entao aux tem que ser alocaca no mesmo instante da
   parelela */

if( PL[aux]!=0 && PS[PL[aux]]!=0 )
     if( inicio!=BT[PL[aux]] ){
          printf("\n1:A OPERACAO %d TEM QUE COMECAR
          EM %d\n",aux,BT[PL[aux]]);
          return 0;
     }

/* Se a tarefa que precede aux e' instavel, entao aux
   tem que comecar assim que termina a sua precedente
*/
if( SB[ANT[aux]]==1 )
 if( inicio!= BT[ANT[aux]] + TP[ANT[aux]] +
 TT[ANT[ANT[aux]]] ){
          printf("\n2:A OPERACAO %d TEM QUE COMECAR
          EM %d\n", aux, BT[ANT[aux]] + TP[ANT[aux]]
          + TT[ANT[ANT[aux]]]);
          return 0;
 }

/*Calcula o start time para aux a partir de inicio */

/* Calcula o start time (pela rota) da posterior das
   operacoes paralelas com tempos de processamento
   diferentes */

if( ANT[ANT[aux]]!=0&& ANT[ANT[aux]]==ANT[ANT[aux]-1]
&& TP[ANT[aux]]!=TP[ANT[aux]-1])
          st_aux = BT[ANT[aux]] + TP[ANT[aux]] +
          TT[ANT[ANT[aux]]] + TT[PL[ANT[aux]]];
else
          st_aux = inicio;
st_aux = max(st_aux,inicio);
st_aux = starttime(aux,st_aux);
if( inicio<BT[aux] && st_aux>inicio ){
     printf("\n3:INSTANTE INFACTIVEL POR ROTA,
     PROCESSADOR OU INDISPONIBILIDADE DE TEMPO\n");
     return 0;
}
/* Calculo do start time da operacao posterior a aux
*/

if( PL[aux]==0 || (PL[aux]==aux-1) ){
     st_pl = st_aux+TP[aux]+TT[ANT[aux]];;
     if( ANT[aux]!=0 && ANT[aux]==ANT[aux-1]
     && TP[aux]!=TP[aux-1] )
          st_pl = st_aux+TP[aux]+TT[ANT[aux]] +
          TT[PL[aux]];
     st_pl = starttime(POS[aux],st_pl);
}
else{
     st_pl = st_aux;
```

```
     st_pl = starttime(PL[aux],st_pl);
     st_aux = max(st_aux,st_pl);
}

/* Se aux e' instavel, entao a operacao que a sucede
   (ou paralela) tem que estar colada em aux */

if( ANT[aux]!=0 ){

/* Encontra a operacao que segue ANT[aux] */

prox = 0;

/* um valor muito grande */
bt_prox = 1000;
for( i=1; i<O; i=i+1 )
     if( PRO[i]==PRO[ANT[aux]] && PS[i]!=0 &&
     BT[i]>BT[ANT[aux]] ){
          if( BT[i]>BT[ANT[aux]] && BT[i]<bt_prox )
               prox = i;
               bt_prox = BT[i];
     }

/* Teste para ANT[aux] */
ct_atual = st_aux + TT[ANT[aux]] +
SU[ANT[aux]][TAR[prox]];
     for( j=0; j<4; j=j+1 ){
          w[0] = ct_atual-WE[j];
          w[1] = DWE[j]-(st_aux+TT[ANT[aux]]);
          if( w[0]>0 && w[1]>0 )

          /* setup invadiu o i-esimo fim de semana */
               ct_atual = ct_atual + (DWE[j]-WE[j]);
     }
ct_ant = BT[ANT[aux]] + TP[ANT[aux]] +
TT[ANT[ANT[aux]]] + TT[ANT[aux]]+
SU[ANT[aux]][TAR[prox]];
     for( j=0; j<4; j=j+1 ){
          w[0] = ct_ant-WE[j];
          w[1] = DWE[j]-BT[ANT[aux]];
          if( w[0]>0 && w[1]>0 )

          /* setup invadiu o i-esimo fim de semana */
               ct_ant = ct_ant + (DWE[j]-WE[j]);
     }
if( SBaux[ANT[aux]]==1 ){
     if( ct_atual - ct_ant > FOLGA[ANT[aux]] ){
          printf("\n4:INFACTIBILIDADE CAUSADA PELA
          OPERACAO %d\n",aux);
          return 0;
     }
}
}
```

```
/* Teste para aux */
if( SB[aux]==1 ){
     if( PL[aux]!=0 ){
          if( st_aux!=inicio || st_aux!=st_pl ){
               printf("\n5:A OPERACAO %d NAO PODE
               COMECAR NESTE INSTANTE\n",
               aux);
               return 0;
          }
          else if( SBaux[aux]==0 )
               return 1;
     }
     else{
          if(st_aux!=inicio || st_aux!=st_pl-TP[aux]-
          TT[ANT[aux]] ){
               printf("\n6:A OPERACAO %d TEM QUE
               COMECAR EM %d\n",
               aux,st_pl-TP[aux]-TT[ANT[aux]]);
               return 0;
          }
          else if( SBaux[aux]==0 )
               return 1;
     }
}
else{
     if( SBaux[aux]==1 ){
          if( PL[aux]!=0 && (PL[aux]!=aux-1) ){
               if( st_aux!=inicio || st_aux!=st_pl ){
                    printf("\n7:A OPERACAO %d TEM QUE
                    COMECAR EM %d\n",
                    aux,st_pl);
                    return 0;
               }
               else if( SBaux[aux]==0 )
                    return 1;
          }
          else{
               if( st_aux!=inicio || st_aux>st_pl -
               TP[aux] - TT[ANT[aux]]+FOLGA[aux] ){
                    printf("\n8:INFACTIBILIDADE
                    CAUSADA PELA OPERACAO %d\n",
                    aux);
                    return 0;
               }
               else if( SBaux[aux]==0 )
                    return 1;
          }
     }
     else
          return 1;
}
bt_aux = BT[aux];
BT[aux] = inicio;
result = time_consistency(POS[aux],st_pl);
BT[aux] = bt_aux;
```

```
          return result;
}

int main()

/* Programa principal */
{
  int i,j,k;

     /* Inicializacao das variaveis */
     for( i=0; i<O; i=i+1 ){
          SBaux[i] = 0;
          FOLGA[i] = 0;
          for( j=0; j<T; j=j+1 )
               SU[i][j] = 0;
     }

     /* entrada de dados */
     dados();

     /* inicializacao das janelas de tempo */
     for( i=0; i<O; i=i+1 ){
          if( ANT[i]==0 )
               BT[i] = W[TAR[i]][0];
          else
               BT[i] = 0;
          if( POS[i]==0 )
               FT[i] = W[TAR[i]][1];
          else
               FT[i] = H;
     }

     /* inicializacao da ocupacao dos processadores */
     for( i=0; i<P; i=i+1 ){
          PLB[i] = 1;
          UOP[i] = 0;
     }

     /* Operacoes em paralelo */
     ANT[3] = ANT[2];
     POS[2] = POS[3];
     ANT[8] = ANT[7];
     POS[7] = POS[8];
     ANT[13] = ANT[12];
     POS[12] = POS[13];
     ANT[18] = ANT[17];
     POS[17] = POS[18];
     ANT[23] = ANT[22];
     ANT[28] = ANT[27];
     ANT[33] = ANT[32];

     /* Operacoes em Paralelo */
     PL[2] = 3;
     PL[3] = 2;
     PL[7] = 8;
```

```
    PL[8] = 7;
    PL[12] = 13;
    PL[13] = 12;
    PL[17] = 18;
    PL[18] = 17;
    PL[22] = 23;
    PL[23] = 22;
    PL[27] = 28;
    PL[28] = 27;
    PL[32] = 33;
    PL[33] = 32;

    /* Instante de inicio e duracao dos finais de semana e
       feriados */

    WE[0]=30;  DWE[0]=66;
    WE[1]=198; DWE[1]=234;
    WE[2]=248; DWE[2]=282;
    WE[3]=366; DWE[3]=402;

  /* atualizacao das janelas das operacoes */
  for( i=1; i<O; i=i+1 )
        if( ANT[i]!=0 ){
            BT[i] = starttime(i,0);
            if( ANT[i+1]==ANT[i] && TP[i]==TP[i+1] ){
                BT[i+1] = BT[i];
                i = i + 1;
            }
            if( ANT[i+1]==ANT[i] && TP[i]!=TP[i+1] ){
                BT[i+1] = BT[i];
                BT[i+2] = BT[i+1]+TP[i+1]+TT[i]+TT[i-1];
                i = i + 2;
            }
        }

  for( i=O; i>=1; i=i-1 )
        if( POS[i]!=0 )
            FT[i] = finishtime(i);

  simulacao();
  printf("\nTerminacao normal do programa!\n");
  exit(0);
  return 1;
}/* FIM */

int fim_semana(int op)
{
    int i,w[2];
    for( i=0; i<4; i=i+1 ){
        if( ANT[op-1]==ANT[op] && TP[op]!=TP[op-1] ){
            w[0] = (BT[op] + TP[op] + TT[ANT[op]] +
            TT[PL[op]] + TT[op])-WE[i];
        }
        else
            w[0] = (BT[op]+TP[op]+TT[ANT[op]]+TT[op])-WE[i];
```

```
            w[1] = DWE[i]-BT[op];
            if( w[0]>0 && w[1]>0 )

                  /* Processamento invadiu o i-esimo fim de semana
                  */
                  return DWE[i];
      }
      /* Processamento nao invadiu nenhum fim de semana */
      return 0;
}/* FIM */

/* FIM DO ARQUIVO RIPPIN.C */
```

```
        /*=============================================*/
        /* FUNCOES DE MANIPULACAO DOS RECURSOS COMUNS */
        /*=============================================*/

 /* SHARE.C */

/* INCLUDES */

#include "rippin.h"

/* CODIGO DAS FUNCOES */

void change_perfil(char *mode, int op, int st, int pvt)
{
  int k,t;
  /* adiciona ao perfil o valor do recurso k consumido por op
*/
  if( strcmp(mode,"add")==0 ){
    for( k=0; k<R; k=k+1)
      for( t=st; t<st+DR[k][op][pvt]; t=t+1 )
        A[k][t] = A[k][t] + CR[k][op][pvt];
  }
  /* retira do perfil o valor do recurso k consumido por op */
  if( strcmp(mode,"del")==0 ){
    for( k=0; k<R; k=k+1)
      for( t=st; t<st+DR[k][op][pvt]; t=t+1 )
        A[k][t] = A[k][t] - CR[k][op][pvt];
  }
}

int in_op_cand(int st, int op)
/* Esta funcao calcula o instante mais cedo, a partir de st,
que o "op"  */
/* pode iniciar o seu processamento obedecendo a oferta de
recursos      */
{
  int k,t,tl,flag,lib;
  flag = 0;
     while( flag==0 ){
          flag = 1;
    for( k=0; k<R; k=k+1 ){
     for( t=st; t<st+DR[k][ANT[op]][1]; t=t+1 )
      if( CR[k][ANT[op]][1]+A[k][t]>C[k] ){
          flag = 0;
          st = t+1;
        break;
      }
      if( flag==0 )
          break;
    }
    if( flag==1 ){
      for( k=0; k<R; k=k+1 ){
          for( tl=t; tl<t+DR[k][op][0]; tl=tl+1 )
        if( CR[k][op][0]+A[k][tl]>C[k] ){
            flag = 0;
```

```
            st = st+1;
            break;
        }

        if( flag==0 )
          break;
     }
    }
     }
     return st;
}/* FIM */

/* FIM DO ARQUIVO SHARE.C */
```

159

REFERÊNCIAS BIBLIOGRÁFICAS

Baker, K. R., "Introduction to Sequencing and Scheduling", John Wiley and Sons, New York, 1974

Campos, M. F. D., "Sequenciamento e Alocação de Operações em Flow-shops com Restrições sobre os Recursos Compartilhados e sobre os Prazos de Entrega das Tarefas: Uma Abordagem de Busca orientada por Restrições", Tese de Mestrado, Unicamp, agosto de 1993

Egli, U. M., Rippin, D. W., "Short-Term Scheduling for Multiproduct Batch Chemical Plants", Comp. Chem. Eng., vol. 10, n. 4, pp. 303-325, 1986.

Graells,M., Espuña, A., Puigganer, L., "Modelling Framework for Scheduling and Planning of Batch Operations", Chemical Engineering Department, Universitat Politècnica de Catalunya, Barcelona, Spain, 1994.

Grossmann, I. E.,Quesada, I., Raman, R., Voudouris, V. T., "Mixed-Integer Optimization Techniques for the Design and Scheduling of Batch Processes", Department of Chemical Engineering and Design Research Center, Carnegie Mellon University, Pittsburgh, U.S.A.

Kondili, E., Pantelides, C. C., Sargent, R. W. H., "A General Algorithm for Short-Term Scheduling of Batch Operations - I. MILP Formulation", Comp. Chem. Eng., vol. 17, n. 2, pp. 211-227, 1993

Ku, H. M., Rajagopalan, D., Karimi, I., " Scheduling in Batch Processes", Chem. Eng. Prog., 83(8), pp. 35-45, 1987

Ku, H. M., Karimi, I., "Scheduling in Serial Multiproduct Batch Processes with Finite Interstage Storange: A Mixed Integer Linear Program Formulation", Ind. Eng. Chem. Res., vol. 27, n. 10, pp. 1840-1848, 1988

Ku, H. M., Karimi, I., "Scheduling in Serial Multiproduct Batch Processes with Due-Date Penalties", Ind. Eng. Chem. Res., vol. 29, n. 4, pp. 580-590, 1990

Reklaitis, G. V., "Review of Scheduling of Process Operation", AIChe Symposium Series, vol. 78, n. 214, pp. 119-133, 1982.

Rodrigues, M. T. M., "Seqüenciamento e Alocação de Operações em Flow Shops na Indústria Química com Restrições sobre os Recursos Compartilhados: Uma Abordagem de Busca Orientada por Restrições", Tese de Doutorado, Unicamp, agosto de 1992.

BIBLIOGRAFIA CONSUTADA

Birewar, D. B., Grossmann, I., Incorporating Scheduling in the Optimal Design of Multiproduct Batch Plants", Comp. Chem. Eng., vol. 13, n.12, pp. 141-161, 1989.

Birewar, D. B., Grossmann, I., "Simultaneous Production Planning and Scheduling in Multiproduct Batch Plants", Ind. Eng. Chem. Res., vol. 29, n. 4, pp. 570-580, 1990.

Campos, F., D. M., Latre, L. G., Rodrigues, M. T. M., Passos, C. A. S.. "Seqüenciamento de Flowshop's com Restrições sobre os Instantes de Alocação das Tarefas e sobre Recursos Compartilhados", XXV SBPO, Campinas-SP, Brasil, novembro 1993.

Djavdan, P., "Design of an On-Line Scheduling Strategy for a Combined Batch/Continuos Plant Using Simulation", European Symposium on Computer Aided Process Engineering-1, pp. S281-S288

Grossmann, I. E., Sargent, R. W. H., "Optimum Design of Multipurpose Chemical Plants", Ind. Eng. Chem. Proc. Des. Dev., vol. 18, n. 2, pp. 343-348, 1979.

Latre, L.G., Campos, M. D., Rodrigues, M. T., Passos, C.A.S.. "A Dispatch Rule with Rolling Prediction Horizon for Chemical Flowshops", Conference on Computer Integrated

Manufacturing in the Process Industries (CIMPRO'94), pp 102-117, New Brunswick, USA, maio de 1994.

Latre, L.G., Rodrigues, M. T., Passos, C.A.S.. "Scheduling de Flowshops na Indústria Química. Técnicas de Busca Orientada pelas Restrições", Anais do 1º Simpósio Brasileiro de Automação Inteligente, pp. 393-402, Rio Claro, Brasil, set. 1993.

Knopf, F. C., Okos, M. R., Reklaitis, G. V., "Öptimal Design of Batch/Semicontinuous Processes", Ind. Eng. Chem. Proc. Des. Dev., vol.21, pp. 79-86, 1982.

Lázaro, M., Espuña, A., Puigjaner, L., "Ä Comprehensive Approach to Production Planning in Multipurpose Batch Plants", Comp. Chem. Eng., vol.13, n.9, pp. 1031-1047, 1989.

Mauderli, A., Rippin, D. W. T., "Production Planning and Scheduling for Multi-Purpose Batch Chemical Plants", Comp. Chem. Eng., vol.3, pp.199-206, 1979

Mauderli, A., Rippin, D. W. T., "Scheduling Production in Multi-Purpose Batch Plants: The Batchman Program", Chem. Eng. Prog., vol.76, n.4, pp.37-45, 1980.

Passos, C. A. S., Latre, L. G., Rodrigues, M. T. M., Campos, M. F. D., "Seqüenciamento de Plantas Químicas Multiprodutos: Uma Abordagem Multi Nível", Congresso Brasileiro de Automática, SBA, Rio de Janeiro, 1994.

Rippin, D. W. T., "Batch Process Systems Engineering: A Retrospective and Prospective Review", Comp. Chem. Eng, vol. 17. Suppl., pp.S1-S536, 1993

Rodrigues, M. T., Passos, C. A. S., Latre, L. G., Campos, M. F. D., "Flow Shop Scheduling: A Resource Based Branch and Bound Perspective", 12th IFAC World Congress, Sydney, Austrália, july 1993.

Shah, N., Pantelides, C. C., "Design of Multipurpose Batch Plants with Uncertain Production Requirements", Ind. Eng. Chem. Res., vol 31, pp. 1325-1337, 1992.

Suhami, I., Mah, R. S. H., "Optimal Design of Multipurpose Batch Plant", Ind. Eng. Chem. Proc. Des. Dev., vol. 21, pp. 94-100, 1982.

Esta versão corresponde à redação final da Tese de Mestrado, defendida pela Engenheira Ana Claudia Gondim de Medeiros, e aprovada pela Comissão Julgadora em 18 de abril de 1995.

Orientadora: Profª. Drª. Maria Teresa Moreira Rodrigues

ABSTRACT

The development in the last years of the fine chemical industries created a new challenge for the chemical engineers: the scheduling of batch plants.

These class of plants have some important features: the great number of products that can be processed in the same processing unit and they share the same resources as electricity, manpower etc.

The main problem is: how to share the time taking into account the production level desired for the different products and the availability of resources like equipaments etc, optimizing a performance criterium. This problem has a combinatorial nature and has been classified as "a computational hard problem", that is, to find an optimal solution could be very difficult and time expensive. In order to obtain a good solution, some time there are constraint that could be relaxed, or a heuristic approach is used.

In this thesis is proposed a simulation approach developed using the well known problem stated by Egli and Rippin. In the proposed approach determined the earliest beginning time for the tasks not yet allocated in order to maintain the schedule feasibility.

The proposed approach is divided into two levels: in the first one "demand window" is determined for each task taking into account the raw material availability and the demand pattern. In the second level is defined the earliest beginning time for each task without relaxing the constraints.